AN EVALUATION FUNCTION

FOR SIMPLE KING AND PAWN ENDINGS

by

Leon Piasetski

A thesis submitted to the Faculty
of Graduate Studies and Research,
in partial fulfillment of the
requirements for the degree of
Master of Science.

School of Computer Science
McGill University
Montreal

June 1977

ABSTRACT

We have designed an evaluation function for each of three simple King and Pawn chess endings. In the case of King and Pawn vs. King, the function determines the correct result for every possible situation, within a short series of instructions. In cases with just two pawns, each function correctly evaluates a fairly large number of positions. The functions are implemented as subroutines in a chess-playing program for general King and Pawn endings, created, in part, by the author's thesis advisor, Professor Monroe Newborn. This reduces the tree search through α-β cut-offs and enables the program to find the right move for the right reason in considerably less time than usual.

The program was tested on a variety of endgame studies for King and Pawn vs. King and Pawn. The results show that improvement in the program's play must be directly attributed to the evaluation functions. Furthermore the same basic approach may be applied in complex endings.

RESUME

Une fonction d'évaluation a été conçue pour chacune de trois simples fins de partie du jeu d'échecs comportant uniquement des rois et des pions. Dans le cas d'un roi et d'un pion contre un roi, la fonction détermine, à l'aide d'une série d'instructions, le résultat correct pour toutes les situations possibles. Pour les positions incluant exactement 2 pions, chaque fonction évalue un assez grand nombre de positions. Ces fonctions sont pratiquement réalisées comme sous-programmes d'un programme de jeux d'echecs écrit pour les fins de partie comportant des rois et des pions en général, créé en partie par le conseiller de thèse de l'auteur, le professeur M. Newborn. Ceci réduit le scrutement de l'arbre des possibilités en créant des coupures α-β et rend le programme capable de trouver le mouvement correct pour la bonne raison en un temps considérablement plus court que d'ordinaire. Le programme a été examiné en l'appliquant sur une variété d'études de fin de partie comportant un roi et un pion contre un roi et un pion. Les résultats montrent que l'amélioration dans la qualité de jeu du programme doit être directement attribuée aux fonctions d'évaluation. De plus, la même méthode de base de l'approche peut être appliquée pour des fins de partie plus compliquées.

ACKNOWLEDGEMENTS

I would like to express sincere gratitude to my thesis advisor, Professor Monroe Newborn, for his encouragement, guidance and patience during the research and preparation of this thesis.

Special thanks go to my fellow students whose dependability and good humour created a conducive atmosphere for solving program bugs.

I am truly indebted to my wonderful typist, Donna-Marie Marosi, for her careful preparation of this thesis.

Finally I would like to thank my family for their support and patience.

CONTENTS

	page
ABSTRACT	i
ACKOWLEDGEMENTS	iii
CONTENTS	iv
LIST OF FIGURES	vi
1.0 INTRODUCTION	1
1.1 Historical Survey	3
1.2 Several Examples	7
2.0 PROGRAM ORGANIZATION	14
2.1 Description of ONEPAN	14
2.2 Description of TWOPAN	19
2.3 Description of PAWN2	21
3.0 EXPERIMENTAL RESULTS	33
3.1 Blocked Pawn Positions	35
3.2 Adjacent Pawn Positions	49
3.3 Passed Pawn Positions	62
3.4 Summary	87a
4.0 FINAL OBSERVATIONS AND CONCLUSION	88
4.1 Improving PAWN2's Performance	89
4.2 General Remarks on Pawn Endings	91
4.3 Conclusion	92

BIBLIOGRAPHY 93

APPENDICES

1. Listing of the program 94
2. Sample trees 166
3. Testing of ONEPAN and sample of results ... 174

LIST OF FIGURES

page

Fig. I-1..... 6

Fig. I-2..... 6

Fig. I-3..... 8

Fig. I-4..... 8

Fig. I-5..... 10

Fig. I-6..... 10

Fig. I-7..... 12

Fig. II-1..... 15

Fig. II-2..... 15

Fig. II-3..... 15

Fig. II-4..... 15

Fig. II-5..... 17

Fig. II-6..... 17

Fig. II-7..... 17

Fig. II-8..... 18

Fig. II-9..... 18

Fig. II-10..... 18

Fig. II-11..... 22

Fig. II-12..... 22

Fig. II-13..... 24

Fig. II-14..... 24

Fig. II-15..... 24

Fig. II-16..... 24

Fig. II-17..... 25

Fig. II-18..... 25

Fig. II-19..... 25
Fig. II-20..... 26
Fig. II-21..... 26
Fig. II-22..... 27
Fig. II-23..... 27
Fig. II-24..... 27
Fig. II-25..... 28
Fig. II-26..... 28
Fig. 1..... 34
Fig. 2..... 36
Fig. 3..... 38
Fig. 4..... 40
Fig. 5..... 40
Fig. 6..... 42
Fig. 7..... 44
Fig. 8..... 46
Fig. 9..... 48
Fig. 10..... 50
Fig. 11..... 50
Fig. 12..... 52
Fig. 13..... 54
Fig. 14..... 54
Fig. 15..... 57
Fig. 16..... 59
Fig. 17..... 61
Fig. 18..... 63

Fig. 19..... 63

Fig. 20..... 65

Fig. 21..... 68

Fig. 22..... 70

Fig. 23..... 72

Fig. 24..... 74

Fig. 25..... 76

Fig. 26..... 78

Fig. 27..... 80

Fig. 28..... 82

Fig. 29..... 84

Fig. 30..... 86

INTRODUCTION

1.0

In the age of the Renaissance it was still possible for one man to grasp all the accumulated wisdom of his predecessors. This of course is no longer true. Modern man must devote his entire efforts to master but one specific branch of knowledge. Yet we have not completely lost the image of "Renaissance Man". Today, in order to solve the increasingly complex problems of our expanding universe, specialists must be trained in other unrelated fields.

Artificial intelligence is primarily directed towards constructing machines that display intelligence. Through the interplay of tactical vision, strategic reasoning and memory, the game of chess has long provided a sufficient challenge to man's intellect. Thus computer researchers have spent much time and energy designing programs that play chess well. Despite their efforts the level of computer play is still far below master chess. This is probably because most programs have been written by computer scientists with little chess experience. Indeed the percentage of programs written or developed in close collaboration with chess players of expert or better strength is exceedingly small. We may expect further significant advances in this field as stronger chess players become involved in the research.

Chess endings offer a unique opportunity to both players and programmers. Although for the chess player, it is less difficult to establish 'truth' in an endgame position, art and imagination are not forsaken. For the computer scientist, the tree search becomes far more reasonable; thus with fewer pieces and clearer

goals the tree can grow long and sparse. Chess masters most often display their superiority over lesser mortals in the endgame. On the other hand, computer programs,designed to play an entire game of chess, have considerably greater problems here (see 1.2). Nowhere is this more true than in pawn endings. Thus it is surprising that little investigation has taken place in this field (see 1.1). We are aware of only one program which was specifically conceived to play general king and pawn endgames. This program, called PEASANT[7], was developed,in part, by the author's thesis advisor, Professor Monroe Newborn, and serves as the touchstone for further research in this field.

Our thesis presents a description of an evaluation function for each of three types of positions

(a) King and Pawn vs. King,

(b) King and two Pawns vs. King and

(c) King and Pawn vs. King and Pawn.

These functions, called ONEPAN, TWOPAN and PAWN2 respectively, are included as subroutines in PEASANT (see Appendix 1). Many more nodes in the search tree are correctly evaluated because the regular processing machinery in such positions has been replaced by these functions.

In the rest of Chapter I we provide an historical survey of computer research in chess endgames and analyse a few examples of program play taken from recent tournament praxis. In Chapter II we describe the program in some detail. In Chapter III we present 30 'problems', culled from Pawn Endings[1], and discuss the program's performance at various search depths in each position. Finally, in Chapter IV we offer some observations on the results and conclude with a few general remarks about the future of computer chess in this area.

HISTORICAL SURVEY

1.1

The first chess endgame program, written by Huberman [3], was designed to play only certain positions. The principle goal of her research was to investigate the process of translating book descriptions of problem solving methods into computer program heuristics. Clearly won endings with little material (e.g. King and Rook vs. King) were chosen for the task area. In order to limit the tree search, the functions better and worse were introduced. The results indicate that in such endgames it is possible to simply choose a good move, instead of the best, without seriously extending the number of moves required to checkmate with best play. It is hard to say whether further research into translation methods for chess books would be rewarding. To begin with, almost every rule in chess has an exception; much of the literature is limited to analysis of master play in specific cases. It is nearly as difficult to transform (sophisticated) chess knowledge into book form (i.e. language) as it is to translate the book descriptions into program heuristics. Moreover, existing book representations would likely be too complex for direct translation. However, it is vitally important to make as much use as possible of book knowledge to develop improved heuristics.

Although the objective in master level chess is still to checkmate, most players resign a hopeless situation if they believe their opponents are capable of discovering the winning continuation. Thus the majority of players would resign a lost King and Rook vs. King position. On the basis of our experience

we would argue that choosing a better, but not best move, in a position that is 'just winning', will, in general, not only lengthen the number of moves required to win but also significantly increase the difficulties in selecting the following move, after the opponent's reply. In simpler terms, the next time around there may be only one winning move and it will probably be more difficult to determine that it wins!

Further investigation in the field of endgame play was carried on by Tan [8]. The purpose of his reseach was to study how knowledge might be represented, organized and used in the task area of single-pawn endings (i.e. King and Pawn vs. King). In the program developed by Tan to play such positions, knowledge is represented as associations between predicates and action schemes and organized to form a binary tree. Predicates are the classification rules used in the decision tree and correspond to its non-terminal nodes. Each action scheme determines a legal move applicable to a given situation. A terminal node consists of an ordered pair <V,A> where V (value) indicates whether the position is won, drawn or undefined and A defines an action scheme. The program proceeds through the decision tree until a terminal node is reached. If the value V of this node is undefined, the game tree is searched until a position is reached that does return a value V, won or drawn, from its decision tree. Such a position matches a known situation or 'pattern'. Part of the chess knowledge in Tan's program is represented by these patterns, which are simply positions where the result, win or draw, is easily determined. (If the square occupied by the pawn is considered as the origin of a coordinate system, the other

pieces form a 'type pattern', which, in general, is not altered by shifting the pawn's square, while maintaining the same relation between the pieces in the coordinate system.) The predicates and action schemes, described earlier, are the chess heuristics which determine when and if it is necessary to generate more moves in the game tree.

Tan's approach would be difficult to apply to more complicated endgames. Selecting adequate predicates and organizing the decision tree would be too time consuming, even for a chess master, because there is no systematic procedure to follow. Also, the program requires too much time in the single pawn ending for it to handle more complex situations.

A recent report by Michalski and Negri [5] describes the results of an experiment in inductive learning. The goal of their research was to establish classification rules which distinguish won and drawn positions in the single pawn ending by applying an inductive inference program, previously developed, to a set of learning events. These events were generated by taking examples of the pattern positions discussed in Tan's program. The preliminary results obtained were satisfactory and the method certainly deserves further investigation, particularly in more complicated endgames. Nevertheless it is debatable whether the process would eventually yield 'stable' decision rules in a reasonable time.

As yet PEASANT [7] is the only chess program designed to play general pawn endings. The original purpose of the research was to determine whether the poor playing ability of ordinary tournament chess programs in the endgame was due to a basic

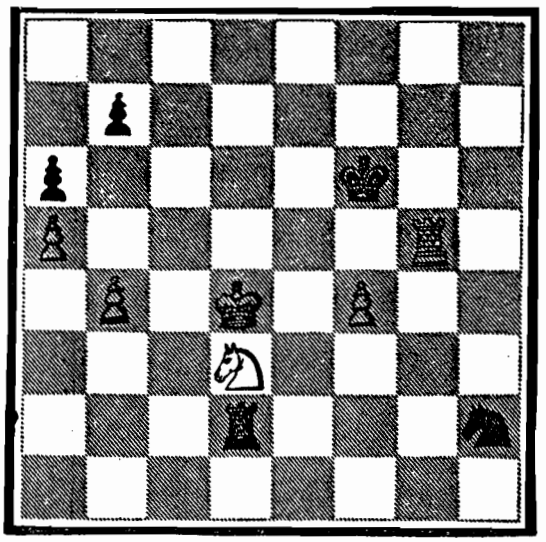


Fig. I-1. Position after 78. K-d4.

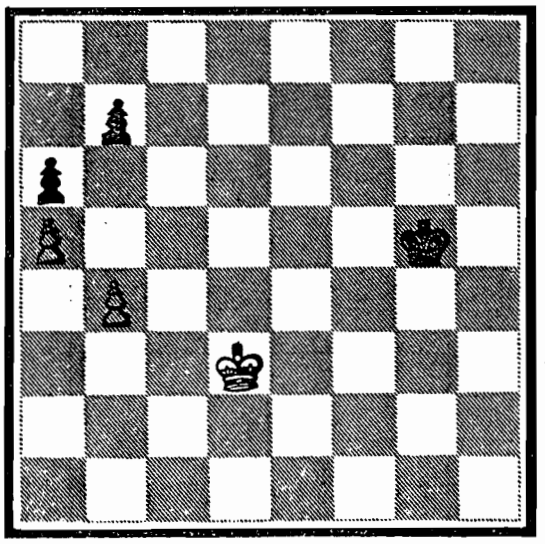


Fig. I-2. Position after 81. ... K:g5.

weakness in the Shannon-Turing minimax approach or to the lack of custom-tailored chess heuristics. PEASANT, based on a conventional α-β minimax search procedure plus tree-pruning heuristics, was developed to examine this question. The program was tested on several examples of varying difficulty from Fine [2] and the results signified that a respectable level of play had been attained. Furthermore there remained considerable room for improvement within the Shannon-Turing structure. A detailed history of general chess-playing computer programs may be found in Newborn [6].

SEVERAL EXAMPLES OF PROGRAM PLAY

1.2

It seems inevitable that as the level of competition in computer chess tournaments rises, a higher proportion of game results will be determined in the endgame. In the meantime we can present only three examples of poor computer play in king and pawn endings.

To begin with we consider the position after White's 78th move in the game Coko III vs. J. Biit from the first United States computer chess championship in 1970. See Fig. I-1. Black would lose after either 78. ... R-a2 79. R-c5 intending R-c7:b7 or 78. ... R-d1 79. K-c3 with R-c5 to follow. Instead Black (wisely! in retrospect) chose to win a pawn and transposed into a lost pawn ending. See Fig. I-2. The game continued 78. ... N-f3+ 79. K-e3 R:d3+! 80. K:d3 N:g5 81. f:g5+ K:g5 82. K-e3? K-f5 83. K-f3 K-e5 84. K-e3 K-d5 85. K-d3 K-e5 86. K-e3 K-d5 87. K-d3 K-e5 and was drawn by repetition.

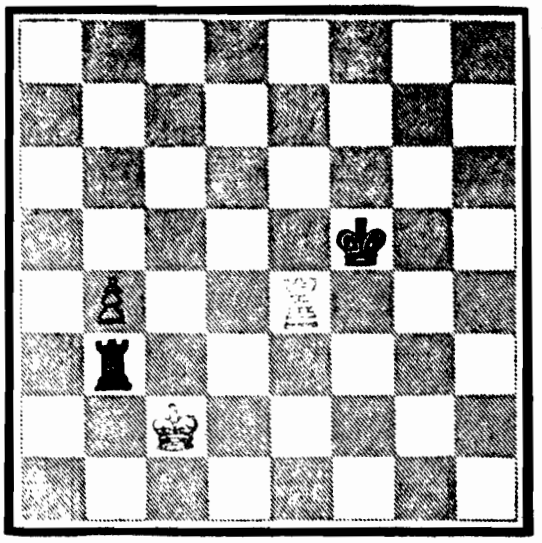


Fig. I-3. Position after 129. K-c2

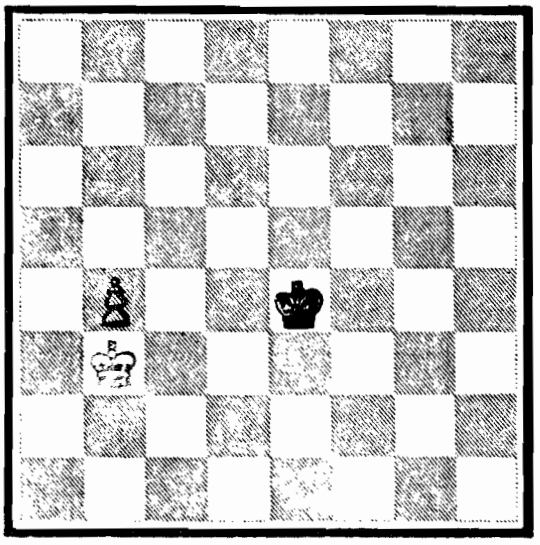


Fig. I-4. Position after 130. K:b3

White could have won by attacking the defenseless queenside pawns with either 82. K-d4 or 82. K-c4 intending K-c5-b6:b7 but this would require a search of about 10 ply. We would at least expect White to select one of these two moves on the basis of general principles i.e. advance toward pawns and cut-off enemy king from pawns. The actual move played allowed Black's king to return quickly and defend his pawns. Black would have had more difficulties after 83. K-d4 but K-f5-c6-d7-c8 would still guarantee a draw. Note that White avoided 83. K-e2?? and 83. K-f2?? which both lose.

The next position arose during a marathon struggle between Tech and Coko III in the third United States computer championship in 1972. White had kept a pawn advantage for nearly 100 moves (!) in a rook ending, while avoiding, at the last possible moment each time, 12 draws by repetition!! See Fig. I-3. Black should now play 129. ... R-h3 with a draw according to an analysis by Grigoriev [4]. Instead Black exchanged rooks into a lost pawn ending. See Fig. I-4. Play continued 129. ... K:e4? 130. K:b3 K-d4?! 131. b5? K-c5 132. K-a4 K-b6 133. K-b4 K-c7 134. K-c5 K-c8?? 135. K-c6! K-d8(?) 136. K-b7 K-d7 137. b6 K-e7?! 138. K-c7 and White mated 7 moves later. Black could have tried 130. ... K-d5 but after 131. K-a4! K-c6 132. K-a5 K-b7 133. K-b5 White wins with direct opposition. By blundering on his 131st move (131. K-a4!) White allowed an easy draw, which Black, in turn, threw away with 134. ... K-c8?? instead of 134. ... K-b7. Note that both 135. K-c6 and 135. K-b6 win for White and that 135. ... K-b8 would have prolonged the struggle but not

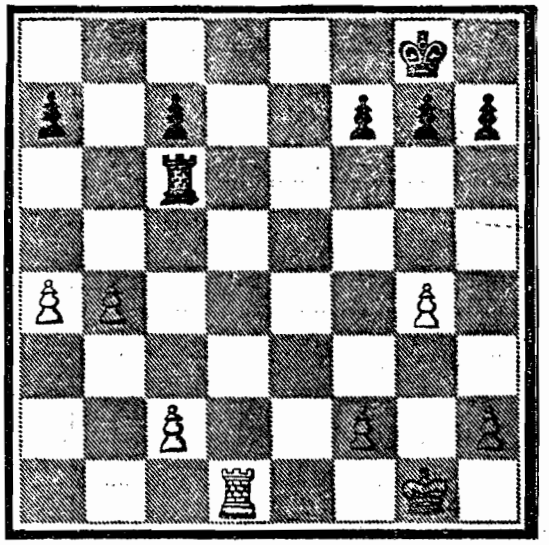


Fig. I-5. Position after 24. R-d1

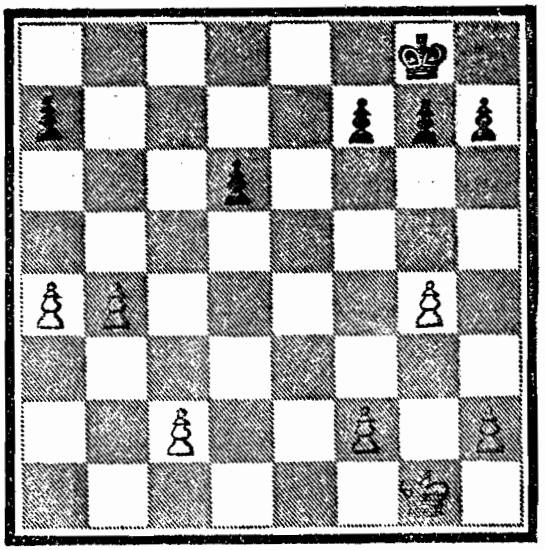


Fig. I-6 Position after 25. ... c:d6

altered the result (unless White were to blunder again!). An incredible performance...

Our last example occurred in the game Ostrich vs. Tech from the fourth United States computer championship in 1973. See Fig. I-5. White has an extra pawn and threatens mate but Black could have simply answered 24. ... K-f8 with an easy draw; the double threat of 25. ... R:c2 and 25. ... R-c4 forking two pawns would soon re-establish material equality. Instead, as in both previous examples, Black chose to trade off into a lost king and pawn ending. See Fig. I-6. The game proceeded 24. ... R-d6? 25. R:d6 c:d6 26. h4 d5 27. c3 g6 28. a5! K-g7 29. f4 K-f6! 30. K-f2 a6(?) 31. K-e3 K-e6 32. h5 f6 33. K-d4! K-d6 34. h6 g5 35. K-e3 K-e6 36. f5+ K-d6 37. K-f3 d4(?) 38. c4 K-e5 39. b5! a:b 40. a:b d3 41. a6! d2 42. K-e2 and White mated 10 moves later. Actually both sides committed a series of blunders. After the exchange of rooks White should have realized his winning advantage in the swiftest manner by forcing a pawn through to queen. Since Black's king was far enough away this would only have required advancing the a and b pawns and, if necessary, sacrificing one to queen the other e.g. 26. b5 K-f8 27. a5 K-e7 28. b6 a:b 29. a6! etc. (potential passed pawn). However, because of his material superiority, White's position was still totally won. Black's 30th move was particularly poor since it permitted White to force through a queen in one less move i.e. 31. b5! and in three more moves White queens; with the pawn on a7 White must play b5-b6. Yet it was not clear that White would have found the road to victory had Black not 'forced the play' with 37. ... d4(?). Note as well that both sides

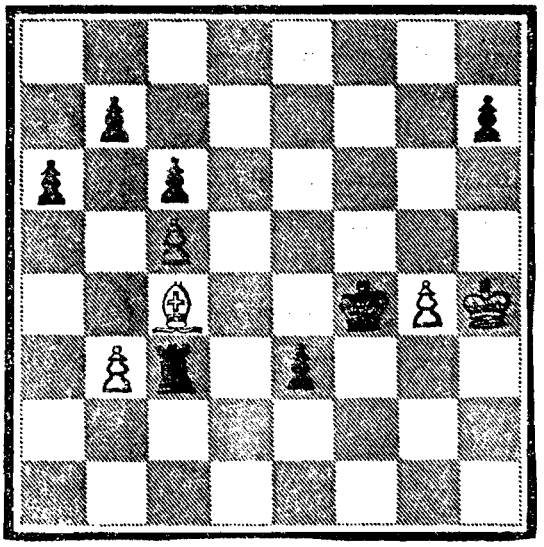


Fig. I-7. Position after 47. B-c4.

temporarily ignored the 'free' passed pawn created after 39. ... a:b .

Finally we present a pleasing example of correct play which involves a transition to a king and pawn ending. The following position arose in the game J. Biit vs. Chess 3.0 from the first United States computer championship. See Fig. I-7. Black finished off neatly with 47. ... R:c4! 48. b:c e2 49. g5 e1(=Q)+ 50. K-h5 Q-h1 mate! It is only fair to point out that most programs would solve this position; a brute search of 5 plies clearly results in a won game because Black queens by force.None of the programs cited in this section had significant endgame heuristics. Furthermore Coko III and J. Biit were not even fully debugged.

A more complete discription of the games noted here may be found in Newborn[6].

PROGRAM ORGANIZATION

2.0

PEASANT is designed to perform a standard tree search, to some arbitrary fixed depth (at most 14 plies), based on an α-β minimax strategy. Specific endgame heuristics eliminate many unnecessary king moves through forward pruning (see PEASANT[7]). At each node in the tree the program applies a series of tests to determine whether that node is terminal and, if so, what score to assign it. The subroutines ONEPAN, TWOPAN and PAWN2, are implemented at each node as static evaluation functions for positions with one or two pawns. ONEPAN correctly evaluates any position with exactly one pawn on the board whereas the other two routines correctly evaluate some positions with exactly two pawns on the board.

ONEPAN

2.1

In ONEPAN we use both a distance test and a classification scheme to solve every possible situation with king and pawn vs. king. The program first checks that the defending Black king is 'within the square' of the enemy pawn. This simple distance test establishes that the pawn may not advance with impunity to the queening square without the aid of the White king (ie. if Black's king is not within the square, White wins). Next the program classifies the position according to the relative x-y coordinates of the defending king and the enemy pawn. Let the White pawn be defined by (x_1, y_1) and the Black king by (x_2, y_2); then the difference in width ($W=|x_1 - x_2|$) and the difference in height ($H= y_2 - y_1$) are used to categorize the position.

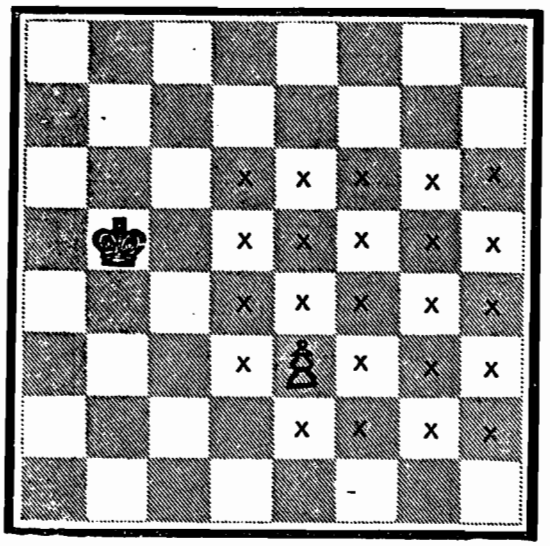


Fig. II-1. Black to play.

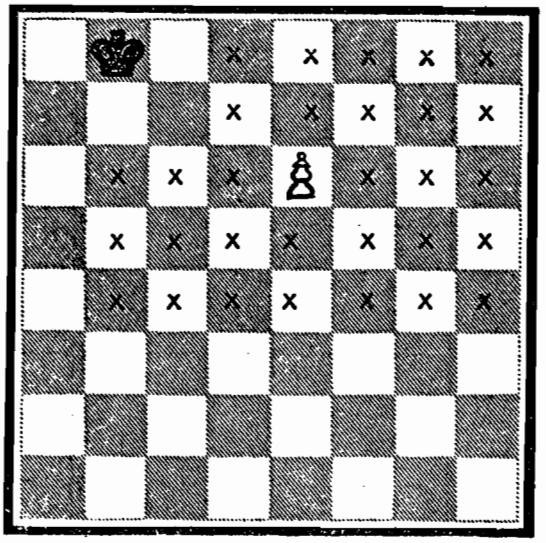


Fig. II-2. Black to play.

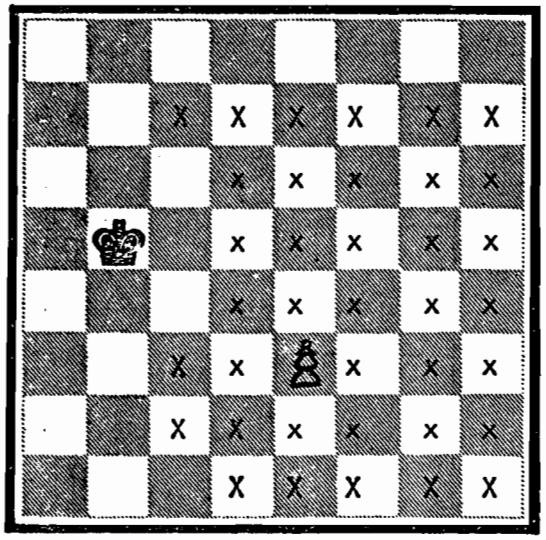


Fig. II-3. White to play.

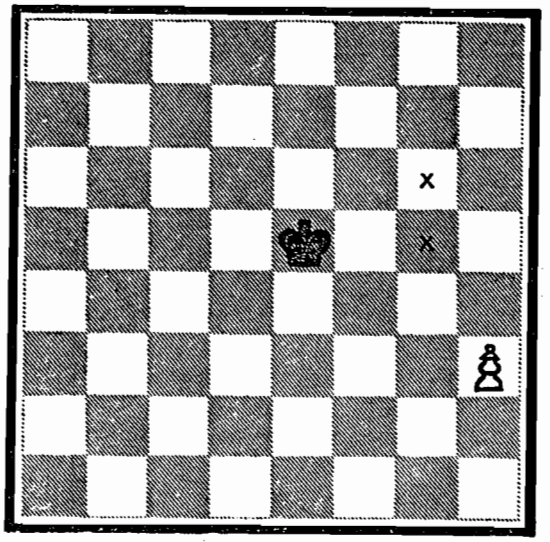


Fig. II-4. Black to play.

Figures 1 and 3 are associated; if the White king in Fig. II-3 occupies one of the 'large X' squares on the perimeter, then White can move to one of the squares marked x in Fig. II-1.

Finally, if the White king occupies a square within a specified region R, the position is won. Otherwise it is drawn.

For example, we consider a situation with a White pawn at e3 and the Black king at b5; then White can win only if his king occupies one of the squares marked by an x in Fig. II-1. This same 'mask' is used if the pawn and Black king are on squares e4,b6 or g3,d5 respectively, etc. Thus the marked squares form a mask which we associate with a specific W_1 (3) and H_1 (2). There are many exceptions to this many-one relationship. For instance, with the W_1 and H_1 of Fig. II-1 (ie.3,2) but with the pawn on the sixth rank, we must use another mask. See Fig. II-2. Naturally we must have different masks for whoever is to move (see Fig. II-3) and for the exceptional case of the White rookpawn (see Fig. II-4). Altogether about 300 masks, covering all possible piece configurations, are employed by ONEPAN to resolve the situation accurately. These were arrived at painstakingly, case by case (see Appendix 3).

There are two types of normalization used in classifying positions. Firstly, because W is defined as an absolute distance measure, the program uses the same mask whether the Black king is on one side of the pawn or the other; the mask (ie. marked squares) is shifted horizontally about the axis of the pawn's file. Secondly we make no distinction between pawns on the b,c,d,e,f or g files, which all use the masks designed for the general case. The only exceptions are rook pawns on the a or h files which both use the masks designed for this special case. The set of masks (a product of the author's chess knowledge) is complete in that any position, with king and pawn vs. king, submitted to the program is classified and scored accurately.

If the position is won for White it is assigned a score of +6000 -200*d -400*P -100*(1-c) + K where d is the depth (in

plies), P is the number of moves to the promotion square, c is equal to 1 if White is on the move, and -1 otherwise, and K is a function of the distance of each king to the pawn which credits bringing the king nearer to the pawn. The score of a drawn position is set equal to K. The score function, excluding K, is identical to the one used in PEASANT[7] . The rationale behind the term -200*d is to differentiate between nodes of identical positions by decreasing the score of the one found at a deeper level in the tree ie. thus determining the shortest branch to the position (eg. the quickest win) . Similarly the terms -400*P and -100*(1-c) are included to stimulate pawn pushing and to distinguish between nodes on the basis of which colour is to move in the position respectively. The function K is made up of three terms ie. K= 7 + Max($|x_1-x_2|$, $|y_1-y_2|$) - Max($|x_3-x_4|$, $|y_3-y_4|$) where (x_1,y_1) and (x_3,y_3) are the squares of the Black king and White king respectively, (x_2,y_2) is the square of the White pawn and (x_4,y_4) is the queening square of the same pawn. The quickest win for White is achieved by controlling the queening square; the quickest draw for Black is gained by capturing the pawn. We add 7 to avoid letting the function K equal zero, a score reserved for 'dead' drawn positions eg. no pawns.

ONEPAN initializes local variables, tests 'within the square', tests some exceptional situations and calls one of two subroutines, RKPAWN or THPAWN, to classify (and score) any other position. The first routine handles all possible rook pawn positions with H varying between -1 and 6 and W varying between 0 and 6. It also takes care of positions with regular pawns with H equal to -1 and W varying between 0 and 6. THPAWN handles the rest of the regular pawn cases with H varying between 0

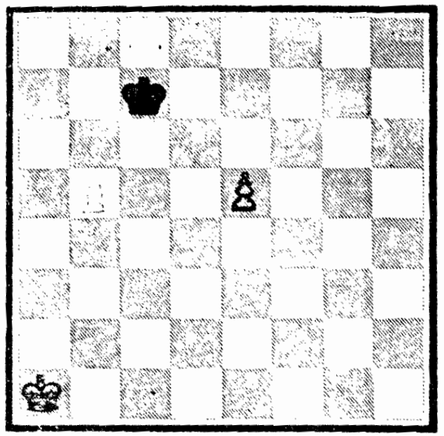


Fig. II-5. White to play.

White wins easily after 1. e6 K-d6 2. b6 K:e6 3. b7 or 2. ... K-c6 3. d7 etc. Even with Black to play, White wins here. 1. ... K-b6 2. e6 etc.

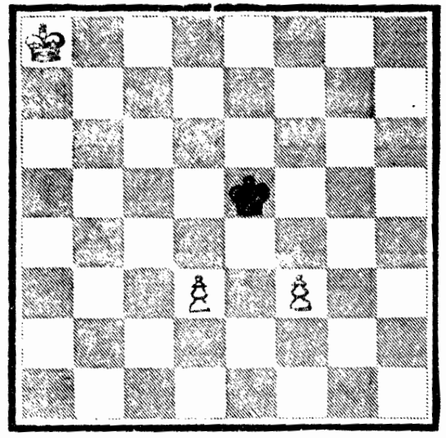


Fig. II-6. Black to play.

If Black attacks either pawn, the other advances. 1. ... K-d4 2. f4 and Black cannot capture or the f-pawns queens. If then 2. ... K-d5 3. K-b7 K-e6 4. d4 and the pawns defend each other again as in Fig. II-6.

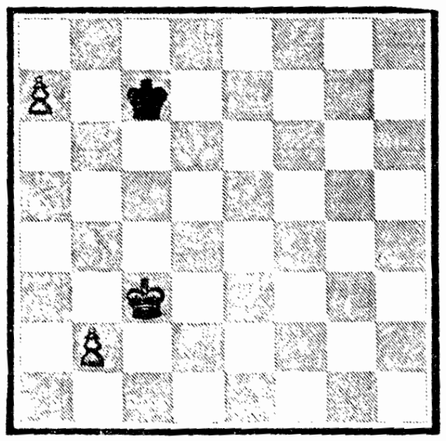


Fig. II-7. Black to play.

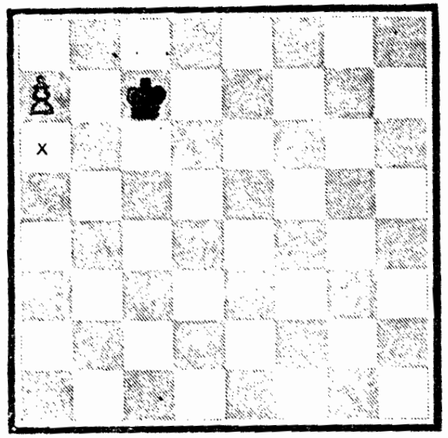


Fig. II-8. Black to play.

Thus the White king on c3 does not win here.

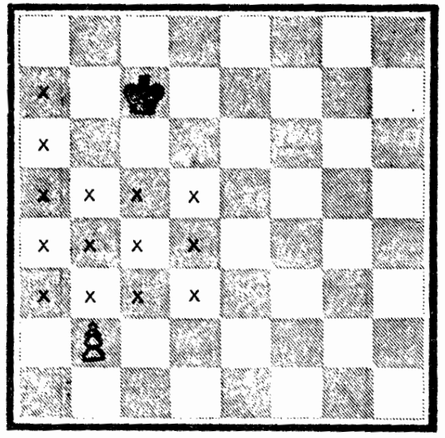


Fig. II-9. Black to play.

Thus ONEPAN scores this position as a win with the White king on c3 (and then so does TWOPAN).

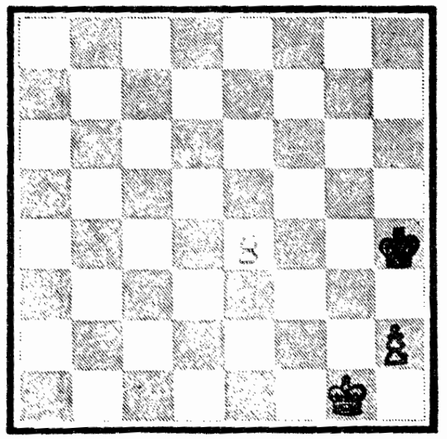


Fig. II-10. White to play.

White wins after 1. Kf2 Kg4 2. K-e3 but if the Black king occupied g4 1. Kf2? K-f4 would only draw! (1. K-g2 Kf4 2. K-h3! wins).

and 6, and W varying between 0 and 6. In either case just a short series of tests is needed to classify and score a position. Since both subroutines are designed to categorize only positions with a White pawn, ONEPAN must 'reverse' the board situation for a Black pawn. This is done at the very beginning of ONEPAN, while initializing local variables for the x-y coordinates of the pieces.

TWOPAN

2.2

Subroutine TWOPAN tests only positions with two pawns of the same colour. The program will consider such a node terminal

(1) if either pawn can advance and queen without being captured,

(2) if the pawns form a 'dynamic duo', where one is sacrificed to queen the other by force i.e. without any aid from the White king, one pawn will queen (see Fig. II-5),

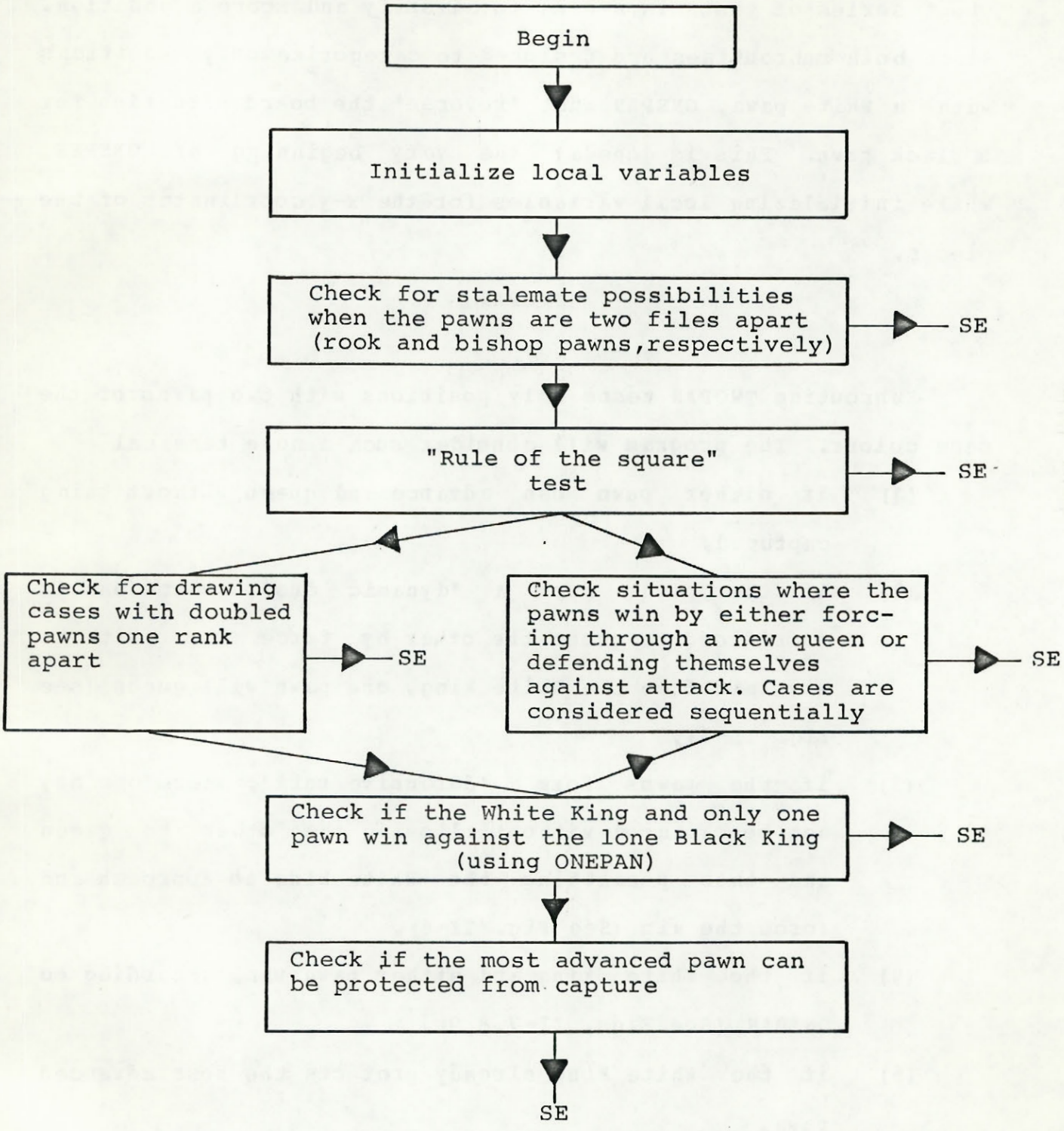
(3) if the pawns form a 'defensive unit', where one may not be captured without allowing the other to queen and thus permitting the White king to approach and force the win (See Fig. II-6),

(4) if the White king and either pawn win, according to ONEPAN (See Figs. II-7,8,9),

(5) if the White king already protects the most advanced pawn,

(6) if the White king can protect the most advanced pawn before Black captures it (See Fig. II-10),

(7) if the pawns and Black king stalemate the White king!,



N.B. S-Scores (a) positions that are won, lost or drawn and
(b) undecided situations at maximum depth.

E-End follows immediately
ie. SE implies S → E

Control of flow in subroutine TWOPAN

(8) in certain exceptional cases with the pawns doubled and three files apart and

(9) at maximum preset search depth.

A terminal node is scored as a win in the first six categories, a draw in the next two and undecided in the last. The scoring function employed is virtually the same as in ONEPAN, except that undecided positions are assigned a score of 25+K, where K was defined earlier. The setup for initializing local variables is also similar in design to that used in ONEPAN.

PAWN2

2.3

Subroutine PAWN2 examines only positions with two pawns of opposite colour. The setup here is somewhat different than in ONEPAN. The program accepts any board position without change but searches for a win only for White. Then, to check Black's winning potential, the board situation is reversed, by calling subroutine REVERS, and the same search process begins again. In another minor switch, a node will not automatically be considered terminal at maximum preset search depth e.g. if a king attacks an undefended pawn, search is extended (but not indefinitely). Also, terminal nodes, at maximum depth or greater, which are not evaluated as a win, loss or draw, are scored using a special function call PCHK. This function distinguishes between positions where either White, Black, both sides or neither can claim at least a draw. IF Black has at least a draw but no clear win, the position is assigned a score of -5+K, where K was defined earlier. On the other hand, if neither side can claim at

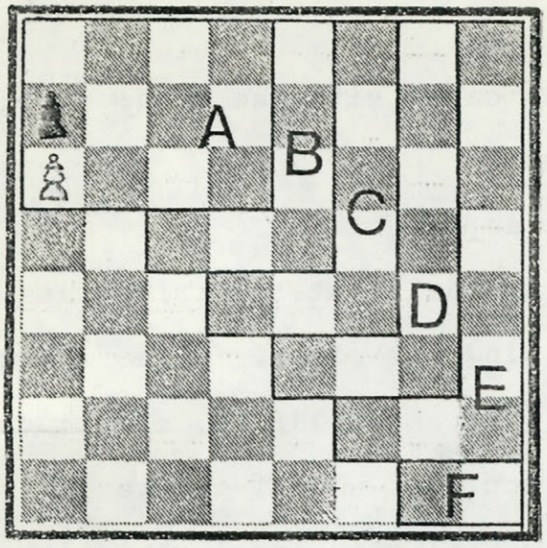


Fig. II-11

If the White king is one square away from the enemy pawn, with White to play, Black's king must occupy a square in Region A to draw for sure. If it is Black to move, the king's range increases to Region (A+B). Next, if the White king is two squares from the enemy pawn with White to play, Black's king's range is again Region (A+B); with Black to play, Region (A+B+C) etc. This was discovered by considering (in sequence) White's king on the h1-a8 diagonal.

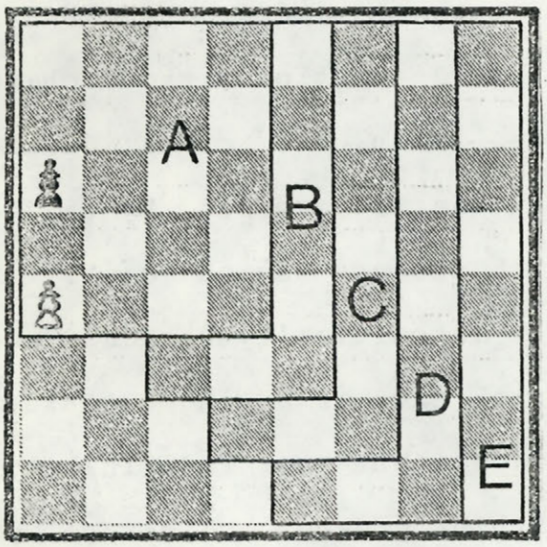


Fig. II-12.

Notice that the region increases if we lower the White pawn. This is because Black can attack the enemy pawn.

least a draw, the score is set equal to K. IF both sides or at least White can claim a draw, the score is set equal to +5+K. Note that a clearly drawn position is assigned a score of zero. Finally, in the scoring function for a win, we substitute the function P_F , which depends on the number of moves to the promotion square for <u>each</u> pawn, for the variable P defined earlier.

Subroutine PAWN2 is divided into three main sections, characterized by blocked, adjacent and passed pawns respectively. In the first category both pawns are on the same file, hence blocked, and neither may advance to queen unless the opposing pawn is captured by the king. We consider blocked pawns

(a) on the rook file,

(b) locked and thus unable to move, and

(c) not locked.

In the exceptional case of blocked rook-pawns the program cannot determine whether a position is won, lost or drawn. Thus a node is considered terminal only if the search terminates at maximum depth or greater. We have developed a function which determines if one side has at least a draw. However to evaluate this type of position as a clear win or draw we must extend the search until a pawn is captured. See Figs. II-11,12. The program does solve many critical positions in which the pawns are blocked and locked against each other. We first check whether or not White can force the capture of the enemy pawn, by examining each king's distance from the pawn. If the pawn is on or above the fifth rank, capturing will win against any defense. Otherwise, to win White must capture and avoid the 'opposition'

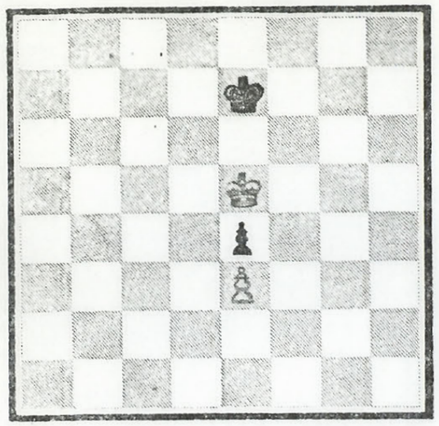


Fig. II-13. White to play.

If White captures on e4, Black plays K-e6 with a draw.

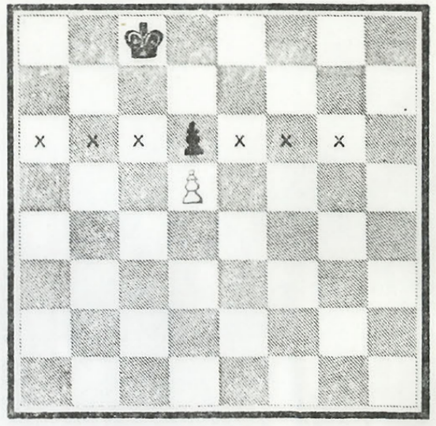


Fig. II-14.

If White occupies a square marked x the position is won.

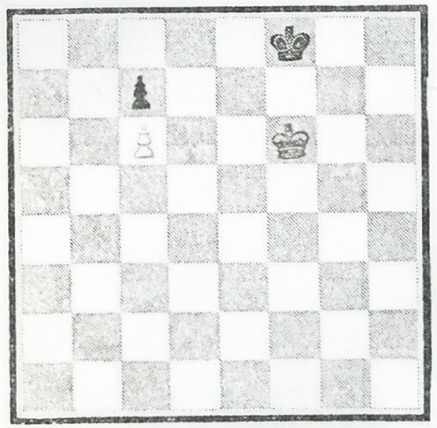


Fig. II-15.

Black to play loses after 1. ... K-e8 2. K-e6 K-d8 3. K-f7 as in Fig. II-14. White to play only draws i.e. 1. K-e6 K-e8 and Black keeps the opposition.

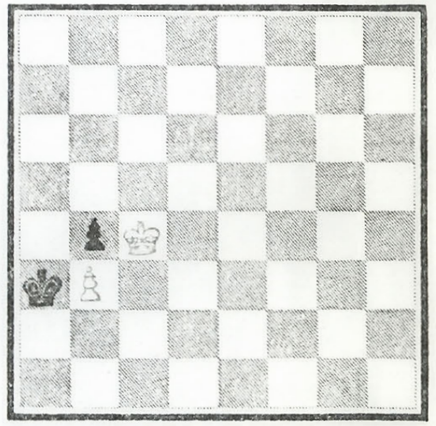


Fig. II-16.

Whoever moves loses!

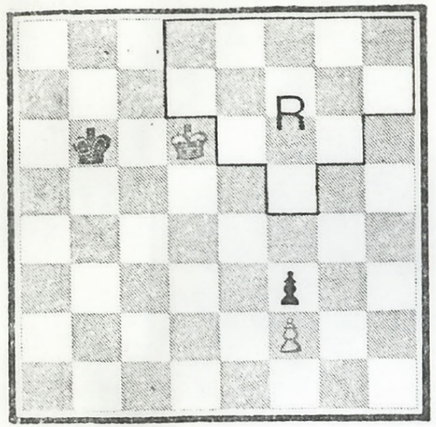


Fig. II-17. White wins.

If White has the upper hand in sideways opposition, and White's king does not occupy a square in Region R, White wins whoever is on the move. e.g. 1. ... K-b7 2. K-e5 K-c6 3. K-f4 K-d5 4. K:f3. But if White's king is on e6 and Black's on c6, then 1. K-e5 K-d7 2. K-f4 K-e6 3. K:f3 K-f5 only draws.

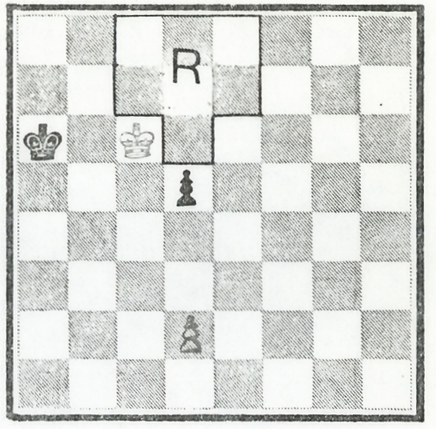


Fig. II-18. Black to play.

For blocked, but not locked pawn positions with White to play, we use the region exemplified in Fig. II-17 to determine whether the position is won. Here, with Black to play, a defensive manoeuvre involving the advance of Black's pawn increases Region R. Thus 1. ... Pd4 2. P-d3 wins as before, but with the kings on c7,a7 respectively, Black draws. i.e. 1. ... d4 2. K-c6 d3 3. K-c5 K-b7 4. K-d4 K-c6 etc.

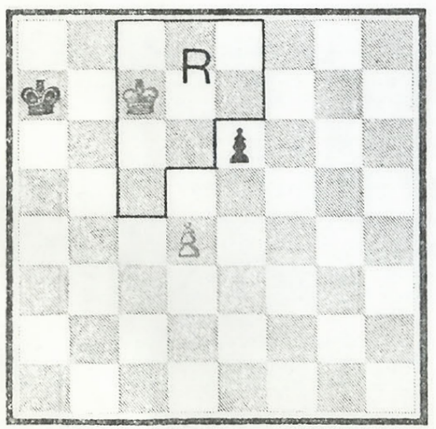


Fig. II-19.

If White occupies a square in Region R with the upper hand in sideways opposition, whoever moves White wins. The same is true in the next two figures.

that occurs in Fig. II-13. Next we note some special cases with the pawn on or above the fifth rank. For example, White wins if the White king occupies one of the 'critical squares' marked by an x in Fig. II-14 and the Black king lies below his pawn's rank (as in the diagram). Also, the program checks for the type of opposition which is exhibited in Fig. II-15. Then we probe situations where both kings are close to the pawns. See Fig. II-16. Finally the program considers another form of opposition ('sideways') displayed in Fig. II-17. Note that in all of these tests, aside from the distance check, the program attempts to match a given position with a known 'pattern' situation. We apply a similar approach to blocked pawns that are not locked. The program scans positions for pattern matches and sideways opposition, uses a distance test and determines whether one side has at least a draw. See Fig. 18.

Adjacent pawns (i.e. one file apart) are treated in much the same fashion. Whereas the exceptionally difficult rookpawn case is given scant attention, the program uses a distance test and checks for sideways opposition as before. See Figs. II-19-21.

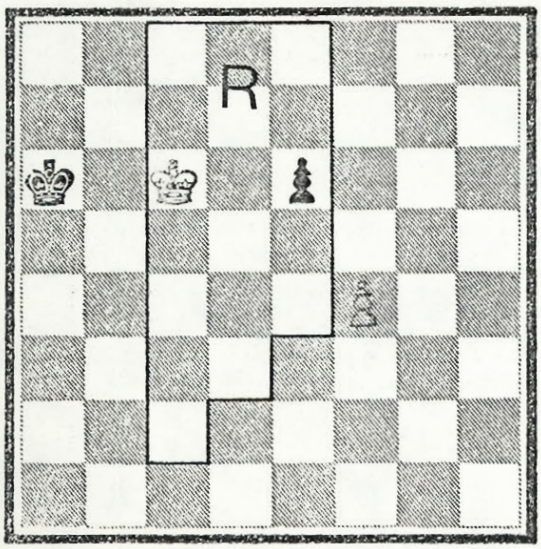


Fig. II-20.

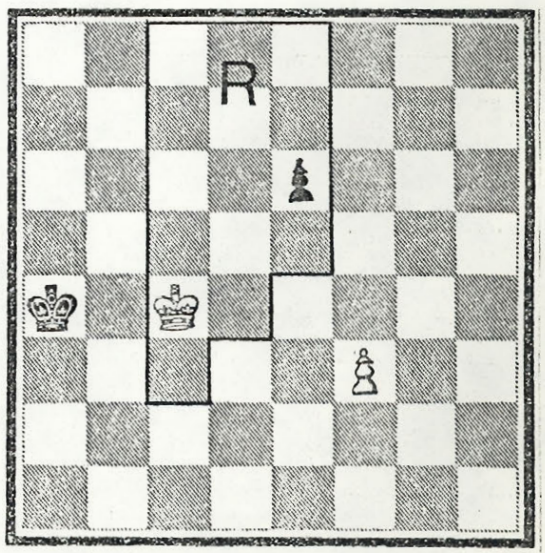


Fig. II-21.

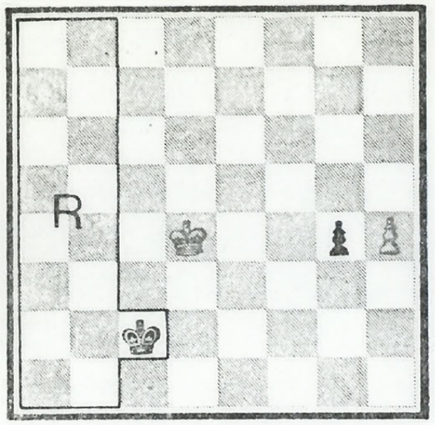


Fig. II-22.

With the Black king in Region R, White can safely advance his pawn. Here, even with Black to play, 1. ...K-d2 2. K-e4 K-e2 3. K-f4 White has no problems. But we must also check White's pawn as in the next diagram.

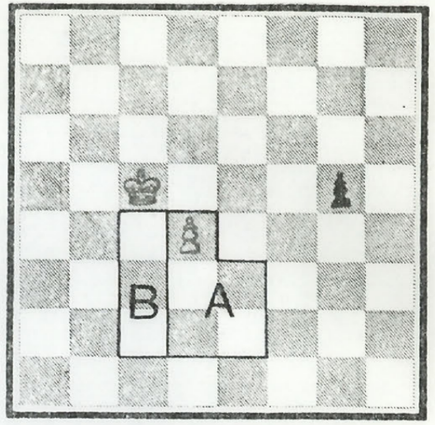


Fig. II-23

A White pawn in Region A would obstruct the White king's march to an advancing enemy pawn. This would also be true for Region B, unless White is on the move and the Black king is an 'extra' move outside the 'square' of the White pawn. Note that both kings must be on the same side of the Black pawn.

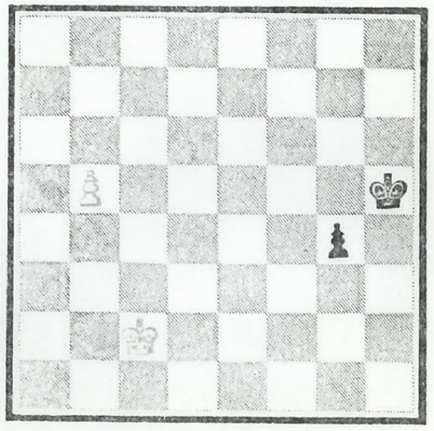


Fig. II-24. White wins.

White to play queens first and wins Black's queen by 'forking' on g8. ie. 1. b6 g3 2. b7 g2 3. b8(=Q) g1(=Q) 4. Q-h8+ K-g4 5. Q-g8+ etc.

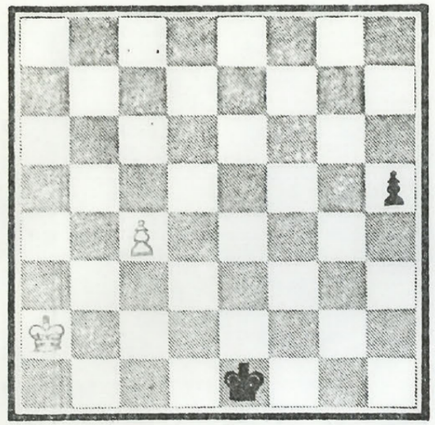


Fig. II-25. Draw

White to play queens first and forks on c1 if Black, in turn, queens. Instead Black draws by approaching his pawn. i.e. 1. c5 h4 2. c6 h3 3. c7 h2 4. c8(=Q) K-f1 and White's king is too far away. Normally a queen wins easily against a pawn on the seventh rank, but with rook or bishop pawns there is a stalemate defense. e.g. 5. Q-h3+ K-g1 6. Q-g3+ K-h1 7. K-b2 stalemate!

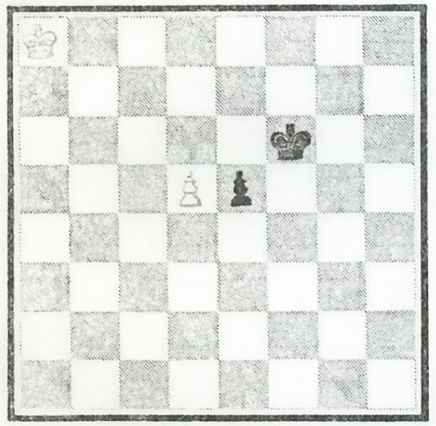


Fig. II-26 White wins

White supports his pawn and either queens with check or forks on e8 i.e. 1. K-b7 e4 2. d6 K-e6 (or 2. ... e3 3. d7 e2 4. d8(=Q)+) 3. K-c7 e3 4. d7 e2 5. d8(=Q)e1(=Q) 6. Q-e8+ etc. Note that the White king must be above the fifth rank and the White pawn above the fourth rank before subroutine PAWN2 will check out any winning possibilities in the position.

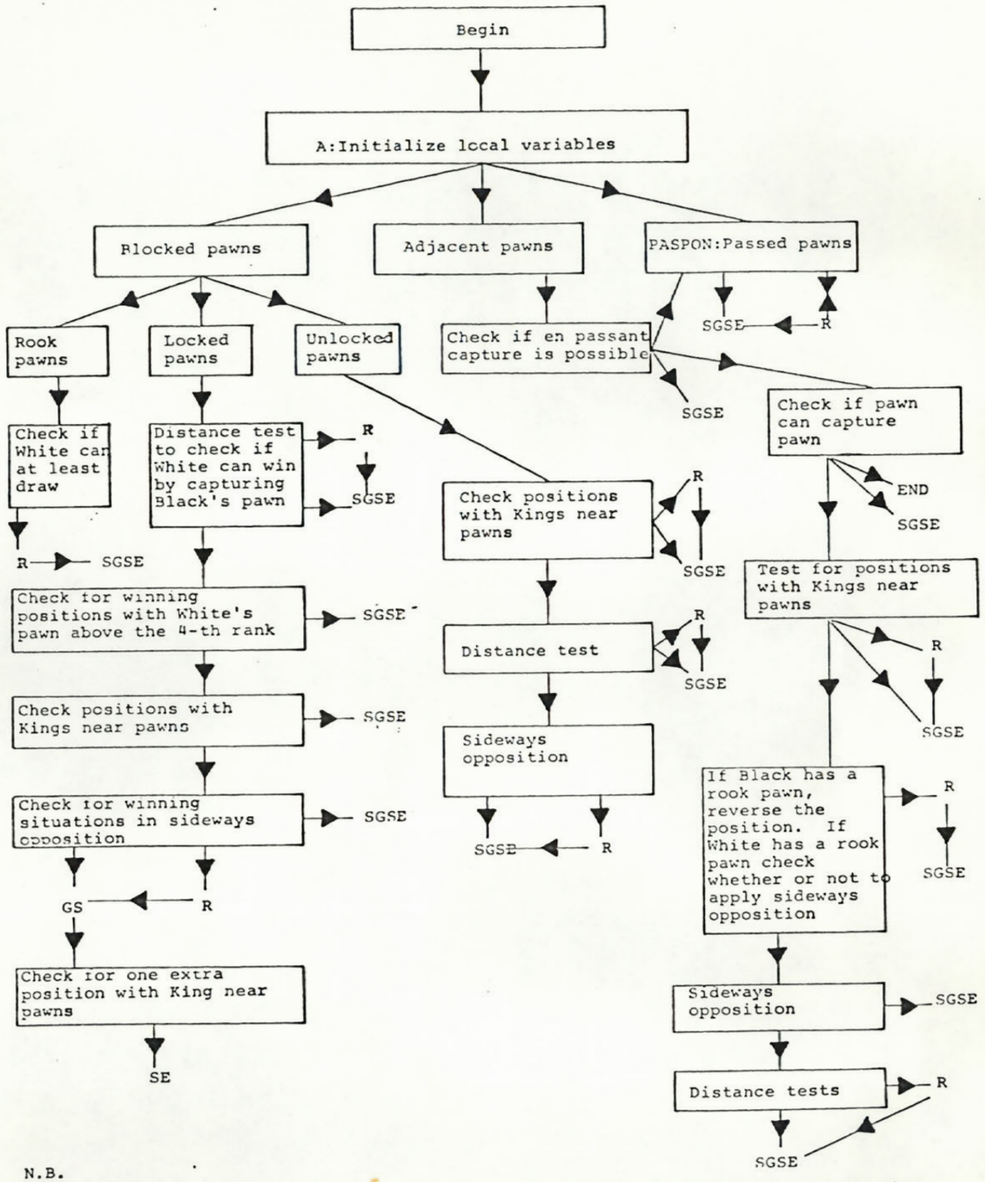
The most interesting problems occur with passed pawns. PAWN2 calls PASPON if the pawns are

(a) more than one file apart,

(b) blocked, but free to advance to the queening square without obstruction by the enemy pawn or

(c) adjacent, and similarly free to queen without threat of capture by the enemy pawn.

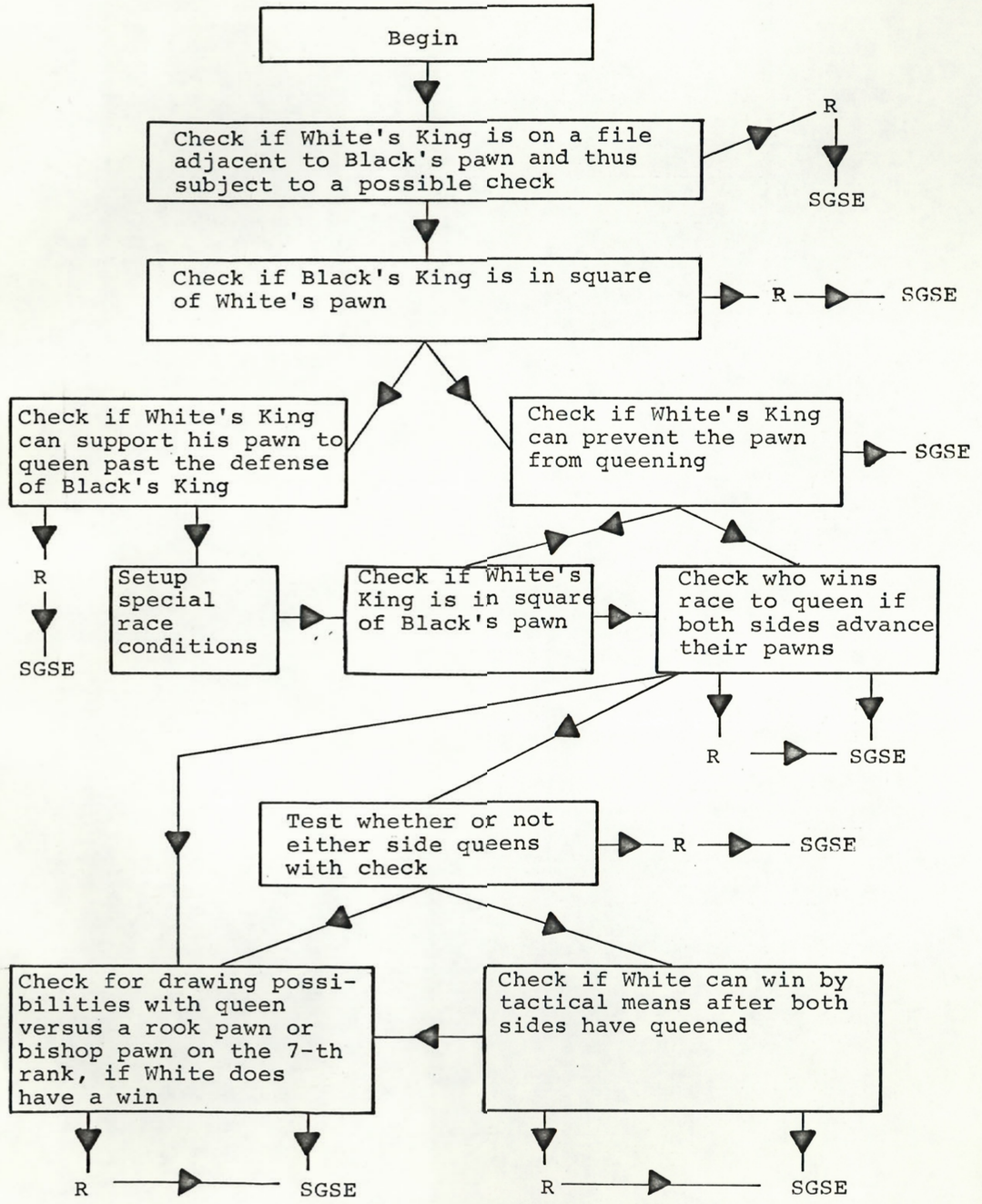
Essentially, we first determine whether Black's king can prevent White's pawn from queening and is thus 'within the square'. If Black cannot, we ascertain whether White's king is within the square and, if so, whether White can effectively foil any counterplay. See Figs. II-22,23. Since Black's king cannot catch the enemy pawn, White can afford to test the theory that queening his pawn wins. Thus we envisage a race to queen by both pawns and if White wins the race we begin a further sequence of tests to decide if the resulting position after White queens is indeed won for White (or at least drawn). This involves classifying the 'future type' positions where White has queened and trying to match it with known pattern situations where White 'forks' or mates, within at most a short series of moves. See Fig. II-24. We must then determine whether Black can postpone queening and retain a drawn endgame with his bishop or rook pawn on the seventh rank. See Fig. II-25. On the other hand, if Black is 'within the square', we must check whether White's king can support his pawn and force it through to queen against Black's best defense. If this is possible, we have another potential race to calculate. See Fig. II-26. Note that if (a) White is in check or (b) in front of his own pawn, he will have to make a



N.B.

R reverses the position chesswise using subroutine REVERS (corresponds to White→Black). Afterwards, control returns to A (except in PASPON) and eventually to the same program segment (eg. Blocked→R→A→Blocked→R→SGSE). If R is called once more, control proceeds directly to SGSE. S-Scores positions (see SE in the flow chart for TWOPAN). GS-Global Setup resets global variables (i.e. in Common) to their original values if the position was reversed (corresponds to Black→White). E-End.

Control of flow in subroutines PAWN2 and PASPON



Control of flow in subroutine PASPON

king move before queening his pawn, thus altering the 'future type' position. In the first case we always consider the node non-terminal and extend the search. In the second situation we also consider the node non-terminal, unless White's pawn wins the race with time to spare or we have already passed the maximum preset search depth.

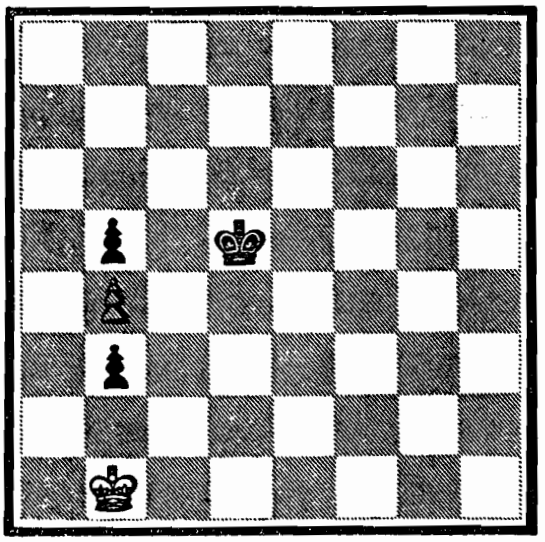
Before passing on to the results in chapter III we would like to point out that a few minor changes were made in the forward pruning section of PEASANT. Also another subroutine, called STLCHK, was introduced to check for stalemate at every node before calling ONEPAN, TWOPAN, and PAWN2.

EXPERIMENTAL RESULTS

3.0

Positions with just one pawn or two pawns of the same colour are essentially trivial.ONEPAN and TWOPAN were designed both for their intrinsic value and as a 'knowledge base' in more complicated endings. Instead we tested PEASANT on a variety of endgame problems from Pawn Endings[1] with King and Pawn vs. King and Pawn. The results are divided into three categories ie. blocked, adjacent and passed pawns. Of the 30 studies selected, 23 are solved; 7 of the 8 positions in the first category, 7 of 8 in the second and 9 of 14 in the third. The program required no more than 23 seconds of cpu time or a search depth of 9 plies to completely solve each study, and often less. (The program was run on the IBM 360 Model 75 computer which has a cpu cycle time of 195 nanoseconds). Also, the right move was chosen for the right reason! A winning move was selected only if all possible replies led to a forced loss for the enemy; a drawing move was winnowed out of a set of losing alternatives. Passed pawn endings claimed the most cpu time; this was probably because there were more moves to consider at each level of the tree ie. both pawns could move.

For each problem the data was collected and displayed in a table format. At every ply level, we record whether or not the move selected was correct, which move was chosen, the cpu time required, the score assigned by the program and the number of terminal nodes in the tree. Scores greater than a 1000 or less than -1000 are only given to clearly won or lost positions respectively; scores between -100 and 100 apply to drawn or undecided situations. If the program has chosen a move with a



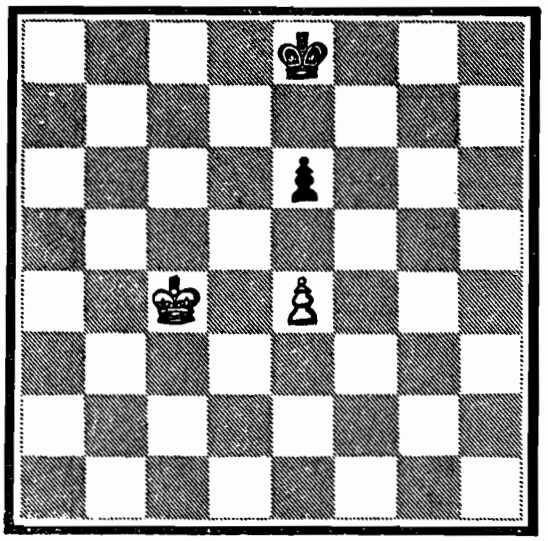
1. White to play and draw.

score between -100 and 100 we can only be sure that it has correctly evaluated the study (ie. it has chosen a move that really draws) by extending search in the principal variations to a terminal node, scored as a clear win, draw or loss. The principal variations refer to the likely sequence of moves, with minor alternatives, which we designate as best play in the given problem. The program determines its own 'principal continuation', which is often subject to error on the last ply of the variation. This is due to the inaccuracy of the scoring function used for undecided positions at maximum depth settings. We extended search till every move of the 'computer's side' (ie. the opponent attemting to draw or win) in the principal continuation was error free up till the last ply.

BLOCKED PAWN POSITIONS

3.1

1. We first present an ending with King and Pawn vs. King and two Pawns to show the transpositional character of such situations. This position occurred in the game Yates versus Tartakower, played in Bad Hamburg in 1927. After 1. K-b2 K-c4 2. K-a3 b2 White saved himself with 3. K-a2! (but note that 3. K:b2 K:b4 would have lost). PEASANT solved this problem quite easily but we had to continue searching 9 plies to get the program to realize that the position was completely drawn.

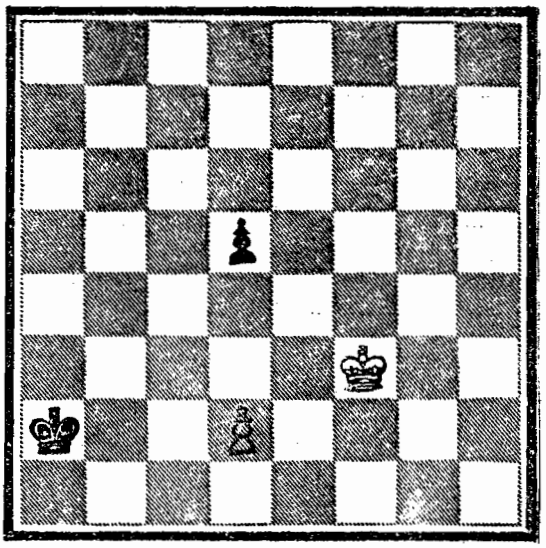


2. White to play and win.

NUMBER OF PLIES	1	2	3	4	5	6	7	8	9
VERIFICATION OF MOVE	✓	✓	×	✓	✓	✓	✓	✓	✓
MOVE CHOSEN BY PROGRAM	K-b2	K-b2	K-a1	K-b2	K-b2	K-b2	K-b2	K-b2	K-b2
CPU TIME	.02	.07	.12	.47	.70	1.55	2.08	2.92	4.02
SCORE OF POSITION	-12	-15	-15	-18	-20	-30	-18	-29	-8
# TERMINAL NODES	3	10	16	50	56	127	145	206	261

Here the result is correct at ply 9 because the principal continuation is 1. K-b2 K-c4 2. K-a3 b2 3. K-a2 b1(=Q)+ 4. K:b1 K:b4 with a clear draw according to ONEPAN. The program prefers this sequence to 1. K-b2 K-c4 2. K-a3 b2 3. K-a2 K-c3 4. K-b1 K:b4 (but not 4. ... K-b3 Stalemate!), which was selected as the principal continuation at ply 8 because TWOPAN cannot clearly evaluate this position. At ply 9 PEASANT 'sees' that 5. K:b2 draws and that the White king is even closer to Black's pawn on b2 than on b1.

2. This problem, composed by Chéron in 1926, exhibits many characteristics of blocked pawn situations. White only draws with king moves because Black can reply 1. ... e5 and gain the the opposition if ever White captures the pawn. So White must begin with 1. e5 and wins after 1. ... K-d7 2. K-b5! K-c7 3. K-c5 (see Fig. II-15) or 1. ... K-f7 2. K-c5 K-g6 3. K-c6! (but not 3. K-d6?? K-f5 and Black wins!) 3. ... K-g5 4. K-d7 K-f5 5. K-d6 (see Fig. II-16). The program actually solves the second variation after 3. ... K-g5 by pattern matching.



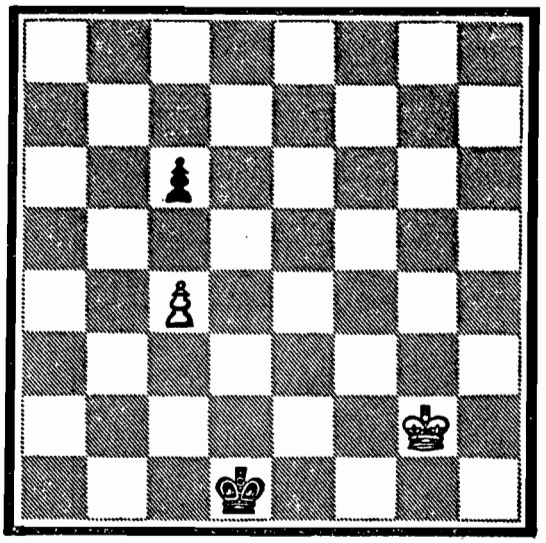
3. White to play and win.

NUMBER OF PLIES	1	2	3	4	5	6
VERIFICATION OF MOVE	×	×	×	×	×	✓
MOVE CHOSEN BY PROGRAM	K-c5	K-d4	K-c5	K-c5	K-c5	e5
CPU TIME	.03	.17	.33	.65	1.45	1.13
SCORE OF POSITION	-1	0	0	0	0	4395
NUMBER OF TERMINAL NODES	3	15	23	44	106	58

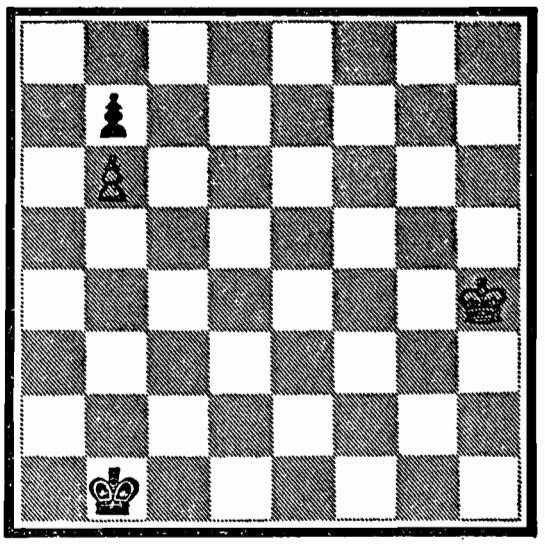
The principal continuation at ply 6 is 1. e5 K-e7 2. K-c5 K-d7 3. K-b6 (see Fig. II-14). The program has determined that this position is won and assigned it a score of 4395 because it has found that every possible reply to 1. e5 leads to a forced loss. See Appendix 2.

3. In this study, composed by Moravec in 1952, White must not advance his pawn because Black gets the better of the draw with 1. ... K-b3. Instead a king manoeuvre wins i.e. 1. K-f4! K-b3 2. K-e5 K-c4 3. d4.

NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	×	×	×	×	✓
MOVE CHOSEN BY PROGRAM	d3	d3	K-e3	K-e3	K-f4
CPU TIME	.05	.12	.92	1.63	3.95
SCORE OF POSITION	0	1	5	5	2406
NUMBER OF TERMINAL NODES	6	8	69	105	297



4. White to play and draw.



5. White to play and draw.

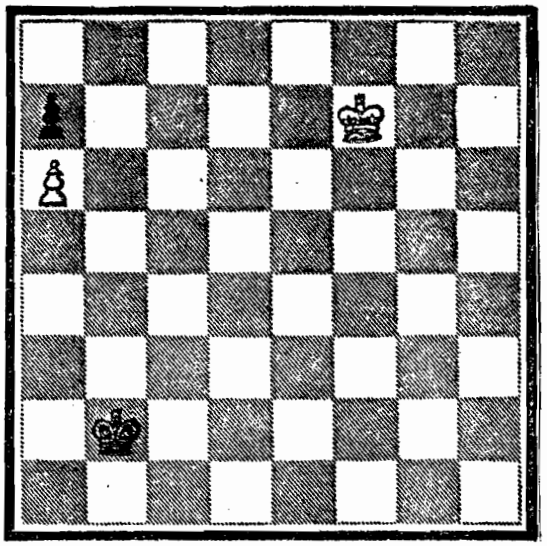
The principal continuation of ply 5 is 1. K-f4 K-b3 2. K-e5 K-b4 3. K:d5.

4. In this problem by Mandler (1949) White must be very careful in choosing his first move. If 1. c5? K-e2 Black wins (see Fig. II-17). Also 1. K-f3? K-d2 2. K-e4 K-c3 3. c5 K-c4 or 2. K-f2 c5 lose for White. The only way to draw is 1. K-f2! K-d2 2. c5! or 1. ... c5 2. K-e3! K-c2 3. K-e2! (not 3. K-e4? K-c3) 3. ... K-c3 4. K-d1 etc.

NUMBER OF PLIES	1	2	3	4	5	6
VERIFICATION OF MOVE	×	×	×	×	×	✓
MOVE CHOSEN BY PROGRAM	c5	K-f3	K-f3	K-f3	K-f3	K-f2
CPU TIME	.03	.15	.37	1.18	1.48	2.85
SCORE OF POSITION	-8	-10	-6	-4	-5	-4
NUMBER OF TERMINAL NODES	3	14	28	95	108	203

The principal continuation at ply 6 is 1. K-f2 K-c1 2. K-e3 K-b2 3. K-e4? K-b3? 4. c5? (3. K-d4!). The program considers 1. K-f2 Kd2 as a terminal node at all ply levels except ply 1 and assigns it a score of 0. Similarly 1. K-f2 c5 2. K-e3 K-c2 3. K-e2 is regarded as a terminal node with the score set equal to 0 (3. K-e4 K-c3 gets a score of -3503).

5. The solution to this study by Grigoriev (1931) is 1. K-g3! K-c2 2. K-f2! K-d3 3. K-e1! K-c4 4. K-d2 K-b5 5. K-c3 K:b6 6. K-b4 with a draw. White must avoid 1. K-g5? K-c2 2. K-f6 K-b3 3. K-e6 K-c4 4. K-d6 K-b5 5. K-c7 K-a6 etc. Note that 1. K-g3! K-c2 2. K-f3? K-d3! 3. K-f2 K-c4 also loses for White.

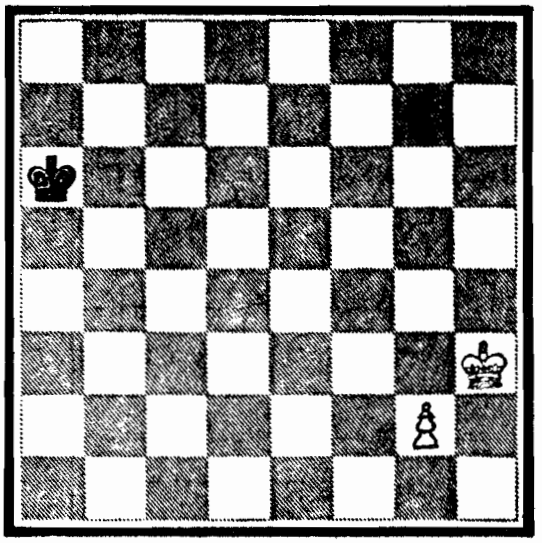


6. White to play and win.

NUMBER OF PLIES	1	2	3	4	5	6	7	8
VERIFICATION OF MOVE	✓	✓	✓	✓	✓	✓	✓	✓
MOVE CHOSEN BY PROGRAM	K-a3	K-a3	K-a3	K-a3	K-a3	K-a3	K-a3	K-g3
CPU TIME	.03	.08	.18	.30	.45	.67	.95	2.47
SCORE OF POSITION	-10	-9	-8	-7	-6	-5	-4	-10
NUMBER OF TERMINAL NODES	3	7	13	21	30	39	58	168

The principal continuation at ply 8 is 1. K-a3 K-c2 2. K-f2 K-d3 3. K-e1 K-e3?! 4. K-d1 K-f2?. If the program were to search 9 plies it would discover that 5. K-d2 wins for White! and thus would have to re-adjust the principal continuation. Nonetheless the program selected the first five plies of the principal variation correctly and a study of the tree verifies that the program 'saw' that the rest of this continuation, as shown earlier, also leads to a draw. Incidentally the principal continuation selected by the program at ply 7 is 1. K-a3 K-a2?! 2. K-f4 K-b3 3. K-e5? K-c4 4. K-d6, which of course, loses for White!

6. As mentioned in Chapter II, we have not developed winning functions for positions with two rook pawns on the same file. Thus to evaluate the following situation as a win the search must be extended till one pawn is captured. This position arose in the game Schlage vs. Ahues in 1921 and after 1. K-e6 K-c3 2. K-d6? K-d4 3. K-c6 K-e5 4. K-b7 K-d6 5. K:a7 K-c7 it ended in a prosaic draw. The correct solution is 1. K-e6 K-c3 2. K-d5!! K-b4 3. K-c6 K-a5 4. K-b7 and White wins.



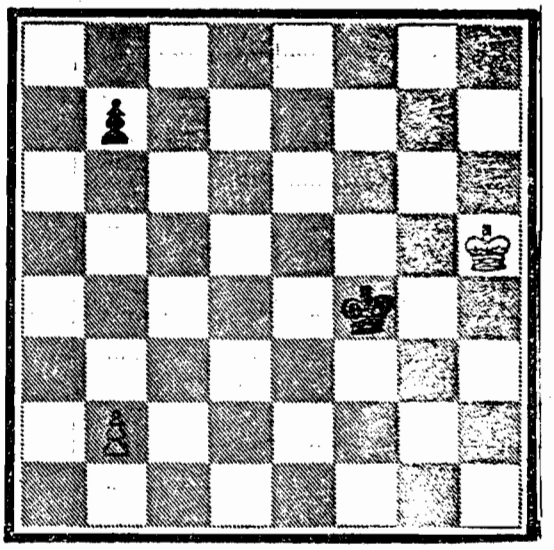
7. White to play and win.

NUMBER OF PLIES	1	2	3	4	5	6	7
VERIFICATION OF MOVE	✓	✓	✓	✓	✓	✓	✓
MOVE CHOSEN BY PROGRAM	K-e6	K-e6	K-e6	K-e6	K-e6	K-e6	K-e6
CPU TIME	.03	.05	.10	.23	.62	1.43	2.20
SCORE OF POSITION	-3	-2	-1	-1	0	-1	3207
NUMBER OF TERMINAL NODES	2	3	6	14	47	86	137

The principal continuation at ply 7 is 1. K-e6 K-b3 2. K-d7 K-b4 3. K-c6 K-a5 4. K-b7 K-b5; at ply 6 it is 1. K-e6 K-c3 2. K-d7? K-d4 3. K-c6 K-e5.

7. This difficult problem is solved correctly and efficiently by the program. The main variation is 1. K-g4! K-b5 2. K-f5 K-c4 3. g4 K-d4 4. g5 K-e3 5. g6! etc. or here 2. ... K-c5 3. K-e5! K-c6 4. K-c6! K-c5 5. g4 etc. Wrong would be 1. K-h4? K-b5 2. K-g5 K-c5! 3. K-g6 K-d5 4. K:g7 K-e5 5. K-g6 K-f4 6. K-h5 K-g3 with a draw or here 3. K-f5 K-d6! 4. K-g6 K-e5 etc. or 3. g4 K-d6 4. K-g6 K-e5 5. g5 K-f4 etc. Also 1. K-g3 K-b5 2. K-f4 K-c6! 3. K-f5 K-d6 etc. permits Black to draw.

NUMBER OF PLIES	1	2	3	4	5	6
VERIFICATION OF MOVE	×	×	×	×	✓	✓
MOVE CHOSEN BY PROGRAM	K-h4	K-h4	K-h4	g3	K-g4	K-g4
CPU TIME	.05	.12	.30	.75	2.90	4.98
SCORE OF POSITION	-4	-3	-2	-2	10	3593
NUMBER OF TERMINAL NODES	4	9	24	54	240	334

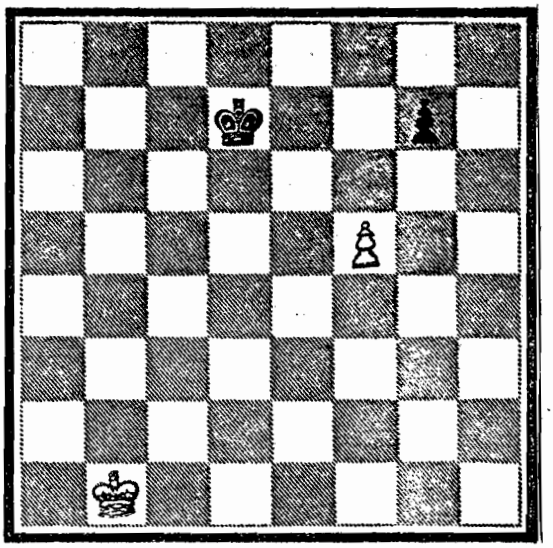


8. White plays, Black wins.

The principal continuation at ply 6 is 1. K-g4 K-b6 2. K-f5 K-c7 3. K-e6! g5. The program gives a worse score (i.e. 3994) to 1. K-g4 K-b5 2. K-f5 K-c5 3. K-e5! because it considers it a won terminal node earlier in the tree. Thus the program decides that Black would prefer to answer 1. K-g4 with 1. ... K-b6 instead of 1. ... K-b5, and yet, both moves lose. Note that the principal continuation at ply 5 is 1. K-g4 K-b6 2. K-f5 K-c7 3. K-g6 K-d6. The program does not realize that 3. K-e6 wins until Black has tried every possible reply at ply 6.

8. This study, composed by Grigoriev (1938), is too difficult for the program to solve in a reasonable amount of time. The principal variation is 1. K-g6 K-e5 2. K-f7 K-d6 3. K-e8 b5 4. K-d8 b4 5. K-c8 K-c6! 6. K-b8 b3 7. K-a7 K-b5! 8. K-b7 K-c4 9. K-b6 K-d3 10. K-b5 K-c2 etc. and Black wins; but not 7. ... K-c5? 8. K-a6 K-c4 9. K-a5 K-d3?? 10. K-b4 and Black even loses! The program would require approximately 15 plies and, conservatively estimated, about 1 hour of cpu time to find the win.

NUMBER OF PLIES	1	2	3	4	5	6
VERIFICATION OF MOVE	✓	✓	✓	✓	✓	✓
MOVE CHOSEN BY PROGRAM	K-g6	K-g6	K-g6	K-g6	K-g6	K-g6
CPU TIME	.05	.27	.65	2.60	3.38	10.08
SCORE OF POSITION	-14	-14	-13	-12	-11	-11
NUMBER OF TERMINAL NODES	5	27	49	220	212	788



9. White to play and draw.

The principal continuation at ply 6 is 1. K-g6 K-e5 2. K-f7 K-d6 3. K-e8 K-c5. That the program selects the first move correctly is irrelevant since we are trying to show that Black wins! and yet the principal continuation is quite reasonable.

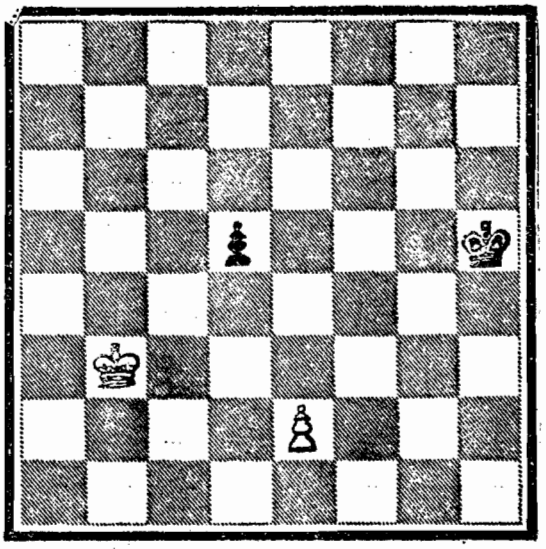
ADJACENT PAWN POSITIONS

3.2

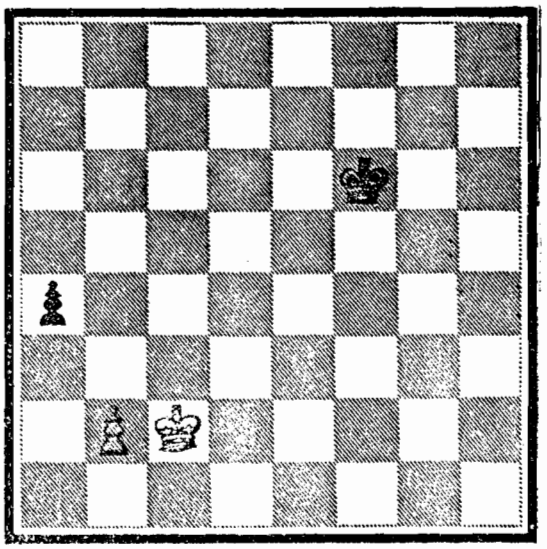
9. We begin this section with a rather simple ending by Moravec (1952). White draws by sacrificing his pawn and gaining the distant opposition i.e. 1. K-c2 K-d6 2. f6! (not 2. K-d3 K-e5) 2. ... g:f 3. K-d2! with a draw. The program has no trouble here.

NUMBER OF PLIES	1	2	3
VERIFICATION OF MOVE	✓	✓	✓
MOVE CHOSEN BY PROGRAM	K-c2	K-c2	K-c2
CPU TIME	.08	.17	.50
SCORE OF POSITION	-12	-13	-11
NUMBER OF TERMINAL NODES	12	21	61

The principal continuation at ply 3 is 1. K-c2 K-e7 2. K-d3 but the program does recognize that 1. Kc2 K-d6 2. f6 g:f is drawn and gives it a score of -6. Note that 1. K-c2 K-d6 2. f6 g6 gets a score of -1 because we use a very simple scoring function (involving PCHK, see 2.3) at maximum depth settings. Nevertheless 3. K-d2 would draw even without the White pawn on f6!



10. White to play and win.



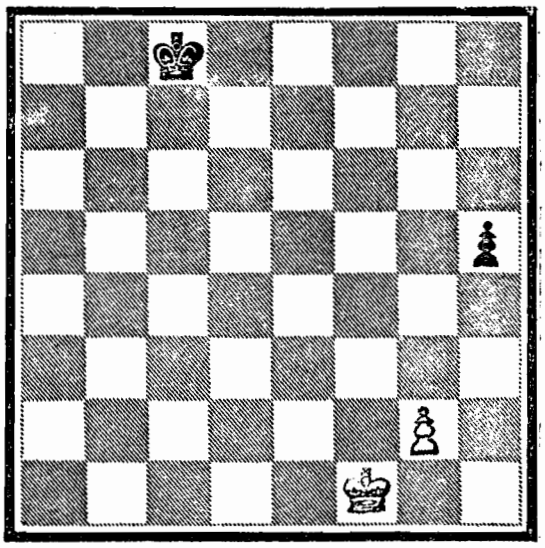
11. White to play and win.

10. The program has more difficulty with this problem by Moravec (1940). The solution is quite neat i.e. 1. K-b4!! K-g4 2. K-c5 K-f4 3. K-d4! K-f5 4. K:d5. Notice that 1. K-c3? K-g4 2. K-d4 K-f4 only draws.

NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	×	×	✓	✓	✓
MOVE CHOSEN BY PROGRAM	e3	e3	K-b4	K-b4	K-b4
CPU TIME	.05	.12	.70	1.53	4.10
SCORE OF POSITION	0	1	6	6	2006
NUMBER OF TERMINAL NODES	4	11	66	135	322

The principal continuation at ply 5 is 1. K-b4 K-g4 2. K-c5 K-f4 3. K-d4 K-g3.

11. This position is quite interesting. The obvious winning method would not work here i.e. 1. K-c3? a3! and if White captures, Black draws easily. Also both 2. b4 K-e5 3. K-b3 K-d5 4. K:a3 K-c6 5. K-a4 K-b6 and 2. b3 K-e5 3. K-c2 K-d4 4. K-b1 K-c3 5. K-a2 K-b4 only draw. The solution is 1. Kb1! a3! 2. b3! K-e5 3. K-a2 K-d5 4. K:a3 K-c6! 5. K-a4! (5. K-b4? K-b6 draws) and White wins.



12. White to play and win.

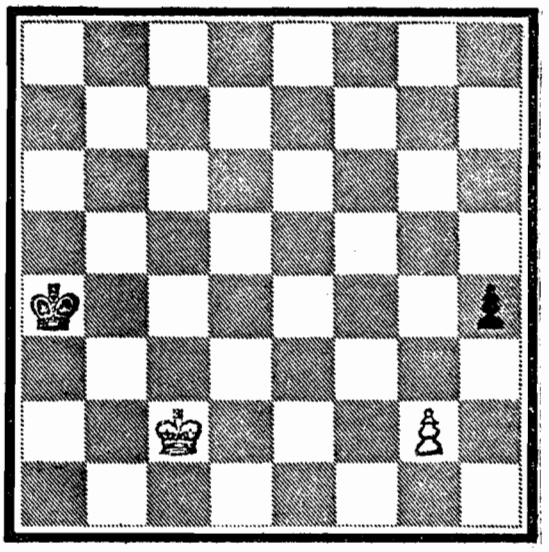
NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	×	×	×	×	✓
MOVE CHOSEN BY PROGRAM	b4	K-c3	K-c3	K-c3	K-b1
CPU TIME	.10	.28	.77	1.35	1.88
SCORE OF POSITION	-1	0	7	7	2005
NUMBER OF TERMINAL NODES	10	32	83	129	153

The principal continuation at ply 5 is 1. K-b1 K-e5 2. K-a2 K-d4 3. K-a3 K-d3.

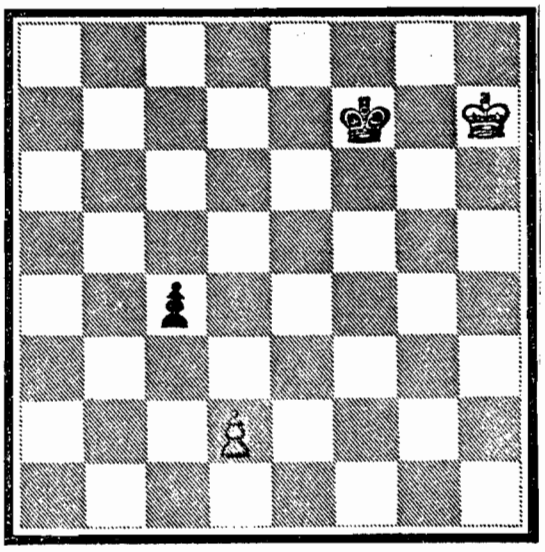
12. This problem, composed by Moravec in 1952 (a prodigious year!), is based on the previous study but adds one new wrinkle. White wins after 1. K-f2! h4 2. K-g1! h3 3. g3 K-d7 4. K-h2 K-e6 5. K:h3 etc. If 1. K-g1 or 1.g3 Black has enough time to defend the pawn.

NUMBER OF PLIES	1	2	3	4	5	6	7
VERIFICATION OF MOVE	×	×	×	×	✓	✓	✓
MOVE CHOSEN BY PROGRAM	g4	g3	g3	g3	K-f2	K-f2	K-f2
CPU TIME	.07	.10	.25	.37	1.58	2.58	6.60
SCORE OF POSITION	-3	-3	-2	-1	7	7	1606
NUMBER OF TERMINAL NODES	6	9	23	27	140	211	536

The principal continuation at ply 7 is 1. K-f2 h4 2. K-g1 K-d7 3. K-h2 K-e6 4. K-h3 K-f5.



13. White to play and win.



14. White to play and draw.

13. In this study by Moravec (1952!), White must again be aware of sacrificial offerings on the rook file. The solution is 1. K-d3 K-b5 2. K-e4 K-c6 3. K-f5! h3 (or 3. ... K-d6 4. K-g6 etc.) 4. g:h K-d7 5. K-f6 K-e8 6. K-g7 and White wins. Note that 3. K-f4? h3! 4. g:h K-d6 draws.

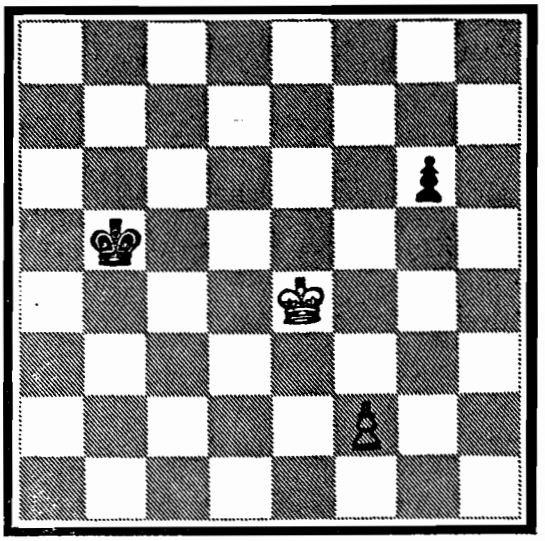
NUMBER OF PLIES	1	2	3	4	5	6	7
VERIFICATION OF MOVE	✓	✓	✓	✓	✓	✓	✓
MOVE CHOSEN BY PROGRAM	K-d3	K-d3	K-d3	K-d3	K-d3	K-d3	K-d3
CPU TIME	.08	.25	.52	1.45	2.47	4.53	6.40
SCORE OF POSITION	-5	-4	-3	-2	-1	0	1605
NUMBER OF TERMINAL NODES	11	32	58	156	252	426	517

The principal continuation at ply 7 is 1. K-d3 K-b5 2. K-e4 K-c4 3. K-f5 K-d3 4. K-g5 K-e2.

14. This position is remarkably easy to solve for the program. White draws after 1. K-h8! K-f6 2. K-g8! K-e5 3. K-f7 K-d4 4. K-e6 K-d3. 5. K-d5. Black wins after 1. K-h6? K-f6 2. K-h5 K-e5 3. K-g4 K-d4 4. K-f3 K-d3. The program sees that 1. K-h6? K-f6 loses because of sideways opposition and what other choice is there! If all the pieces were moved one file over (i.e. White's king from h7 to g7 etc.), the solution would be exactly the same. The program would reject any king move to the h-file because the distance formula would show that it loses (but forward pruning would reject it even sooner!).

NUMBER OF PLIES	1	2	3
VERIFICATION OF MOVE	x	✓	✓
MOVE CHOSEN BY PROGRAM	K-h6	K-h8	K-h8
CPU TIME	.08	.18	.35
SCORE OF POSITION	-15	-14	-13
NUMBER OF TERMINAL NODES	10	23	35

The principal continuation at ply 3 is 1. K-h8 K-f6 2. K-g8 . Notice that 2. K-h7 was rejected (with a score of -4509) because of the distance formula. We feel justified in cutting off the investigation because the program is quite obviously on the right track (both here and in problem #9). It is impossible to force the program to pick the main variation as the principal continuation unless search is extended to much greater depths. Since Black's king is closer to the pawns, PAWN2 will give Black a bonus when scoring at maximum depths and thus Black will procrastinate as long as possible (while avoiding repetitious drawing sequences) ie. 1. K-h8 K-f6 2. K-g8 K-e7 3. K-g7 K-e6 4. K-g6 K-e5 5.K-f7! K-d6 6. K-f6 K-d5 7. K-f5 K-d4 8. K-e6 K-e4 9. K-d6 K-d3 10.K-d5 etc.

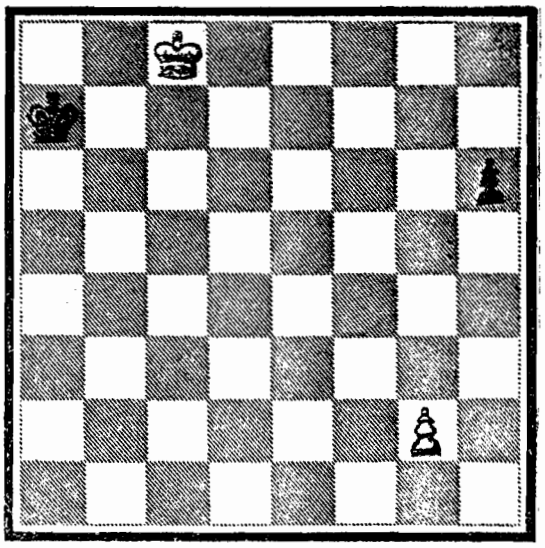


15. White to play and win.

15. This is the most difficult problem solved by the program in adjacent pawn endings. White wins after 1. K-d4! K-c6 2. K-e5 K-c5 3. f4! but both 1. f4? K-c4 2. K-e5 K-d3 and 1. K-d5? K-b4! 2. K-d4 K-b3! 3. f4 K-c2 only draw.

NUMBER OF PLIES	1	2	3	4	5	6
VERIFICATION OF MOVE	×	×	×	×	✓	✓
MOVE CHOSEN BY PROGRAM	f3	f3	K-e5	K-e5	K-d4	K-d4
CPU TIME	.08	.20	1.07	2.13	8.82	17.77
SCORE OF POSITION	-1	0	7	7	8	2808
NUMBER OF TERMINAL NODES	9	14	88	175	737	1370

The principal continuation at ply 6 is 1. K-d4 g5 2. f3 K-c6 3. K-e5 g4 4. f:g.

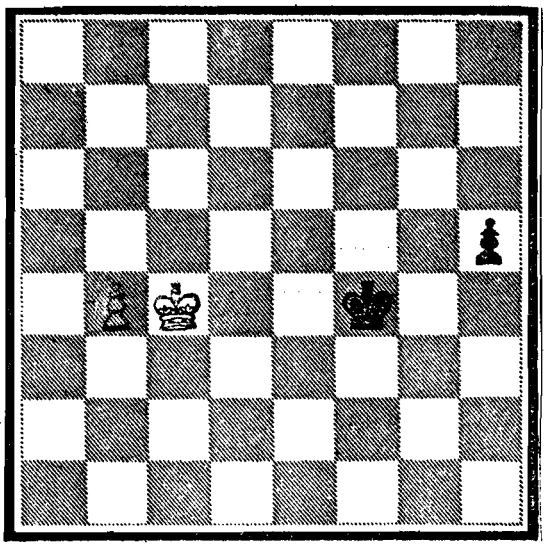


16. White to play and win.

16. The program has no chance with this amazing study, composed by Adamson in 1915. The solution is 1. K-c7! K-a6 2. K-c6! K-a5 3. K-c5 K-a4 4 K-c4 K-a3 5. K-c3 K-a2 6. K-c2 K-a3 7. g3!!! K-a2 8. g4!! K-a3 9. K-d3 K-b4 10. K-e4 K-c5 11. K-f5 and wins. Both 1. K-d7? K-b6 2. K-e6 K-c5 3. K-f5 K-d4 4. K-g6 K-e4 5. K:h6 K-f4 etc. and 1. g4? K-b6 only draw. In the main variation retreating the Black king at any point loses i.e. 1. ... K-a8 2. g4 K-a7 3. K-d7 etc.; similarly 5. ... K-a1 6. g4 etc. and 6. ... h5 7. K-d3 both lose for Black. White must play 7. g3!!! because 7. g4? K-b4 8. K-d3 K-c5 9. K-e4 K-d6 10. K-f5 h5! 11. g:h K-e7 allows Black to draw. The program would require about 25 plies and 1300 hours of cpu time!! to solve this correctly. We could cut out about 8 plies with a formula for exceptional rook pawn cases but it would probably still be too expensive! (≈ 80 minutes).

NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	×	✓	✓	×	×
MOVE CHOSEN BY PROGRAM	K-d8	K-c7	K-c7	K-d7	K-d7
CPU TIME	.05	.17	.72	1.75	5.28
SCORE OF POSITION	-10	-6	-5	-5	-4
NUMBER OF TERMINAL NODES	5	16	64	136	430

The principal continuation at ply 6 is 1. K-d7 K-h8 2. K-d6 h5 3. K-e7.



17. White to play and win.

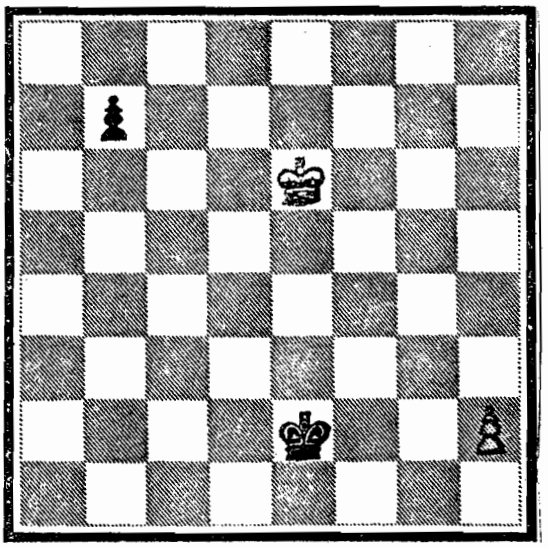
PASSED PAWN POSITIONS

3.3

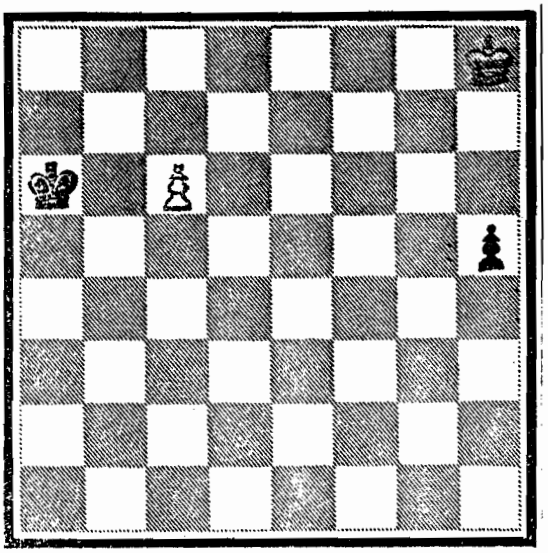
17. The winning motif in this problem by Moravec (1952!) is somewhat unusual i.e. 1. b5! K-e5 2. b6! K-d6 3. K-b5 h4 4. K-a6 h3 5. b7 K-c7 6. K-a7 and wins. White only draws both after 1. K-c5? when Black queens first and after 2. K-c5? K-e6! 3. b5 (or 3. K-c6 h4 and both sides queen) 3. ... K-d7 when Black can prevent the pawn's queening. In the main variation 3. ... K-d7 4. K-a6 K-c8 5. K-a7 is no better for Black.

NUMBER OF PLIES	1	2	3	4	5	6	7
VERIFICATION OF MOVE	✓	×	✓	✓	✓	✓	✓
MOVE CHOSEN BY PROGRAM	b5	K-b3	b5	b5	b5	b5	b5
CPU TIME	.07	.30	.52	.77	1.33	3.45	6.38
SCORE OF POSITION	-4	-6	-3	-4	-2	-2	3991
NUMBER OF TERMINAL NODES	8	35	53	78	128	281	446

The principal continuation at ply 7 is 1. b5 K-e5 2. b6 K-d6 3. K-b5 h4 4. K-a6.



18. White to play and win.



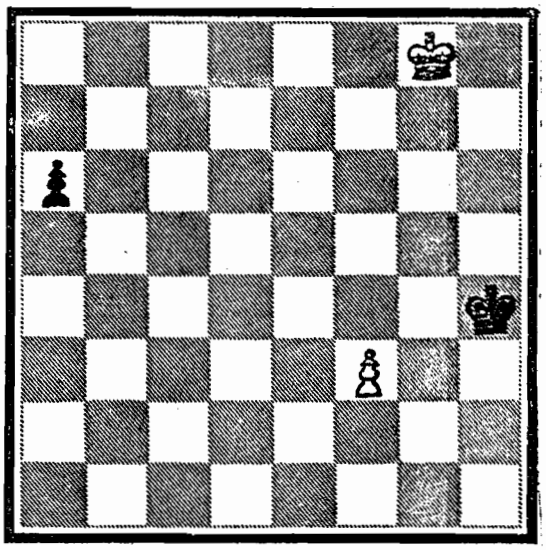
19. White to play and draw.

18. The idea in this study by Brenev (1931) is to force Black's king to occupy d3 whereupon White wins by forking i.e. 1. h4 b5 2. K-d5! K-d3 3. h5 b4 4. h6 b3 5. h7 b2 6. h8(=Q) b1(=Q) 7. Q-h7+. The program recognizes this on the fourth ply but must search deeper to be sure of a win if the pawn runs i.e. 1. h4 b5 2. K-d5 b4 3. K-c4 etc.

NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	×	✓	✓	✓	✓
MOVE CHOSEN BY PROGRAM	K-d6	h4	h4	h4	h4
CPU TIME	.10	.30	2.17	3.55	7.88
SCORE OF POSITION	0	-1	-1	2	3796
NUMBER OF TERMINAL NODES	10	28	203	296	530

The principal continuation at ply 5 is 1. h4 b5 2. K-d5 b4 3. K-c4.

19. This famous problem was composed by Reti in 1921 and, since then, its motif has been copied repeatedly. It would seem impossible for White to draw here but White's king moves serve a dual purpose. By both attacking the enemy pawn and threatening to support his own White draws i.e. 1. K-g7 h4 2. K-f6! h3 3. K-e6! h2 4. c7 etc. or 1. ... K-b6 2. K-f6! h4 3. K-e5! h3 4. K-d6 h2 5. c7 etc. An imaginative study!



20. White to play and draw.

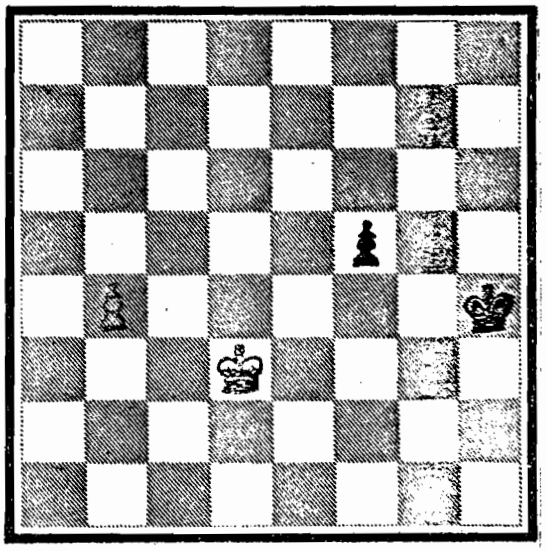
NUMBER OF PLIES	1	2	3	4	5	6	7	8
VERIFICATION OF MOVE	✓	✓	✓	✓	✓	✓	✓	✓
MOVE CHOSEN BY PROGRAM	K-g7	K-g7	K-g7	K-g7	K-g7	K-g7	K-g7	K-g7
CPU TIME	.03	.42	.55	1.48	1.77	4.50	6.50	11.90
SCORE OF POSITION	1	-3	-3	-3	-3	-8	-8	-10
NUMBER OF TERMINAL NODES	4	36	45	111	127	329	444	895

The principal continuation at ply 8 is 1. K-g7 h4 2. K-f6 h3 3. K-e7! K-b6 4. K-d6 h2. The program must search 8 plies deep to find the draw because of the horizon effect at ply 7 eg. 1. K-g7 h4 2. K-g6? h3 3. K-g5 h2 4. c7 and the winning move 4. ... K-b7 is only discovered at ply 8. The principal continuation at ply 7 is 1. K-g7 h4 2. K-g6? K-b6 3. K-g5 h3 4. K-g4 h2.

20. In this study by Prokes (1946), White also draws with a dual purpose king manoeuvre i.e. 1. K-f7! a5 2. f4 a4 3. f5 a3 4. f6 a2 5. K-g8! a1(=Q) 6. f7 and draws. White must avoid several pitfalls. Bad would be 1. f4? K-g4 2. K-f7 K-f5!! or in the main variation. 2. K-e6? a4 3. f4 a3 4. f5 a2 5. f6 a1(=Q) 6. f7 Q-a3 with Q-f8 winning for Black. Note that 1. K-f7 K-g3?! 2. K-e6 and White is in the square of Black's pawn; also 1. K-f7 a5 2. f4 K-g4?! 3. K-e6 a4 4. f5 draws easily. Finally we point out that 5. K-e7? in the principal variation would lose for White after 5. ...a1(=Q) 6. f7 Q-e5+ 7. K-d7 Q-f6 8. K-e8 Q-e6+ 9. K-f8 K-g5 10. K-g7 Q-e7 11. K-g8 K-g6 12. f8(=Q) Q-h7 mate!

NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	×	×	✓	✓	✓
MOVE CHOSEN BY PROGRAM	f4	f4	K-f7	K-f7	K-f7
CPU TIME	.03	.25	1.03	2.63	3.38
SCORE OF POSITION	-3	-11	-12	-10	-12
NUMBER OF TERMINAL NODES	4	23	92	237	260

The principal continuation at ply 5 is 1. K-f7 a5 2. f4 a4 3. f5. The program did not find a win in this variation because it realizes that Black's king is not close enough, if White plays 5. K-g8!, to allow White to queen and force the mate afterwards, as shown in an earlier sub-variation. (Thus Black cannot win after 5. K-g8 a1(=Q) 6. f7 Q-g1+ 7. K-h7 Q-f2 8. K-g7 Q-g3+ 9. K-h7 Q-f4 10. K-g7 Q-g5+ 11. K-h7 Q-f6 12. K-g8 Q-g6+ 13. K-h8! (not 13. K-f8? because the Black king can then approach and win) and 13. ... Q:f7 is stalemate!) A detailed discussion of queen vs. rook or bishop pawn on the seventh endings, including a description of the region in which the White king must be to secure the win, may be found in Chapter 9 of Pawn Endings [1]. These same results were (unfortunately) derived independantly by the author!

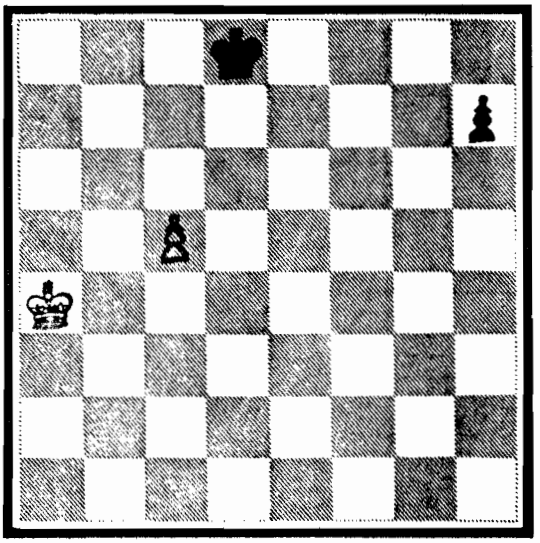


21. Black to play and draw.

21. This is a good practical example of finding the right first move in a tricky situation. In the game Najdorf vs. Vinuesa (Mar del Plata 1941) Black achieved a neat draw with 1. ... K-h3!! 2. b5 f4 3. K-e4 K-g3! 4. b6 f3 5. b7 f2 6. b8(=Q)+ K-g2. White would have won after 1. ... K-g3? 2. b5 f4 3. b6 f3 4. b7 f2 5. b8(=Q)+ K-g2 6. Q-g8+ K-h2 7. K-e2. Also 1. ... K-h3!! 2. b5 f4 3. K-e4 K-g4? would have lost for Black after 4. b6 f3 5. K-e3! K-g3 6. b7 f2 7. b8(=Q)+ etc.

NUMBER OF PLIES	1	2	3	4	5	6	7
VERIFICATION OF MOVE	×	×	×	×	×	×	√
MOVE CHOSEN BY PROGRAM	K-g5	K-g5	f4	f4	f4	f4	K-h3
CPU TIME	.05	.70	.67	2.90	4.80	10.60	22.30
SCORE OF POSITION	-2	-11	-3	-7	-11	-11	-11
NUMBER OF TERMINAL NODES	6	76	65	288	399	962	1787

The principal continuation at ply 7 is 1. ... K-h3 2. b5 f4 3. b6 f3 4. K-e3 K-g4? Both 4. ... K-g4? and 4. ... K-g2 have the same score at ply 7 but the former move is generated first! The program does not realize that 1. ... K-h3 2. b5 f4 3. K-e4 K-g4? 4. b6 f3 5. K-e3 K-g3 loses because this only comes up at ply 9; but it does see that 1. ... K-g3? loses and gives it a score of -4107 on the first ply! We must search 7 plies to refute the variation 1. ... f4 2. K-e2 (not 2. b5 f3 3. b6 K-h3!!) 2. ... K-g4 3. b5 K-f5 4. b6 K-e6 and only here does the program recognize that White will safely queen and Black will not be able to support his pawn to queen.

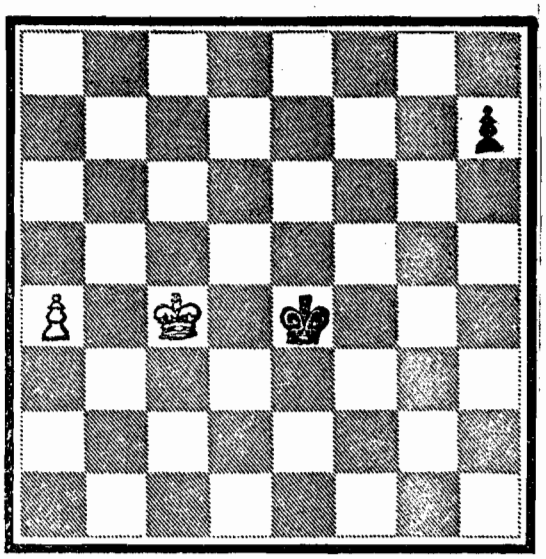


22. White to play and draw.

22. In another study by Moravec (1952!!) White draws by threatening both to re-enter the square of the enemy pawn and support his own i.e. 1. K-b5 h5 2. K-c6!! h4 3. K-b7 h3 4. c6 h2 5. c7+ etc. or 2. ... K-c8 3. K-d5.

NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	✓	✓	✓	✓	✓
MOVE CHOSEN BY PROGRAM	K-b5	K-b5	K-b5	K-b5	K-b5
CPU TIME	.12	.35	.70	1.70	3.28
SCORE OF POSITION	-14	-13	-12	-11	-11
NUMBER OF TERMINAL NODES	12	37	61	166	234

The principal continuation at ply 5 is 1. K-b5 K-c7 2. K-c4 K-b7?! 3. K-d5 h6?! (and yet Black still has a draw in hand i.e. 4. K-d6 K-c8!). The program assigns slightly better scores (because of PCHK) to both 1. K-b5 h5 2. K-c6 h4 3. K-b7 (-4) and 1. K-b5 h5 2. K-c6 K-c8 3. K-d5 (-2) and thus prefers the principal continuation.

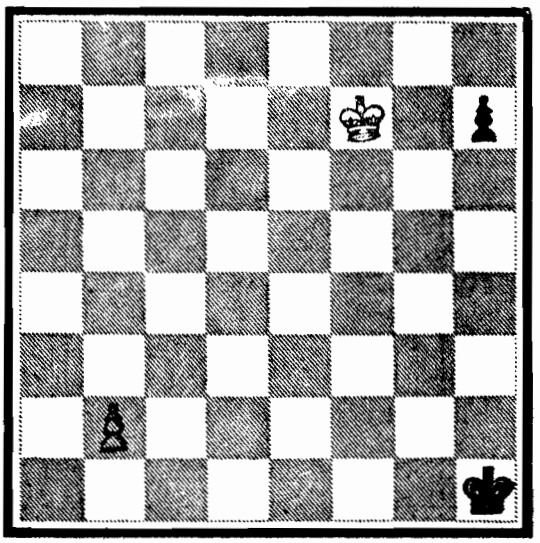


23. Black to play and draw.

23. Here Black draws by re-entering the square of White's pawn ie. 1. ... K-e5! 2. K-c5 K-e6 3. K-b6 K-d7 4. a5 K-c8 5. a6 K-b8. Black wins after both 3. K-c6? h5 4. a5 h4 5. a6 h3 6.a7 h2 7. a8(=Q) h1(=Q)+ forking the queen and, in the principal variation again, 4. K-b7? h5 5. a5 h4 6. a6 h3 7. a7 h2 8. a8(=Q) h1(=Q)+ 9. K-b8 Q-h2+ 10. K-a7 Q-f2+ 11. K-b8 Q-f4+ 12. K-a7 Q-d4+ 13. K-b8 Q-e5+ 14. K-a7 Q-a5+ 15. K-b7 (not 15. K-b8 Q-c7 mate!) 15. ... Q-b5+ 16. K-a7 K-c7 with mate to follow. The program actually knows that Black would win in this last variation on the fourth ply! because the pattern with K-d7 vs. K-b7 and two rook pawns is one of the future type positions discussed in 2.3.

NUMBER OF PLIES	1	2	3	4	5	6
VERIFICATION OF MOVE	×	×	✓	✓	✓	✓
MOVE CHOSEN BY PROGRAM	h6	h6	K-e5	K-e5	K-e5	K-e5
CPU TIME	.07	.13	.60	1.55	2.95	5.02
SCORE OF POSITION	-9	-15	-9	-14	-13	-14
NUMBER OF TERMINAL NODES	7	12	52	144	263	466

The principal continuation at ply 6 is 1. ... K-e5 2. K-b5 K-d6 3. K-b6 K-d7 4. K-b5? (4. a5!). Search would have to be extended several plies for the program to realize that 4. K-b5? just loses.

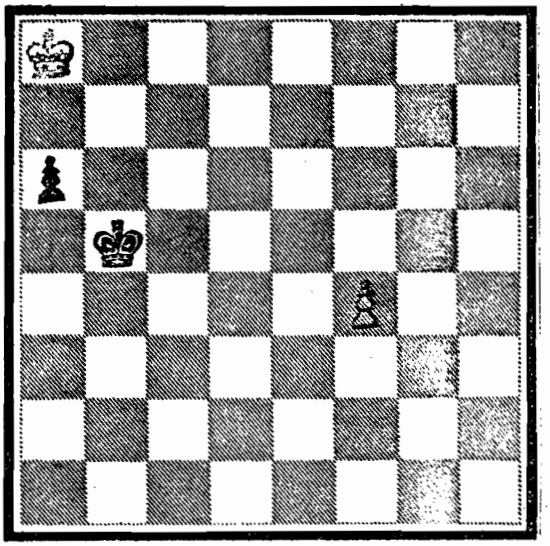


24. White to play and win.

24. In this position White forces the enemy king to a 'bad square' by attacking his pawn i.e. 1. K-f6! K-g2 2. h4 h5 3. K-g5 K-g3 4. b5 h4 5. b6 h3 6. b7 h2 7. b8(=Q)+ K-g2 8. K-g4! etc. Note that 1. b4? only draws with queen vs. rook pawn on the seventh and that 2. K-g5? allows Black to re-enter the square. Also 1. K-e6? h5 2. K-f5 h4 3. K-g4 K-g2 4. K:h4 K-f3 draws easily.

NUMBER OF PLIES	1	2	3	4	5	6
VERIFICATION OF MOVE	×	×	×	✓	✓	✓
MOVE CHOSEN BY PROGRAM	b4	b4	b4	K-f6	K-f6	K-f6
CPU TIME	.15	.18	1.43	3.08	10.27	12.67
SCORE OF POSITION	-3	-3	-3	-1	5	2407
NUMBER OF TERMINAL NODES	16	15	134	241	873	955

The principal continuation at ply 6 is 1. K-f6 h5 2. K-g5 h4 3. K:h4. The program knows that White's king is close enough to win the potential queen vs. rook pawn on the seventh ending, that occurs in the main variation, after 1. K-f6 K-g2 2. h4 h5 3. K-g5 K-g3 and assigns this position a score of 3994.

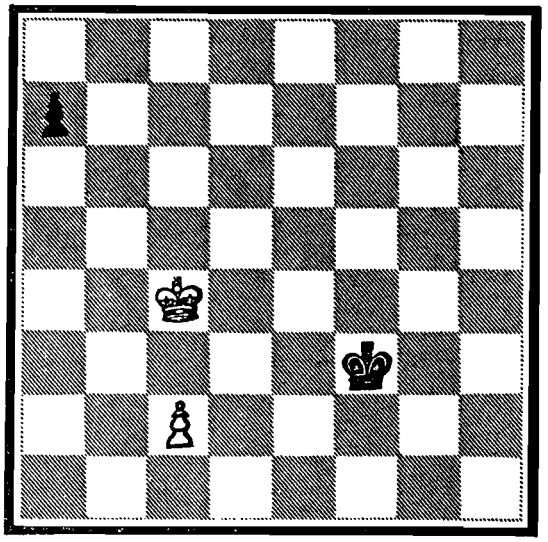


25. White to play and draw.

25. The program solves this study, composed by De Feijter in 1949, very efficiently i.e. 1. K-b7 a5 2. K-c7 K-c5 3. K-d7 K-d5 4. K-e7! K-e4 5. K-e6 K:f4 6. K-d5 with a draw. White loses after 2. f5? K-c5 3. K-c7 K-d5 4. K-d7 K-e5 etc. Also, in the main variation, 4. ... a4 5. f5 a3 6. f6 a2 7. f7 a1(=Q) 8. f8(=Q) draws.

NUMBER OF PLIES	1	2	3	4	5	6	7	8	9
VERIFICATION OF MOVE	✓	✓	✓	✓	✓	✓	✓	✓	✓
MOVE CHOSEN BY PROGRAM	K-b7	K-b7	K-b7	K-b7	K-b7	K-b7	K-b7	K-b7	K-b7
CPU TIME	.03	.17	.20	.30	.42	.85	1.08	1.82	2.10
SCORE OF POSITION	0	-1	-1	-12	-12	-13	-13	-11	-11
# TERMINAL NODES	3	18	20	29	39	76	90	155	166

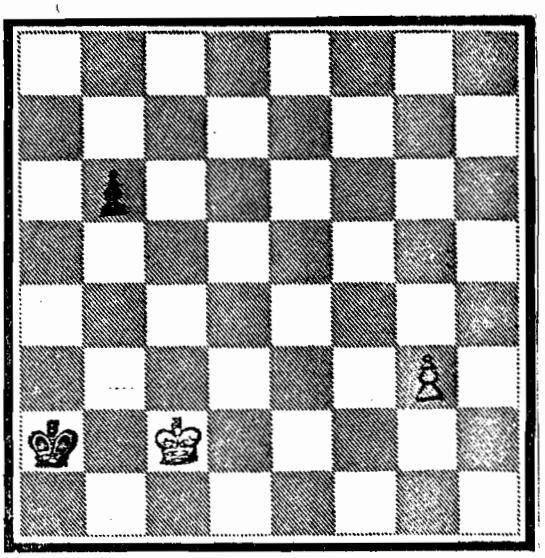
The principal continuation at ply 9 is 1. K-b7 a5 2. K-c7 K-c4 3. K-c6 a4 4. f4 a3 5. f6. The program preferred this to the main variation which gives White a better score (-6). See Appendix 2.



26. White to play and win.

26. This problem, composed by Grigoriev (1929), bears similarities to a few earlier studies (7,8,15,16). White's king must move off the pawn's file to allow its advance and simultaneously cut-off ('hold back') the enemy king and catch the enemy pawn. The solution is quite elegant i.e. 1. K-d4! K-f4 2. c4 K-f5 3. K-d5 K-f6 4. K-d6 K-f7 5. c5 K-e8 6. K-c7! a5 7. c6 a4 8. K-b8 a3 9. c7 a2 10. c8(=Q)+ and White wins. Obviously 1. K-d5? is bad because of 1. ...a5 2. c4 a4 etc. White also draws after 1. K-b4? K-e4! (but 1. ... K-e3? loses for Black as demonstrated in another study by Grigoriev (1931) in Pawn Endings) 2. c4 K-e5! 3. K-b5 K-d6 etc. or here 3. c5 K-e6! Black has no trouble after 1. K-b5? K-e4 2. c4 K-d4! 3. c5 K-d5 4. c6 K-d6 etc. or here 3. K-b4 a5+. The important tactical point which justifies White's play in the main variation occurs after 1. K-d4 K-f4 2. c4 K-f5 3. K-d5 a5 4. c5 K-f6 (4. ...a4 5. c6 and White queens first with check!) 5. K-d6 (not 5. c6? K-e7) 5. ...a4 (5. ... K-f7 6. c6 K-e8 7. c7 is worse) 6. c6 a3 7. c7 a2 8. c8(=Q) a1(=Q) 9. Q-h8+ forking!! A very nasty point... The program would require about 10 plies and about 10 minutes of cpu time to solve this.

NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	x	x	x	x	x
MOVE CHOSEN BY PROGRAM	K-b5	K-b5	K-b5	K-b5	K-b5
CPU TIME	.27	.77	1.93	5.78	20.52
SCORE OF POSITION	1	1	7	7	7
NUMBER OF TERMINAL NODES	28	72	165	488	1721



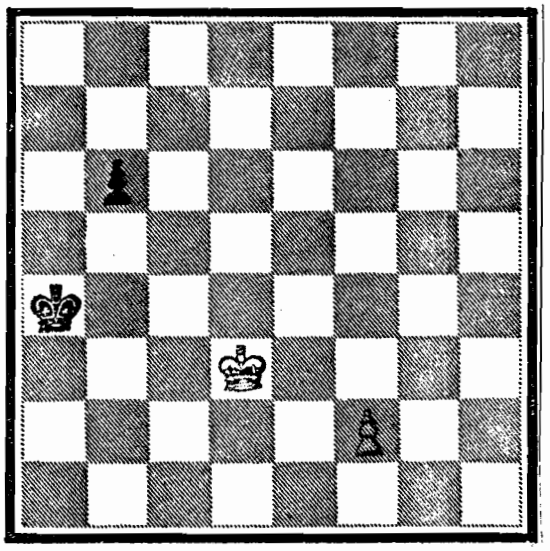
27. White to play and win.

The principal continuation at ply 5 is 1. K-b5 K-e3 2. K-a6 K-d4 3. K:a7. The program guarantees a slight edge by at least winning the enemy pawn.

27. In another study by Grigoriev (1928) White wins with an aesthetic king manoeuvre i.e. 1. K-c3! K-a3 2. K-c4! K-a4 3. g4 b5+ 4. K-d3! K-a3 5. g5 b4 6. g6 b3 7. g7 b2 8. K-c2! K-a2 9. g8(=Q)+ and mates next move. White would queen with check after 1. g4? but Black checks first with 1. ...b5 2. g5 b4 3. g6 b3+ and draws after 4. K-c3! b2 5. g7 b1(=Q) 6. g8(=Q)+ K-a1! Black must play 4. ... K-a3 in the principal variation because 4. ...b4?! 5. K-c2! K-a3 6. K-b1 is even worse. The program would require about 16 plies and 100 minutes of cpu time to solve this problem.

NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	√	×	×	×	×
MOVE CHOSEN BY PROGRAM	K-c3	g4	g4	g4	g4
CPU TIME	.10	.37	.92	1.62	3.02
SCORE OF POSITION	-4	-4	-3	-3	-3
NUMBER OF TERMINAL NODES	9	33	80	129	243

The principal continuation at ply 5 is 1. g4? b5 2. g5 b4 3. g6 K-a3?

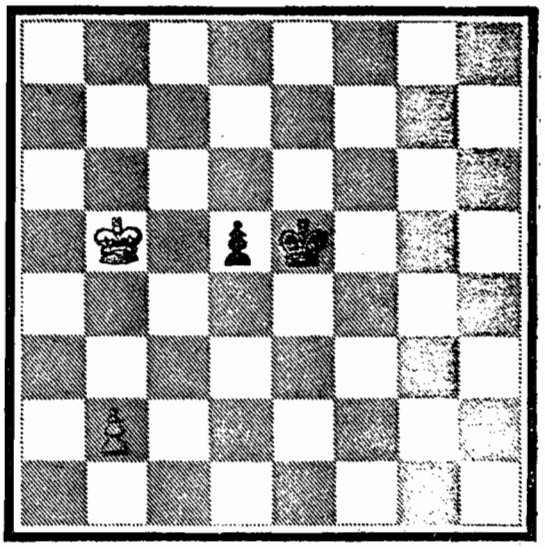


28. White to play and win.

28. In this problem (Grigoriev 1928!) White eventually forces Black's king to occupy a losing square. The solution is 1. K-d4!! (holding back!) 1. ...b5 2. f4 b4 3. f5 b3 4. K-c3! K-a3 5. f6 b2 6. f7 b1(=Q) 7. f8(=Q)+ K-a4 (or 7. ... K-a2 8. Q-a8 mate!) 8. Q-a8+ K-b5 9. Q-b8+ forking the queen. There is one major sub-variation i.e. 1. K-d4!! K-b5 2. K-d5! K-a6 3. f4 (but not 3. K-c6? b5 4. K-c5 K-a5 5. f4 b4 6. K-c4 b3! 7. K:b3 K-b5 with a draw) 3. ... K-b7 4. f5 K-c7 5. K-e6! K-d8 6. K-f7! b5 7. f6 b4 8. K-g8 and White queens first with check. Note that Black can delay getting into the main variation with 1. K-d4!! K-b5 2. K-d5! K-a4 3. f4 b5 4. f5 b4 5. K-c4! b3 6. K-c3! K-a3 etc. The program would require about 12 plies and about 13 minutes of cpu time to solve this study.

NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	x	x	x	x	x
MOVE CHOSEN BY PROGRAM	K-e3	f4	K-c4	f3	f4
CPU TIME	.18	.37	1.93	1.90	6.32
SCORE OF POSITION	-3	-2	-2	-2	-2
NUMBER OF TERMINAL NODES	19	35	182	150	575

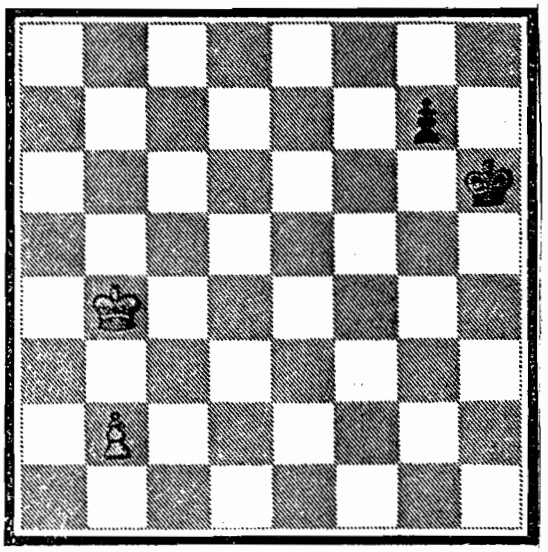
The principal continuation at ply 5 is 1. f4? b5? 2. f5 b4 3. K-c2 K-a5 (but 1. ... K-b5! draws).



29. White to play and draw.

29. This study by Moravec (1952!!!) exhibits a neat drawing manoeuvre i.e. 1. K-b4! K-d4 2. K-a5!! and, even though Black queens first and then checks on a1, White can still save his queen after K-b6. In order to refute White's other first move tries, the program must be partly redesigned. First we note that 1. Kb4! K-e4 2. K-c3 K-e3 3. K-c2 K-e2 4. K-c3 draws by repetition. Now both 1. K-b4! K-d4 2. K-a3? and 2. K-a4? lose after 2. ... K-e3 because Black forks on a1-b1 and queens with check, respectively. However 2. K-b3? is more difficult to refute. Black wins after 1. K-b4! K-d4 2. K-b3? K-d3! 3. K-a2! K-c2!! 4. b4 d4 5. b5 d3 6. b6 d2 7. b7 d1(=Q) 8. b8(=Q) Q-d5+ 9. K-a3 Q-a5 mate!!! An incredible variation... Notice here that 3. ...d4 4. b4 K-c4 5. K-b2 draws. The program would normally choose 1. K-b4 anyways at some ply depth but the same position occurs after a transposition of moves after 1. K-a4? K-e4! (not 1. ... K-d4? 2. b4! K-c4 3. K-a3 etc. or here 2. ... K-c3 3. b5 d4 4. b6 d3 5. b7 d2 6. b8(=Q) d1(=Q)+ 7. K-a5! Q-a1+ 8. K-b6 with a draw) 2. K-b3! K-d3! Thus it is a toss up between 1. K-a4? and 1. K-b4! and the former is generated first!?! Therefore the program cannot solve this problem correctly at any ply depth. PFASANT does not play queen endings!

NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	×	×	√	×	×
MOVE CHOSEN BY PROGRAM	K-c5	K-c5	K-b4	b3	b3
CPU TIME	.07	.35	1.23	2.25	7.25
SCORE OF POSITION	-4	-4	-4	-4	-4
NUMBER OF TERMINAL NODES	9	31	107	190	613



30. White to play and win.

The principal continuation at ply 5 is 1. b3? d4 2. K-c4 K-e4 3. b4. The program would realize this loses after 3. ...d3 4. K-c3 K-e3.

30. We end this section with a well-known study by Duras (1905) which is also too difficult for the program to solve in a reasonable amount of time. The solution is 1. K-c5!! K-g6 2. b4 K-f7 3. b5 K-e7 4. K-c6! K-d8 5. K-b7 g5 6. b6 g4 7. K-a8 and White queens first with check. Black's other major alternative also loses after 1. K-c5!! g5 2. b4 g4 3. K-d4! K-g5 4. b5 g3 (if 4. ... K-f4 White queens with check) 5. K-e3 K-g4 6. b6 K-h3 7. b7 g2 8. K-f2! K-h2 9. b8(=Q)+ etc.

The program would require about 16 plies and 300 minutes of cpu time.

NUMBER OF PLIES	1	2	3	4	5
VERIFICATION OF MOVE	√	×	×	×	×
MOVE CHOSEN BY PROGRAM	K-c5	K-c4	K-c4	K-c4	K-c4
CPU TIME	.05	.17	.67	1.78	6.33
SCORE OF POSITION	-10	-9	-8	-8	-7
NUMBER OF TERMINAL NODES	4	18	63	151	596

The principal continuation is 1. K-c4 K-g5 2. b4 K-f4? 3. K-d4! but the program doesn't realize yet that this position is winning for White (2. ... K-f6! draw).

SUMMARY

3.4

The basic approach used in solving these problems with the computer was to extend search until all the main tactical variations discussed in each position were correctly scored in the tree. However increasing the search depth would not have changed the principal variations once these had been correctly determined by the program. Note that when the program proposed an incorrect move that lost ie. instead of the drawing move (as in problem #4), the loss was over the horizon and so the search depth had to be increased. In some examples (see problem #5) the program found the right move for the wrong reason and the search depth also had to be increased until the problem was correctly solved.

FINAL OBSERVATIONS & CONCLUSIONS

4.0

We began work with the goal of improving PEASANT's tree-pruning heuristics and decided to attack the problem by first considering the simplest situation i.e. one pawn. As it turned out, we completely solved this case through exhaustive analysis. This approach, suggested by Newborn, was quite practical here because we wanted the program to play this ending very well but encountered great difficulty in programming such concepts as 'zugzwang' or 'opposition'. Next we moved on to positions with two pawns and designed many specific heuristics. It was impossible to apply the same procedure to positions with more than one pawn because the number of cases we would need to consider would be too great. We intended to build upon the foundation of our earlier efforts by using ONEPAN as a 'perfect information base'. This meant that,

(a) if a position with two pawns degenerated at some point in the tree into a one pawn case, we could count on ONEPAN to evaluate it perfectly, and

(b) if one of the pawns was 'ignored', we could check whether king and pawn were sufficient to win against a lone king by testing the appropriate position in ONEPAN.

In the simple case of two pawns of the same colour this approach was successful. However, we could not implement the latter portion of this procedure in positions with two pawns of opposite colour and indeed, as hoped for, in more complicated endings because of zugzwang. To try and decide whether a king could

support his pawn past the defense of the enemy king by examining ONEPAN would not work in positions where the win is attained through zugzwang (i.e. obligation to move) because the enemy can push his pawn instead of his king. One way to overcome this problem would be to categorize all situations in ONEPAN where zugzwang is necessary to win, but this would probably be too time consuming. In PAWN2 we developed a formula which needed constant revision due to the many exceptions found. Lack of time did not permit further research in complex pawn endings. Nevertheless we achieved quite reasonable results and gained an insight into the Herculean task facing chess programmers.

IMPROVING PAWN2's PERFORMANCE

4.1

"In spite of the limited material, pawn vs. pawn endings offer a wide variety of exciting ideas and special features. These ideas, which underlie very complex endings with more pawns, deserve deep study". Pawn Endings[1].

We chose a fair cross section of the problems available in Pawn Endings and the results of this experiment proved the program's efficacy. Yet there remains considerable room for improvement. Particularly noteworthy and pleasing was the solution of the moderately intricate studies nos. 7 and 15. Each exhibited the success of the sideways function in blocked and adjacent pawn positions respectively. This suggests that a similar function would be quite useful in passed pawn endings e.g. #26 clearly demonstrates the value of cutting off the enemy king from the pawns. The difficulty in evolving a function to

handle this complex a situation might be reduced through a moderate case by case approach. Furthermore it may be quite possible to develop new formulas for certain squares! For example, a White king on c6 is often very powerful in relation to a Black king on a6,a7,a8,b8 or c8.

The simplest way to improve the program would be to consider a position where each side has at least a draw as equal and assign it a score of zero. However this does entail double-checking of the formulas where PCHK is used. Other task areas would be to devise functions for the exceptional rook pawn situations and, in general, revise existing formulas.

We have not compared the program's performance before and after the inclusion of the three subroutines. It would be interesting to find out whether or not the increase in α-β cut-offs during the tree search more than compensates for the time lost by extensive evaluation at each node. A sensible test procedure would be to compare PEASANT's performance at different depth settings when,

(a) both ONEPAN and PAWN2 are implemented, and

(b) only ONEPAN is called at each node.

We believe that the benefits derived in applying complex evaluation functions far outweigh their inherent cost problems (at least potentially!).

Essentially the patterns, described in Tan's research (see 1.1), are also used in PAWN2. However, the underlying concept was considerably stretched in terms of the sideways function and 'future type' positions (see 2.3). Similar ideas will probably be created in more complicated endings.

GENERAL REMARKS ON PAWN ENDINGS

4.2

In pawn vs. pawn endings the author first attempted to develop a broad encompassing theory but settled on a limited yet practical approach. The results were nonetheless quite encouraging. However, we realize that extending our efforts to more difficult endings (where the complexity increases exponentially!) will not be simple, if at all possible, without evolving some basic general principles. Although it requires further development, the method of 'critical squares', discussed in Pawn Endings, deserves attention.

PEASANT was originally designed to continue the tree search past maximum preset ply depth only in reply to queening moves. There are several heuristics in PAWN2 which decide whether to prolong the search past maximum depth in order to reach fairly stable or 'quiescent' positions. We have experimented a bit with different depth settings (i.e. the number of extra plies permitted past max-depth) for one specific heuristic i.e. King attacking an undefended pawn. The results, though inconclusive, seem to indicate that increasing search, based on this heuristic, would be very useful in some positions (especially with passed pawns) and counter-productive in others! If this is indeed true, we could determine when it ought to be applied and why this is so. There must also be other heuristics worth developing. Eventually, in order to display a reasonably high level of play, endgame programs, even in pawn endings, must be designed to efficiently search depths of 20 to 30 plies as standard fare.

CONCLUSION

4.3

Information in chess literature and, as a consequence, expertise in chess programming is acquired in piecemeal fashion, through laborious investigation. We followed the example of books on chess endings and focused in great detail on each type of position, starting with the simplest. This approach would also be beneficial in major piece endings. e.g. Rook and Pawn vs. Rook [Some reports indicate that a team of computer chess specialists in Russia have solved this case completely, despite its extreme difficulty (masters misplay it!)]. It is basic chess knowledge that a position with bishop and rook pawn of the wrong colour generally does not win. It should be basic program knowledge as well.

Research in computer chess should probably concentrate on developing a hierarchy of intelligent, goal oriented decision rules within the tree search. Clearly we want to probe deeply in some positions and cut off the investigation in others. Combinations often occur where specific patterns are evident (e.g. a bishop sacrifice on h7 followed by a mating attack). Further psychological research is necessary to enable chess players to effectively communicate their thought processes. In this way we may learn how to find the right continuation by weeding out all the weaker alternatives. Even in a simple, pawn-ending tree-search much time is wasted on futile meandering variations... There is great progress to be made!

BIBLIOGRAPHY

1. Averbach, Y. and Maizelis, I.; Pawn Endings; B.T. Batsford, London 1974 (first published in the U.S.S.R.).

2. Fine, R.; Basic Chess Endings; David McKay, New York 1941.

3. Huberman, B.J.; A Program to Play Chess Endings; Stanford Artificial Intelligence Project, Memo AI-65, CS 106, Stanford University 1968.

4. Levenfish, G. and Smyslov, V.; Rook Endings; B.T. Batsford, London 1971 (first published in the U.S.S.R.).

5. Michalski, R.S. and Negri, P.; An Experiment on Inductive Learning in Chess Endgames - The King-Pawn-King Case, NATO Advanced Study Institute on Machine Representation of Knowledge, Santa Cruz 1975.

6. Newborn, M.M.; Computer Chess; Academic Press, New York 1975.

7. Newborn, M.M.; "PEASANT: An endgame program for kings and pawns", Chess Skill in Man and Machine, Frey (ed.); Springer Verlag New York 1977.

8. Tan, S.T.; Representation of Knowledge for Very Simple Pawn Endings in Chess; Memorandum MIP-R-98, University of Edinburgh 1972.

The reader who would like to learn more about recent research in computer endgame play may seek out the references listed below:

9. Beal, D.; King and Pawn vs. King - Discriminating Wins from Draws; Published privately in association with M.R.B. Clarke, Queen Mary College, London 1977.

10. Bramer, M.A.; Representations of Knowledge for Chess Endgames; Report from the Open University, Milton Keyes, U.K. 1975.

11. Bramer, M.A.; King and Pawn against King: Some Quantitative Data; Report from the Open University, Milton Keyes, U.K. 1977.

12. Clarke, M.R.B.; A quantitative study of king and pawn against king, Advances in Computer Chess, Clarke (ed.); Edinburgh University Press 1977.

13. Levy, D.N.L.; Chess and Computers; Batsford 1976.

14. Michie, D.; King and rook against king: Historical background and a problem on the infinite board, Advances in Computer Chess, Clarke (ed.); Edinburgh University Press 1977.

15. Tan, S.T.; A knowledge-based program to play chess endgames, Computer Chess, Bell (ed.); Conference Proc., Atlas Computer Laboratory, Chiltern, Berkshire 1973.

16. Tan, S.T.; Kings, Pawn and Bishop; Technical Report, Department of Machine Intelligence, Edinburgh, MIP-R-108 1974.

17. Tan, S.T.; Describing pawn structures, Advances in Computer Chess, Clarke (ed.); Edinburgh University Press 1977.

18. Zuidema, C.; Chess, How to Play the Exceptions ?; Report IW 21/76 Mathematisch Centrum, Amsterdam 1974.

APPENDIX 1

PEASANT is written in FORTRAN G for the IBM 360/370 series computers. It consists of about 3600 instructions and requires about 158K bytes of storage. During execution, the entire program resides in core memory. The following is a listing of the entire program with subroutines ONEPAN, TWOPAN and PAWN2 located towards the end. About 400 hours of programming work have been invested in the program developed by the author for this thesis.

```
      COMMON /ELIMD/ KT(20),SAVED(14,200),VECTOR(14,200),FIRSTO(20),SER
     1H,SERCHS
      COMMON /COLOR/ COLOUR
      COMMON/GAME/ WM(100),BM(100),STPOS(8,8),COUNT
       COMMON /DEEP/ DEPTH
      COMMON /DEBUG/ MSGLVL
       COMMON /RESULT/ PRINCA(14,14),IREFUT
       COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /IODATA/ MOVEST,MOVESC,NMOVES,NPOSIT
      COMMON /SPEC/ KTYPE(8,8),BOARDI(64)
      COMMON /OLDD/ LASTOX,LASTOY,LASTNX,LASTNY
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
       INTEGER DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
       INTEGER LASTOX,LASTOY,LASTNX,LASTNY
       INTEGER PLY,SCOLOR,C,MOVEP,D,MP,SCORE,BOARD,CAPTY,BOARDI
      INTEGER DEPTH,OX,OY,PRINCA,COLOUR
      INTEGER JMOVE(8),COUNT,STPOS,REPLAY,ITEST
      INTEGER KT,SAVED,VECTOR,FIRSTO,SERCH,SERCHS
      REAL*8 WM,BM,BLANK,PRGMOV,MANMOV
      LOGICAL*1 BL(8)
      EQUIVALENCE (BI,BLANK)
      DATA BL/' ',' ',' ',' ',' ',' ',' ',' '/
      MANMOV=0.0
1     CALL NEWONE
      CALL GETPOS
       MSGLVL=0
       DEPTH=1
      COUNT=0
      D=6
       CALL PRINTB
       IF (COLOUR.EQ.0) DEPTH=-DEPTH
      READ(9,400) ITEST
400   FORMAT(I1)
      IF (COLOUR.EQ.0) GOTO 90
      IF (ITEST.EQ.1) GOTO 202
200    C=1
      DO 10 I=1,14
      DO 10 J=1,14
      PRINCA(I,J)=0
10    CONTINUE
       IREFUT=0
       KONTA=0
       KONTS=0
      DRAWC1=0
      DRAWC2=0
      DRAWC3=0
      SCOLOR=1
      NPOSIT=0
      MOVEST=0
      MOVESC=0
      NMOVES=0
      DO 30 J=2,7
      DO 30 I=2,7
   30 KTYPE(I,J)=1
      KTYPE(1,8)=2
      DO 31 J=2,7
   31 KTYPE(J,8)=3
```

```
      KTYPE(8,8)=4
      DO 32 J=2,7
   32 KTYPE(8,J)=5
      KTYPE(8,1)=6
      DO 33 J=2,7
   33 KTYPE(J,1)=7
      KTYPE(1,1)=8
      DO 34 J=2,7
   34 KTYPE(1,J)=9
      LASTOX=0
      LASTOY=0
      LASTNX=0
      LASTNY=0
      SERCH=0
      SERCHS=0
      SCORE(1)=-10000
      DO 47 J=2,20
   47 SCORE(J)=(-1)*SCORE(J-1)
      CALL GENPL
      CALL  CLOCK1
      CALL TREE
      CALL PRINTR
      CALL CLOCK2(N)
      SECS=N/60.0
      WRITE(6,141)SECS
141   FORMAT(1H ,'JOB TIME=',F6.2,'SECONDS')
      IF (ITEST.NE.1) GOTO 49
      WRITE(6,193)
193   FORMAT(1H ,'REPLAY?')
      READ(9,400) REPLAY
158   FORMAT(I2)
      IF (REPLAY.NE.1) GOTO 49
202    WRITE(6,1193)
1193  FORMAT(1H ,'D=??')
       READ(9,158)REPLAY
      D=REPLAY
      WRITE(6,1194)
1194  FORMAT(1H ,'MSG=??')
      READ(9,158)REPLAY
      MSGLVL=REPLAY
      GOTO 200
49    MOV=IABS(PRINCA(1,1))
       DO 50 I=1,7
       K=MOD(MOV,10)
       MOV=(MOV-K)/10
       JMOVE(I)=K
50      CONTINUE
       JMOVE(8)=MOV
       CALL UPDA(JMOVE(3),JMOVE(2),JMOVE(5),JMOVE(4),JMOVE(8),JMOVE(1))
      CALL PMOVE(PRINCA(1,1),PRGMOV)
      COUNT=COUNT+1
      WM(COUNT+COLOUR)=MANMOV
      BM(COUNT+COLOUR)=PRGMOV
      WM(COUNT+1-COLOUR)=PRGMOV
90    BM(COUNT+1-COLOUR)=BLANK
      CALL USER(MANMOV,&1,&1,&90)
      BM(COUNT+1-COLOUR)=MANMOV
      IF (ITEST.EQ.1) GOTO 202
```

```
      GOTO 200
      END
      BLOCK DATA
      IMPLICIT INTEGER(A-Z)
      REAL*8 WM,BM
      COMMON /ELIMD/ KT(20),SAVED(14,200),VECTOR(14,200),FIRSTO(20),SER
     1H,SERCHS
      COMMON /GAME/ WM(100),BM(100),STPOS(8,8),COUNT
      COMMON /RESULT/ PRINCA(14,14),IREFUT
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /IODATA/ MOVEST,MOVESC,NMOVES,NPOSIT
      COMMON /SPEC/ KTYPE(8,8),BOARDI(64)
      COMMON /OLDD/ LASTOX,LASTOY,LASTNX,LASTNY
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /PASSEC/ PPSCOR,NUMBW,NUMBB,WHITEQ,BLACKQ,ADVW,ADVB,WPPP,B
     1PP
      COMMON /QCHECK/ MCHECK,ICHECK,JCHECK
      COMMON /GENSP/ JENTYP(20)
      COMMON /WMOVE/ UPPER(20),LOWER(20),NOVE(20),NP(20)
      COMMON /ADV/ WADV(20),BADV(20),WLFT(20),BLFT(20),WRT(20),BRT(20)
      DATA KT/20*0/,SAVED/2800*0/,VECTOR/2800*0/,FIRSTO/20*0/,
     1SERCH/0/,SERCHS/0/,WM/100*0.0/,BM/100*0.0/,COUNT/0/,
     1STPOS/64*0/,IREFUT/0/,MOVE/700*0/,SCOLOR/0/,MOVEP/0/
      DATA MP/20*1/,SCORE/20*0/,MOVEST/0/,MOVESC/0/,
     1CAPTY/20*0/,BOARD/64*0/,NMOVES/0/,NPOSIT/0/
      DATA KTYPE/64*0/,BOARDI/64*0/,LASTOX/0/,LASTOY/0/,
     1LASTNY/0/,LASTNX/0/,DRAWT/0/,DRAWC1/0/,DRAWC2/0/
      DATA DRAWC3/0/,KONTA/0/,KONTS/0/,WHITEX/11*0/,
     1WHITEY/11*0/,BLACKX/11*0/,BLACKY/11*0/,PPSCOR/0/,NUMBW/0/,
     1NUMBB/0/,WHITEQ/0/,BLACKQ/0/,ADVW/0/,ADVB/0/,WPPP/0/,BPPP/0/
      DATA MCHECK/0/,ICHECK/0/,JCHECK/0/,JENTYP/20*0/,UPPER/20*0/,
     1LOWER/20*0/,NOVE/20*0/,NP/20*0/,WRT/20*0/,BRT/20*0/
      DATA WADV/20*0/,BADV/20*0/,WLFT/20*0/,BLFT/20*0/
      END
      SUBROUTINE GENPI
      IMPLICIT INTEGER (A-Z)
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /SKORE/ MATERW,MATERB
      COMMON /PASSEC/ PPSCOR,NUMBW,NUMBB,WHITEQ,BLACKQ,ADVW,ADVB,WPPP,B
     1PP
      M=1
      N=1
      MATERW=0
      MATERB=0
      DO 10 I=1,11
      BLACKP(I)=0
      WHITEP(I)=0
10     CONTINUE
      DO 20 J=1,8
      DO 20 I=1,8
      IF(BOARD(I,J).NE.6)   GO TO 16
      WHITEP(1)=6
      WHITEX(1)=I
```

```
      WHITEY(1)=J
      GO TO 20
   16 IF(BOARD(I,J).NE.-6)  GO TO 17
      BLACKP(1)=-6
      BLACKX(1)=I
      BLACKY(1)=J
      GO TO 20
   17 IF(BOARD(I,J).NE.-1)  GO TO 20
      IF (J.EQ.2) GOTO 100
      M=M+1
      MATERB=MATERB+BOARD(I,J)
      BLACKP(M)=BOARD(I,J)
      BLACKX(M)=I
      BLACKY(M)=J
   20 CONTINUE
      DO 32 I=1,8
      DO 32 J=1,8
      IF(BOARD(9-J,9-I).NE.1)  GO TO 32
      IF (9-I.EQ.7) GOTO 100
      N=N+1
      MATERW=MATERW+1
      WHITEP(N)=1
      WHITEX(N)=9-J
      WHITEY(N)=9-I
   32 CONTINUE
      NUMBB=M
      NUMBW=N
      WHITEP(N+1)=0
      BLACKP(M+1)=0
      RETURN
  100  WRITE(6,101)
  101 FORMAT(1H ,'  PLEASE USE OTHER PROGRAM  ')
      STOP20
      END
      SUBROUTINE  TREE
      IMPLICIT INTEGER (A-Z)
      COMMON /DEBUG/ MSGLVL
      COMMON /RESULT/ PRINCA(14,14),IREFUT
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /IODATA/ MOVEST,MOVESC,NMOVES,NPOSIT
      COMMON /QCHECK/ MCHECK,ICHECK,JCHECK
      COMMON /SKORE/ MATERW,MATERB
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /GENSP/ JENTYP(20)
      COMMON /PASSEC/ PPSCOR,NUMBW,NUMBB,WHITEQ,BLACKQ,ADVW,ADVB,WPPP,B
     1PP
      COMMON /ELIMD/ KT(20),SAVED(14,200),VECTOR(14,200),FIRSTO(20),SER
     1H,SERCHS
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /WMOVE/ UPPER(20),LOWER(20),NOVE(20),NP(20)
      PLY=0
      IF(MSGLVL.LE.2) GOTO 301
      WRITE(6,302)
  302 FORMAT(' ','ENTER RESTRICTIONS',/)
      READ(9,303) (UPPER(I),I=1,8)
      READ(9,303) (LOWER(I),I=1,8)
```

```
303   FORMAT(8I3)
301   CONTINUE
  100 IF(PLY.EQ.0)  GO TO 150
      MOVE(700)=0
      CALL STLCHK
      IF (DRAWC3.EQ.1) GOTO 444
      CALL ONEPAN
      CALL TWOPAN
      CALL PAWN2
444   IF(DRAWC3.EQ.1) SERCH=SERCH+1
      IF(DRAWC3.EQ.1) GOTO 1003
      IF(MOD(IABS(MOVE(MP(PLY))),10).EQ.3)  GO TO 150
      IF((MATERW.LE.8).AND.(MATERB.GE.-8))  GO TO 110
      SCORE(PLY+3)=0
      IF(MATERB.LT.-8)  SCORE(PLY+3)=-6000+200*PLY
      IF(MATERW.GT.8)  SCORE(PLY+3)=SCORE(PLY+3)+6000-200*PLY
      GO TO 1003
  110 CALL PASSED
      IF(PPSCOR.NE.0)  GO TO 1002
      IF((SCORE(PLY+2)*(-C)).GE.(6000-(PLY+3)*200)) GO TO 1005
      IF((WHITEQ.EQ.BLACKQ).AND.(WHITEQ.NE.10))  GO TO 1006
      IF((PLY.GE.D).AND.(MOVE(700).EQ.0))  GO TO 1000
      CALL DRAWN
      IF(DRAWT.NE.0)  GO TO 1003
      SERCHS=SERCHS+1
      IF (PLY.LE.2) GO TO 150
      IF(KT(PLY-2)+KT(PLY).NE.2)  GO TO 150
      KSQ=MOD(IABS(MOVE(MP(PLY)))/1000,100)
      OPPMV=MOD(IABS(MOVE(MP(PLY-1))),100000)
      CODE=100*OPPMV+KSQ
      K=PLY-2
      J=FIRSTO(K)-1
   50 IF(J.EQ.0) GO TO 150
      IF(CODE.EQ.VECTOR(K,J)) GO TO 1007
      J=J-1
      GO TO 50
 1007 SCORE(PLY+3)=SAVED(K,J)
      GO TO 1003
150   IF ((MATERW+IABS(MATERB).GT.2).OR.(PLY.EQ.0)) CALL GENERT
      NMOVES=NMOVES+1
      FIRSTO(PLY+1)=1
      IF(MOVEP.EQ.50*PLY)  GO TO 999
      SCORE(PLY+3)=SCORE(PLY+1)
      MP(PLY+1)=50*PLY+1
      IF ((PLY.EQ.0).AND.(MATERW+IABS(MATERB).LE.2)) CALL PASSED
      CALL PLAUS(MP(PLY+1))
      IF(MOVE(MP(PLY+1)).NE.0)  GO TO 200
      SCORE(PLY+3)=-C*8000
      GO TO 1003
  200 PLY=PLY+1
      CALL UPDATE
      GO TO 100
  999 SCORE(PLY+3)=(-C)*MCHECK*(6000-100*PLY)
      GOTO 1003
 1000 SCORE(PLY+3)=(MATERW+MATERB)*10+(WPPP-BPPP)*5-ADVW+ADVB
     1              +10*(BLACKQ-WHITEQ)
1008  CALL KSCORE
      IF(WHITEX(1).NE.BLACKX(1)) GO TO 1003
```

```
      IF(WHITEY(1).EQ.BLACKY(1)-2) SCORE(PLY+3)=SCORE(PLY+3)-(3*C)
      GO TO 1003
 1005 SCORE(PLY+3)=8000*C
      GO TO 1003
 1006 SCORE(PLY+3)=0
      GO TO 1008
 1002 SCORE(PLY+3)=PPSCOR
 1003 NPOSIT=NPOSIT+1
      SCORED=1
      IF (PLY.GE.MSGLVL) GOTO 1001
      DO 131 I=1,MSGLVL
      IF((UPPER(I).NE.0).AND.(MP(I).GT.UPPER(I))) GOTO 1001
      IF((LOWER(I).NE.0).AND.(MP(I).LT.LOWER(I))) GOTO 1001
131   CONTINUE
      DO 113 I=1,PLY
      POVE=IABS(MOVE(MP(I)))
      NOVE(I)=MOD(POVE,100000)
      POVE=POVE/10000000
      IF(MOVE(MP(I)).GT.0) NOVE(I)=NOVE(I)+POVE*100000
      IF(MOVE(MP(I)).LT.0) NOVE(I)=-NOVE(I)-POVE*100000
113   NP(I)=MOD(MP(I),50)
      WRITE(6,2) SCORE(PLY+3), (NP(I),NOVE(I),I=1,PLY)
    2 FORMAT(' ',I5,10(I3,I7))
 1001 IF(SCOLOR*C.EQ.1)   GO TO 1010
      IF(SCORE(PLY+3).GT.SCORE(PLY+2))  GO TO 1020
      GO TO 1030
 1010 IF(SCORE(PLY+3).LT.SCORE(PLY+2))  GO TO 1020
      GO TO 1030
 1020 SCORE(PLY+2)=SCORE(PLY+3)
      PRINCA(PLY,PLY)=MOVE(MP(PLY))
      IF(SCORED.NE.1) GOTO 1013
      PRINCA(PLY,PLY+1)=-1
      GOTO 1030
1013  J=1
1015  PRINCA(PLY,PLY+J)=PRINCA(PLY+1,PLY+J)
      IF(PRINCA(PLY,PLY+J).EQ.-1) GOTO 1030
      IF(PLY+J.GT.D) GOTO 1030
      J=J+1
      GOTO 1015
 1030 PLY=PLY-1
      SCORED=0
      CALL RESTOR
      IF(PLY.NE.MSGLVL-1) GOTO 1037
      DO 1331 I=1,MSGLVL
      IF((UPPER(I).NE.0).AND.(MP(I).GT.UPPER(I))) GOTO 1037
      IF((LOWER(I).NE.0).AND.(MP(I).LT.LOWER(I))) GOTO 1037
1331  CONTINUE
      DO 1113 I=1,MSGLVL
      POVE=IABS(MOVE(MP(I)))
      NOVE(I)=MOD(POVE,100000)
      POVE=POVE/10000000
      IF(MOVE(MP(I)).GT.0) NOVE(I)=NOVE(I)+POVE*100000
      IF(MOVE(MP(I)).LT.0) NOVE(I)=-NOVE(I)-POVE*100000
1113  NP(I)=MOD(MP(I),50)
      WRITE(6,2) SCORE(MSGLVL+2),(NP(I),NOVE(I),I=1,MSGLVL)
 1037 IF(MOVE(MP(PLY+1)+1).NE.0)   GO TO 1100
 1031 IF(PLY.EQ.0)    GO TO 4000
      GO TO 1001
```

```
 1100 IF(SCOLOR*C.NE.1)  GO TC 1110
      IF(SCORE(PLY+3).GE.SCORE(PLY+2))  GO TO 1130
      GO TO 1150
 1110 IF(SCORE(PLY+3).LE.SCORE(PLY+2))  GO TO 1130
      GO TO 1150
 1130 PLY=PLY-1
      CALL RESTOR
      IF(PLY.NE.MSGLVL-1) GOTC 1041
      DO 13331 I=1,MSGLVL
      IF((UPPER(I).NE.0).AND.(MP(I).GT.UPPER(I))) GOTO 1041
      IF((LOWER(I).NE.0).AND.(MP(I).LT.LOWER(I))) GOTO 1041
13331 CONTINUE
      DO 11113 I=1,MSGLVL
      POVE=IABS(MOVE(MP(I)))
      NOVE(I)=MOD(POVE,100000)
      POVE=POVE/10C00000
      IF(MOVE(MP(I)).GT.0) NOVE(I)=NOVE(I)+POVE*100000
      IF(MOVE(MP(I)).LT.0) NOVE(I)=-NOVE(I)-POVE*100000
11113 NP(I)=MOD(MP(I),50)
      WRITE(6,2) SCORE(MSGLVL+2),(NP(I),NOVE(I),I=1,MSGLVL)
 1041 IF(MOVE(MP(PLY+1)+1).EQ.0)  GO TO 1031
 1150 MP(PLY+1)=MP(PLY+1)+1
      GO TO 200
 4000 RETURN
      END
      SUBROUTINE KSCORE
      IMPLICIT INTEGER (A-Z)
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      CCMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      XT=0
      YT=0
      PC=0
      DO 10 I=2,10
      IF(WHITEP(I).EQ.0) GOTO 20
      IF(WHITEP(I).NE.1) GOTO 10
      PC=PC+1
      XT=XT+WHITEX(I)
      YT=YT+WHITEY(I)
10    CONTINUE
20    DO 30 I=2,10
      IF(BLACKP(I).EQ.0) GOTO 40
      IF(BLACKP(I).NE.-1) GOTO 30
      PC=PC+1
      XT=XT+BLACKX(I)
      YT=YT+BLACKY(I)
30    CONTINUE
40    DW=IABS(WHITEX(1)*PC-XT)
      IF(IABS(WHITEY(1)*PC-YT).GT.DW) DW=IABS(WHITEY(1)*PC-YT)
      DB=IABS(BLACKX(1)*PC-XT)
      IF(IABS(BLACKY(1)*PC-YT).GT.DB) DB=IABS(BLACKY(1)*PC-YT)
      SCORE(PLY+3)=SCORE(PLY+3)+DB-DW
      RETURN
      END
      SUBROUTINE PRINTR
      IMPLICIT INTEGER (A-Z)
      CCMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
```

```
      COMMON /RESULT/ PRINCA(14,14) , IREFUT
      COMMON /IODATA/ MOVEST,MOVESC,NMOVES,NPOSIT
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      COMMON /ELIMD/ KT(20),SAVED(14,200),VECTOR(14,200),FIRSTO(20),SER
     1H,SERCHS
      COMMON /DEBUG/ MSGLVL
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
       IF (MSGLVL.EQ.0) RETURN
      J=1
14    IF(PRINCA(1,J).EQ.-1) GOTO 15
      IF (J.EQ.14) GOTO 15
      J=J+1
      GOTO 14
15    WRITE(6,17) (PRINCA(1,I), I=1,J)
   17 FORMAT(/,' THE PRINCIPAL CONTINUATION IS ',/,(9(I9,'-')))
      WRITE(6,18) NPOSIT,NMOVES,MOVEST,MOVESC,DRAWC1,DRAWC2,SCORE(3)
   18 FORMAT(' TERM =',I7,' NONTER =',I7,
     1' TRANS MOVES=',I7,' CAPTURES=',I7,/
     1' 3-PLY DRAWN SEQ=',I5,' 4-PLY DRAWN SEQ=',I5,' SCORE=',I6)
      WRITE(6,19) IREFUT,KONTA,KONTS
   19 FORMAT(' REFUT. ON PC=',I6,' KING MOVES/KING ON 7=',I5,'/',I5)
      WRITE(6,20) D,MP(1),SERCH,SERCHS
   20 FORMAT(' D=',I2,'  MOVES=',I2,' 3-P CONT.=',I8,'  TN FOUND=',I8)
      RETURN
      END
      SUBROUTINE  UPDATE
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /ELIMD/ KT(20),SAVED(14,200),VECTOR(14,200),FIRSTO(20),SER
     1H,SERCHS
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /SKORE/ MATERW,MATERB
      C=(-1)*C
      P=MP(PLY)
      IMOVE=IABS(MOVE(P))
      IT=MOD(IMOVE,10)
      IP=MOVE(P)/10000000
      KT(PLY)=0
      IF(IABS(IP).NE.6)  GO TO 20
      IF(IT.EQ.0) KT(PLY)=1
   20 NX=MOD(IMOVE/10000,10)
      NY=MOD(IMOVE/1000,10)
      OX=MOD(IMOVE/100,10)
      OY=MOD(IMOVE/10,10)
      BOARD(OX,OY)=0
      BOARD(NX,NY)=IP
      IF(IT.EQ.2) BOARD(NX,NY+C)=0
      IF(IT.EQ.3)  BOARD(NX,NY)=-5*C
      POINT=MOD(IMOVE/100000,10)
      IF(IP.LT.0)  GO TO 100
      WHITEX(POINT)=NX
      WHITEY(POINT)=NY
      IF(IT.EQ.0)  RETURN
      IF(IT.EQ.3)  GO TO 50
      POINT2=MOD(IMOVE/1000000,10)
      CAPTY(PLY)=BLACKP(POINT2)
```

```
      MATERB=MATERB-BLACKP(POINT2)
      BLACKP(POINT2)=+1
      RETURN
   50 WHITEP(POINT)=9
      MATERW=MATERW+8
      RETURN
  100 BLACKX(POINT)=NX
      BLACKY(POINT)=NY
      IF(IT.EQ.0)    RETURN
      IF(IT.EQ.3)  GO TO 150
      POINT2=MOD(IMOVE/1000000,10)
      CAPTY(PLY)=WHITEP(POINT2)
      MATERW=MATERW-WHITEP(POINT2)
      WHITEP(POINT2)=(-1)
      RETURN
  150 BLACKP(POINT)=-9
      MATERB=MATERB-8
      RETURN
      END
      SUBROUTINE   RESTOR
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /SKORE/ MATERW,MATERB
      COMMON /ELIMD/ KT(20),SAVED(14,200),VECTOR(14,200),FIRSTO(20),SER
     1H,SERCHS
      C=(-1)*C
      P=MP(PLY+1)
      IMOVE=IABS(MOVE(P))
      IF(PLY.LE.1) GO TO 25
      IF ((MATERW+IABS(MATERB).LE.2).AND.(PLY.GE.D-1)) GOTO 25
      IF(KT(PLY-1)+KT(PLY+1).NE.2)  GO TO 25
      KSQ=MOD(IMOVE/1000,100)
      OPPMV=MOD(IABS(MOVE(MP(PLY))),100000)
      CODE=100*OPPMV+KSQ
      K=PLY-1
      J=FIRSTO(K)
      VECTOR(K,J)=CODE
      SAVED(K,J)=SCORE(PLY+4)
      FIRSTO(K)=FIRSTO(K)+1
      IF(FIRSTO(K).GT.200) FIRSTO(K)=200
   25 IT=MOD(IMOVE,10)
      IP=MOVE(P)/10000000
      OX=MOD(IMOVE/100,10)
      OY=MOD(IMOVE/10,10)
      NX=MOD(IMOVE/10000,10)
      NY=MOD(IMOVE/1000,10)
      BOARD(OX,OY)=IP
      BOARD(NX,NY)=0
      IF(IT.EQ.1)   BOARD(NX,NY)=CAPTY(PLY+1)
      IF(IT.EQ.2)   BOARD(NX,NY+C)=CAPTY(PLY+1)
      IF(IT.EQ.3)   BOARD(OX,OY)=IP
      IF(IP.LT.0)  GO TO 100
      POINT=MOD(IMOVE/100000,10)
      WHITEX(POINT)=OX
      WHITEY(POINT)=OY
```

```
      IF(IT.EQ.3)  GO TO 50
      IF(IT.EQ.0)  RETURN
      MATERB=MATERB+CAPTY(PLY+1)
      POINT2=MOD(IMOVE/1000000,10)
      BLACKP(POINT2)=CAPTY(PLY+1)
      RETURN
   50 WHITEP(POINT)=1
      MATERW=MATERW-8
      RETURN
  100 POINT=MOD(IMOVE/100000,10)
      BLACKX(POINT)=OX
      BLACKY(POINT)=OY
      IF(IT.EQ.3)  GO TO 150
      IF(IT.EQ.0)  RETURN
      MATERW=MATERW+CAPTY(PLY+1)
      POINT2=MOD(IMOVE/1000000,10)
      WHITEP(POINT2)=CAPTY(PLY+1)
      RETURN
  150 BLACKP(POINT)=(-1)
      MATERB=MATERB+8
      RETURN
      END
      SUBROUTINE GENERT
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /QCHECK/ MCHECK,ICHECK,JCHECK
      COMMON /PASSEC/ PPSCOR,NUMBW,NUMBB,WHITEQ,BLACKQ,ADVW,ADVB,WPPP,E
     1PP
      COMMON /RESULT/ PRINCA(14,14),IREFUT
      MOVEP=50*PLY
      CALL CHECK
      CALL PAWNG
      CALL KINGG
      MOVE(MOVEP+1)=0
      RETURN
      END
      SUBROUTINE KINGG
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /SPEC/ KTYPE(8,8),BOARDI(64)
      IF(C.EQ.1)   GO TO 5
      X=BLACKX(1)
      Y=BLACKY(1)
      GO TO 7
    5 X=WHITEX(1)
      Y=WHITEY(1)
    7 Z=KTYPE(X,Y)
      GO TO (10,20,30,40,50,60,70,80,90) , Z
   10 CALL KING1(X,Y,X-1,Y+C)
      CALL KING1(X,Y,X+1,Y+C)
      CALL KING1(X,Y,X,Y+C)
      CALL KING1(X,Y,X-1,Y)
      CALL  KING1(X,Y,X+1,Y)
      CALL KING1(X,Y,X+1,Y-C)
      CALL KING1(X,Y,X-1,Y-C)
      CALL KING1(X,Y,X,Y-C)
```

```
      RETURN
   20 CALL KING1(X,Y,X+1,Y)
      CALL KING1(X,Y,X+1,Y-1)
      CALL KING1(X,Y,X,Y-1)
      RETURN
   30 CALL KING1(X,Y,X-1,Y)
      CALL KING1(X,Y,X-1,Y-1)
      GO TO 20
   40 CALL KING1(X,Y,X-1,Y)
      CALL KING1(X,Y,X-1,Y-1)
      CALL KING1(X,Y,X,Y-1)
      RETURN
   50 CALL KING1(X,Y,X-1,Y+1)
      CALL KING1(X,Y,X,Y+1)
      GO TO 40
   60 CALL KING1(X,Y,X-1,Y)
      CALL KING1(X,Y,X-1,Y+1)
      CALL KING1(X,Y,X,Y+1)
      RETURN
   70 CALL KING1(X,Y,X+1,Y)
      CALL KING1(X,Y,X+1,Y+1)
      GO TO 60
   80 CALL KING1(X,Y,X,Y+1)
      CALL KING1(X,Y,X+1,Y)
      CALL KING1(X,Y,X+1,Y+1)
      RETURN
   90 CALL KING1(X,Y,X,Y-1)
      CALL KING1(X,Y,X+1,Y-1)
      GO TO 80
      END
      SUBROUTINE KING1(X,Y,XN,YN)
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /IODATA/ MOVEST,MOVESC,NMOVES,NPOSIT
      IF(C.EQ.1)   GO TO 10
      IF(BOARD(XN,YN).LT.0)   RETURN
      XP=WHITEX(1)-XN
      YP=WHITEY(1)-YN
      GO TO 30
   10 IF(BOARD(XN,YN).GT.0)    RETURN
      XP=BLACKX(1)-XN
      YP=BLACKY(1)-YN
   30 DIST=IABS(XP)
      IF(DIST.LT.IABS(YP))    DIST=IABS(YP)
      IF(DIST.EQ.1)    RETURN
      IF(XN.EQ.8)    GO TO 20
      IF ((YN+C.LT.1).OR.(YN+C.GT.8)) GOTO 40
      IF(BOARD(XN+1,YN+C).EQ.(-C))   RETURN
   20 IF(XN.EQ.1)   GO TO 40
      IF ((YN+C.LT.1).OR.(YN+C.GT.8)) GOTO 40
      IF(BOARD(XN-1,YN+C).EQ.(-C))   RETURN
   40 IF(BOARD(XN,YN).EQ.0)   GO TO 50
      CALL CAPTUR(6*C,X,Y,XN,YN,1,1,IFIND(XN,YN))
      RETURN
   50 MOVEP=MOVEP+1
```

```
      MOVEST=MOVEST+1
      MOVE(MOVEP)=C*(10*(Y+10*(X+10*(YN+10*XN)))+60100000)
      RETURN
      END
      SUBROUTINE  PAWNG
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /OLDD/ LASTOX,LASTOY,LASTNX,LASTNY
      COMMON /QCHECK/ MCHECK,ICHECK,JCHECK
      COMMON /IODATA/ MOVEST,MOVESC,NMOVES,NPOSIT
      PAWNP=2
    1 IF(C.NE.1)    GO TO 7
      IF(WHITEP(PAWNP).EQ.0)    RETURN
      IF(WHITEP(PAWNP).NE.C)    GO TO 80
      X=WHITEX(PAWNP)
      Y=WHITEY(PAWNP)
      GO TO 20
    7 IF(BLACKP(PAWNP).EQ.0)    RETURN
      IF(BLACKP(PAWNP).NE.C)    GO TO 80
      X=BLACKX(PAWNP)
      Y=BLACKY(PAWNP)
   20 IF(BOARD(X,Y+C).NE.0)    GO TO 30
      IF(MCHECK.NE.0)   GO TO 30
      MOVEP = MOVEP + 1
      MOVEST=MOVEST+1
      MOVE(MOVEP)=C*10*(Y+10*(X+10*((Y+C)+10*(X+10*(PAWNP+100)))))
      IF((Y.EQ.2).AND.(C.NE.1))  MOVE(MOVEP)=MOVE(MOVEP)-3
      IF((Y.EQ.7).AND.(C.EQ.1))   MOVE(MOVEP)=MOVE(MOVEP)+3
      IF((Y.EQ.2).AND.(C.EQ.1))    GO TO 25
      IF((Y.EQ.7).AND.(C.NE.1))    GO TO 25
      GO TO 30
   25 IF(BOARD(X,Y+2*C).NE.0)    GO TO 30
      MOVEP = MOVEP + 1
      MOVEST=MOVEST+1
      MOVE(MOVEP)=C*10*(Y+10*(X+10*((Y+2*C)+10* (X+10*(PAWNP+100)))))
   30 IF(X.EQ.8)    GO TO 40
      IF((BOARD(X+1,Y+C)).NE.(-C))    GO TO 40
      IF(MCHECK.EQ.0)   GO TO 31
      IF((ICHECK.NE.(X+1)).OR.(JCHECK.NE.(Y+C)))  GO TO 40
   31 CALL CAPTUR(C,X,Y,X+1,Y+C,1,PAWNP,IFIND(X+1,Y+C))
   40 IF(X.EQ.1)    GO TO 50
      IF((BOARD(X-1,Y+C)).NE.(-C))    GO TO 50
      IF(MCHECK.EQ.0)   GO TO 41
      IF((ICHECK.NE.(X-1)).OR.(JCHECK.NE.(Y+C)))  GO TO 50
   41 CALL CAPTUR(C,X,Y,X-1,Y+C,1,PAWNP,IFIND(X-1,Y+C))
   50 IF(PLY.EQ.0)    GO TO 55
      P=MP(PLY)
      IF(IABS(MOD(IABS(MOVE(P))/10,10)-MOD(IABS(MOVE(P))/1000,10)).NE.2
     1GO TO 80
      IF((Y.EQ.5).AND.(C.EQ.1))    GO TO 51
      IF((Y.EQ.4).AND.(C.NE.1))     GO TO 51
      GO TO 80
   51 IIX=MOD(IABS(MOVE(P))/10000,10)
      IF(IABS(X-IIX).NE.1)   GO TO 80
      IF(MCHECK.EQ.0)   GO TO 53
```

```
      IF((ICHECK.NE.IIX).OR.(JCHECK.NE.Y))  GO TO 80
   53 CALL CAPTUR(C,X,Y,IIX,Y+C,2,PAWNP,IFIND(IIX,Y))
      GO TO 80
   55 IF((IABS(LASTOY-LASTNY)).NE.2) GO TO 80
      IF((Y.EQ.5).AND.(C.EQ.1))    GO TO 52
      IF((Y.EQ.4).AND.(C.EQ.-1))    GO TO 52
      GO TO 80
   52 IF((IABS(X-LASTNX)).NE.1)    GO TO 80
      IF(MCHECK.EQ.0)  GO TO 57
      IF((ICHECK.NE.LASTNX).OR.(JCHECK.NE.Y))  GO TO 80
   57 CALL CAPTUR(C,X,Y,LASTNX,Y+C,2,PAWNP,IFIND(LASTNX,Y))
   80 PAWNP=PAWNP+1
      GO TO 1
      END
      SUBROUTINE PLAUS(I)
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /RESULT/ PRINCA(14,14) , IREFUT
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /SKORE/ MATERW,MATERB
      IF ((PLY.EQ.0).AND.(MATERW+IABS(MATERB).GT.2)) GOTO 10
      CALL FILTER(I)
      CALL BACKW(I)
10    PLYT=PLY+1
      PLYR=PLY
      IF(PLY.EQ.0) PLYR=1
      J=I
   20 IF(MOVE(J).EQ.0) GO TO 200
      DO 25 L=PLYT,D,2
      DO 24 K=PLYR,L
      IF(MOVE(J).EQ.PRINCA(K,L))  GO TO 100
   24 CONTINUE
   25 CONTINUE
      J=J+1
      GO TO 20
  100 IF(MOD(MOVE(I),10).NE.0)  GO TO 199
      I1=MOVE(I)
      MOVE(I)=MOVE(J)
      MOVE(J)=I1
199   IREFUT=IREFUT+1
200   CONTINUE
      RETURN
      END
      INTEGER FUNCTION IFIND(XN,YN)
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      IF(C-1)  100,200,100
  100 DO 120 J=1,11
      IF(WHITEX(J).NE.XN)    GO TO 120
      IF(WHITEY(J).NE.YN)    GO TO 120
      IFIND=J
      RETURN
  120 CONTINUE
  200 DO 220 J=1,11
      IF(BLACKX(J).NE.XN)    GO TO 220
      IF(BLACKY(J).NE.YN)    GO TO 220
```

```
      IFIND=J
      RETURN
  220 CONTINUE
      CALL PRINTB
      WRITE(6,1000) PLY,C,XN,YN,MOVEP,PLY
 1000 FORMAT(1H ,'IFIND',6I2)
      STOP
      END
      SUBROUTINE CAPTUR(IPIECE,ICX,IOY,INX,INY,ITYPE,IPTO,IPTQ)
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /GENSP/ JENTYP(20)
      COMMON /IODATA/ MOVEST,MOVESC,NMOVES,NPOSIT
      MOVEP=MOVEP+1
      MOVESC=MOVESC+1
      IQ=50*PLY+1
      IF(JENTYP(PLY+1).NE.0)  IQ=MP(PLY+1)+1
    3 IF(MOVEP.EQ.IQ)  GO TO 30
      IF(MOD(MOVE(IQ),10).EQ.0)  GO TO 5
      IQ=IQ+1
      GO TO 3
    5 MOVE(MOVEP)=MOVE(IQ)
   30 MOVE(IQ)=C*(ITYPE+10*(IOY+10*(IOX+10*(INY+10*(INX+10*(IPTO+10*(IP
     1Q+10*IABS(IPIECE)))))))
      RETURN
      END
      SUBROUTINE CHECK
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /QCHECK/ MCHECK,ICHECK,JCHECK
      MCHECK=0
      IF(PLY.EQ.0)  GO TO 33
      IF(MOVE(MP(PLY))/10000000.NE.-C)  RETURN
   33 IF(C.EQ.1)  GO TO 37
      I=BLACKX(1)
      J=BLACKY(1)
      IF(J.EQ.1)  RETURN
      GO TO 40
   37 I=WHITEX(1)
      J=WHITEY(1)
      IF(J.EQ.8)  RETURN
   40 IF(I.EQ.1)  GO TO 50
      IF(BOARD(I-1,J+C).NE.(-C))  GO TO 50
      ICHECK=I-1
      GO TO 55
   50 IF(I.EQ.8)  RETURN
      IF(BOARD(I+1,J+C).NE.(-C))  RETURN
      ICHECK=I+1
   55 MCHECK=1
      JCHECK=J+C
      RETURN
      END
      SUBROUTINE DRAWN
      IMPLICIT INTEGER (A-Z)
```

```
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /SKORE/ MATERW,MATERB
      COMMON /ELIMD/ KT(20),SAVED(14,200),VECTOR(14,200),FIRSTO(20),SER
     1H,SERCHS
      DRAWT=0
      IF((MATERW.EQ.0).AND.(MATERB.EQ.0)) GOTO 50
      IF(PLY.LT.3)  RETURN
      IOS2=MOD(IABS(MOVE(MP(PLY-2)))/10,100)
      INS0=MOD(IABS(MOVE(MP(PLY)))/1000,100)
      IF(IOS2.NE.INS0)  RETURN
      IF(KT(PLY)+KT(PLY-1)+KT(PLY-2).NE.3) RETURN
      IF(PLY.LE.4)  GO TO 40
      IF(KT(PLY-4).NE.1)  GO TO 40
      IOS3=MOD(IABS(MOVE(MP(PLY-3)))/10,100)
      INS1=MOD(IABS(MOVE(MP(PLY-1)))/1000,100)
      IF(IOS3.NE.INS1)  GO TO 40
      DRAWC1=DRAWC1+1
      DRAWT=1
      SCORE(PLY+3)=PLY*(-C)
      RETURN
40    IF (C*SCORE(PLY+2).GT.0) RETURN
      DRAWC2=DRAWC2+1
50    SCORE(PLY+3)=0
      DRAWT=1
      RETURN
      END
      SUBROUTINE FILTER(I)
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      IF (PLY.EQ.0) GOTO 50
      IF(MOD(IABS(MOVE(MP(PLY))),10).NE.3)  GO TO 50
      IF(MOVE(I).EQ.0)  RETURN
      II=I
20    II=II+1
      IF(MOD(MOVE(II),10).NE.0)  GO TO 20
      MOVE(II)=0
50    J=I
60    IMOVE=IABS(MOVE(J))
      IF(IMOVE.GT.60000000)  GOTO 70
      IF (MOVE(J).EQ.0) RETURN
      J=J+1
      GOTO 60
70    KONTA=KONTA+1
      KY=IMOVE/10
      IF(MOD(KY,10).EQ.2)  GO TO 100
      IF(MOD(KY,10).EQ.7)  GO TO 100
      KX=IMOVE/100
      IF(MOD(KX,10).EQ.2)  GO TO 100
      IF(MOD(KX,10).EQ.7)  GO TO 100
      RETURN
100   KK=J
105   IF(MOD(MOVE(KK),10).NE.0)  RETURN
```

```
      IF(MOVE(KK).EQ.0)  GO TO 200
      KK=KK+1
      GO TO 105
  200 KCONT=KK-J
      IF(KCONT.NE.8)  RETURN
      KONTS=KONTS+1
      KK=KK-1
      II=J
  210 IF(II.GE.KK)  GO TO 400
      MI=IABS(MOVE(II))
      MXN=MOD(MI/10000,10)
      IF(MXN.EQ.1)  GO TO 220
      IF(MXN.EQ.8)  GO TO 220
      MYN=MOD(MI/1000,10)
      IF(MYN.EQ.1)  GO TO 220
      IF(MYN.EQ.8)  GO TO 220
      II=II+1
      GO TO 210
  220 KI=IABS(MOVE(KK))
      KXN=MOD(KI/10000,10)
      IF(KXN.EQ.1)  GO TO 230
      IF(KXN.EQ.8)  GO TO 230
      KYN=MOD(KI/1000,10)
      IF(KYN.EQ.1)  GO TO 230
      IF(KYN.EQ.8)  GO TO 230
      GO TO 240
  230 KK=KK-1
      IF(KK.EQ.II)  GO TO 400
      GO TO 220
  240 MM=MOVE(KK)
      MOVE(KK)=MOVE(II)
      MOVE(II)=MM
      KK=KK-1
      II=II+1
      GO TO 210
  400 IF (MOVE(KK).EQ.0) RETURN
      MK=IABS(MOVE(KK))
      IF(MOD(MK/1000,10).EQ.1)  GO TO 500
      IF(MOD(MK/1000,10).EQ.8)  GO TO 500
      IF(MOD(MK/10000,10).EQ.1)  GO TO 500
      IF(MOD(MK/10000,10).EQ.8)  GO TO 500
      KK=KK+1
      GO TO 400
  500 MOVE(KK)=0
      RETURN
      END
      SUBROUTINE PASSED
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PASSEC/ PPSCOR,NUMBW,NUMBB,WHITEQ,BLACKQ,ADVW,ADVB,WPPP,B
     1PP
      DIMENSION WPPAWN(8),BPPAWN(8),DISTKW(8),DISTKB(8),DISTW(8),DISTB(
     1)
      COMMON /ADV/ WADV(20),BADV(20),WLFT(20),BLFT(20),WRT(20),BRT(20)
      COMMON /SKORE/ MATERW,MATERB
      PPSCOR=0
```

```
      WHITEQ=10
      BLACKQ=10
      ADVB=9
      ADVW=0
      BLEFT=0
      WLEFT=0
      BRIGHT=9
      WRIGHT=9
      WPPP=0
      P=2
5     IF(P.GT.NUMBW)  GO TO 100
      IF(WHITEP(P).NE.1)  GO TO 50
      Q=2
   10 IF(Q.GT.NUMBB)  GO TO 40
      IF(BLACKP(Q).NE.-1)  GO TO 20
      IF(BLACKY(Q).LE.WHITEY(P))  GO TO 20
      IF(IABS(BLACKX(Q)-WHITEX(P)).LE.1)  GO TO 52
   20 Q=Q+1
      GO TO 10
   40 IF(BLACKX(1).NE.WHITEX(P))  GO TO 42
      IF(BLACKY(1).GT.WHITEY(P))  GO TO 52
   42 IF(WHITEX(1).NE.WHITEX(P))  GO TO 44
      IF(WHITEY(1).GT.WHITEY(P))  GO TO 52
   44 WPPP=WPPP+1
      WPPAWN(WPPP)=P
      DISTW(WPPP)=8-WHITEY(P)
   52 IF(WHITEY(P).GT.ADVW)  ADVW=WHITEY(P)
      IF(WHITEX(P).GT.WLEFT)  WLEFT=WHITEX(P)
      IF(WHITEX(P).LT.WRIGHT)  WRIGHT=WHITEX(P)
50     P=P+1
      GO TO 5
100    BPPP=0
      ADVW=8-ADVW
      P=2
105    IF(P.GT.NUMBB)  GO TO 400
      IF(BLACKP(P).NE.-1)  GO TO 150
      Q=2
  110 IF(Q.GT.NUMBW)  GO TO 140
      IF(WHITEP(Q).NE.1)  GO TO 120
      IF(WHITEY(Q).GE.BLACKY(P))  GO TO 120
      IF(IABS(BLACKX(P)-WHITEX(Q)).LE.1)  GO TO 152
  120 Q=Q+1
      GO TO 110
  140 IF(WHITEX(1).NE.BLACKX(P))  GO TO 142
      IF(WHITEY(1).LT.WHITEY(P))  GO TO 152
  142 IF(BLACKX(1).NE.BLACKX(P))  GO TO 144
      IF(BLACKY(1).LT.BLACKY(P))  GO TO 152
  144 BPPP=BPPP+1
      BPPAWN(BPPP)=P
      DISTB(BPPP)=BLACKY(P)-1
  152 IF(BLACKY(P).LT.ADVB)  ADVB=BLACKY(P)
      IF(BLACKX(P).GT.BLEFT)  BLEFT=BLACKX(P)
      IF(BLACKX(P).LT.BRIGHT)  BRIGHT=BLACKX(P)
150   P=P+1
      GO TO 105
  400 ADVB=ADVB-1
      IF(WPPP.EQ.0)  GO TO 600
      P=1
```

```
      KX=BLACKX(1)
      KY=BLACKY(1)
520    QSX=WHITEX(WPPAWN(P))
      DIST=IABS(KX-QSX)
      IF(DIST.LT.(8-KY))  DIST=8-KY
      DISTKW(P)=DIST
      IF(P.EQ.WPPP)  GO TO 600
      P=P+1
      GO TO 520
600    IF(BPPP.EQ.0)  GO TO 800
      P=1
      KX=WHITEX(1)
      KY=WHITEY(1)
620    QSX=BLACKX(BPPAWN(P))
      DIST=IABS(KX-QSX)
      IF(DIST.LT.(KY-1))  DIST=KY-1
      DISTKB(P)=DIST
      IF(P.EQ.BPPP)   GO TO 800
      P = P + 1
      GO TO 620
  800 IF(BPPP.EQ.0)  GO TO 1000
      P=1
  720 DIFF=DISTKB(P)-DISTB(P)
      IF(C.NE.1)  GO TO 730
      DIFF=DIFF-1
  730 IF(DIFF.LE.0)  GO TO 760
      IF(DISTB(P).LT.BLACKQ)   BLACKQ=DISTB(P)
  760 IF(P.EQ.BPPP)   GO TO 1000
      P=P+1
      GO TO 720
 1000 IF(WPPP.EQ.0)  GO TO 2000
      P=1
  820 DIFF=DISTKW(P)-DISTW(P)
      IF(C.EQ.1)  GO TO  830
      DIFF=DIFF-1
  830 IF(DIFF.LE.0)  GO TO 860
      IF(DISTW(P).LT.WHITEQ)  WHITEQ=DISTW(P)
  860 IF(P.EQ.WPPP)  GO TO 2000
      P=P+1
      GO TO 820
 2000 WADV(PLY+1)=ADVW
      BADV(PLY+1)= ADVB
      WLFT(PLY+1)=WLEFT
      BLFT(PLY+1)=BLEFT
      WRT(PLY+1)=WRIGHT
      BRT(PLY+1)=BRIGHT
      IF(C.EQ.-1)  GO TO 2200
      I=WHITEQ-BLACKQ
      IF(I.GE.0) GO TO 2020
      J=WHITEQ-ADVB
      IF(J.GE.0)  GO TO 2020
2010  IF (MATERW+IABS(MATERB).LE.2) RETURN
      PPSCOR=6000-200*PLY-400*WHITEQ-200*MOD(PLY,2)
      RETURN
 2020 I=BLACKQ+1-WHITEQ
      IF(I.GE.0)  RETURN
      J=BLACKQ+1-ADVW
      IF(J.LT.0)  GO TO 2210
```

```
      RETURN
 2200 I=BLACKQ-WHITEQ
      IF(I.GE.0) GO TO 2220
      J=BLACKQ-ADVW
      IF(J.GE.0)  GO TO 2220
2210  IF (MATERW+IABS(MATERB).LE.2) RETURN
      PPSCOR=-6000+200*PLY+400*BLACKQ+200*MOD(PLY+1,2)
      RETURN
 2220 I=WHITEQ+1-BLACKQ
      IF(I.GE.0)  RETURN
      J=WHITEQ+1-ADVB
      IF(J.LT.0)  GO TO 2010
      RETURN
      END
      SUBROUTINE BACKW(I)
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /PASSEC/ PPSCOR,NUMBW,NUMBB,WHITEQ,BLACKQ,ADVW,ADVB,WPPP,B
     1PP
      COMMON /ADV/ WADV(20),BADV(20),WLFT(20),BLFT(20),WRT(20),BRT(20)
      COMMON /SKORE/ MATERW,MATERB
      IF (PLY.EQ.0) GOTO 1
      IF (MOD(IABS(MOVE(MP(PLY))),10).EQ.3) RETURN
1     J=I
5     IF(IABS(MOVE(J)).GT.60000000)  GOTO 10
      IF (MOVE(J).EQ.0) RETURN
      J=J+1
      GOTO 5
10    IF(C.NE.1)  GO TO 40
      MOST=BADV(PLY+1)+1
      IF(MOST.GT.BLACKY(1))  MOST=BLACKY(1)
      IF ((MATERW.NE.1).OR.(MATERB.NE.-1)) GOTO 19
      P=2
11    IF (WHITEP(P).EQ.1) GOTO 12
      P=P+1
      GOTO 11
12    IF ((WHITEY(1).EQ.WHITEY(P)).AND.
     1(IABS(WHITEX(1)-WHITEX(P)).EQ.1)) GOTO 19
      MOST=MOST+1
19    IF(WHITEY(1).GT.MOST-3)  GO TO 100
      ROW=WHITEY(1)
      GO TO 50
   40 MOST=8-WADV(PLY+1)
      IF(MOST.LT.WHITEY(1))  MOST=WHITEY(1)
      IF ((MATERW.NE.1).OR.(MATERB.NE.-1)) GOTO 49
      P=2
41    IF (BLACKP(P).EQ.-1) GOTO 42
      P=P+1
      GOTO 41
42    IF ((BLACKY(1).EQ.BLACKY(P)).AND.
     1(IABS(BLACKX(1)-BLACKX(P)).EQ.1)) GOTO 49
      MOST=MOST-1
49    IF(BLACKY(1).LT.MOST+3)  GO TO 100
      ROW=BLACKY(1)
   50 II=J
```

```
   55 IF(MOVE(II).EQ.0)  GO TO 100
      NROW=MOD(IABS(MOVE(II)/1000),10)
      IF(NROW-ROW.EQ.-C)  GO TO 60
      IF (NROW.EQ.ROW) GOTO 400
   58 II=II+1
      GO TO 55
   60 JJ=II
   61 IF(MOVE(JJ+1).EQ.0)  GO TO 70
      MOVE(JJ)=MOVE(JJ+1)
      JJ=JJ+1
      GO TO 61
   70 MOVE(JJ)=0
      GO TO 55
  100 LEFT=BLFT(PLY+1)
      IF(LEFT.LT.WLFT(PLY+1)) LEFT=WLFT(PLY+1)
      IF(C.EQ.1)  GO TO 140
      IF (WHITEX(1).GT.LEFT) LEFT=WHITEX(1)
      IF(BLACKX(1).LE.LEFT+1)  GO TO 200
      COL=BLACKX(1)
      GO TO 150
140   IF (BLACKX(1).GT.LEFT) LEFT=BLACKX(1)
      IF(WHITEX(1).LE.LEFT+1)  GO TO 200
      COL=WHITEX(1)
  150 II=J
  155 IF(MOVE(II).EQ.0)  GOTO 300
      NCOL=MOD(IABS(MOVE(II)/10000),10)
      IF(NCOL-COL.GT.0)  GO TO 160
      IF (NCOL.EQ.COL) GOTO 420
  158 II=II+1
      GO TO 155
  160 JJ=II
  161 IF(MOVE(JJ+1).EQ.0)  GO TO 170
      MOVE(JJ)=MOVE(JJ+1)
      JJ=JJ+1
      GO TO 161
  170 MOVE(JJ)=0
      GO TO 155
  200 RIGHT=BRT(PLY+1)
      IF(RIGHT.GT.WRT(PLY+1)) RIGHT=WRT(PLY+1)
      IF(C.EQ.1)  GO TO 240
      IF (WHITEX(1).LT.RIGHT) RIGHT=WHITEX(1)
      IF(BLACKX(1).GE.RIGHT-1) GOTO 300
      COL=BLACKX(1)
      GO TO 250
240   IF (BLACKX(1).LT.RIGHT) RIGHT=BLACKX(1)
      IF(WHITEX(1).GE.RIGHT-1)  GOTO 300
      COL=WHITEX(1)
  250 II=J
  255 IF(MOVE(II).EQ.0) GOTO 300
      NCOL=MOD(IABS(MOVE(II)/10000),10)
      IF (NCOL-COL.LT.0) GOTO 260
      IF (NCOL.EQ.COL) GOTO 430
258   II=II+1
      GOTO 255
260   JJ=II
261   IF (MOVE(JJ+1).EQ.0) GOTO 270
      MOVE(JJ)=MOVE(JJ+1)
      JJ=JJ+1
```

```
      GOTO 261
270   MOVE(JJ)=0
      GOTO 255
300   Q=2
      IF ((MATERW.NE.1).OR.(MATERB.NE.-1)) RETURN
      IF (C.NE.1) GOTO 330
      IF (WHITEX(1).EQ.WHITEX(P)) RETURN
311   IF (BLACKP(Q).EQ.-1) GOTO 312
      Q=Q+1
      GOTO 311
312   IF (BLACKY(Q).EQ.2) RETURN
      LEAST=BLACKY(Q)
      IF (LEAST.LT.BLACKY(1)) LEAST=BLACKY(1)
      IF (WHITEY(1).LT.LEAST+1) RETURN
      ROW=WHITEY(1)
      GOTO 350
330   IF (BLACKX(1).EQ.BLACKX(P)) RETURN
341   IF (WHITEP(Q).EQ.1) GOTO 342
      Q=Q+1
      GOTO 341
342   IF (WHITEY(Q).EQ.7) RETURN
      LEAST=WHITEY(Q)
      IF (LEAST.GT.WHITEY(1)) LEAST=WHITEY(1)
      IF (BLACKY(1).GT.LEAST-1) RETURN
      ROW=BLACKY(1)
350   II=J
355   IF (MOVE(II).EQ.0) RETURN
      NROW=MOD(IABS(MOVE(II)/1000),10)
      IF (NROW-ROW.EQ.C) GOTO 360
      IF (NROW.EQ.ROW) GOTO 440
358   II=II+1
      GOTO 355
360   JJ=II
361   IF (MOVE(JJ+1).EQ.0) GOTO 370
      MOVE(JJ)=MOVE(JJ+1)
      JJ=JJ+1
      GOTO 361
370   MOVE(JJ)=0
      GOTO 355
400   IF (C.NE.1) GOTO 410
      IF (ROW.GT.MOST-4) GOTO 58
      GOTO 60
410   IF (ROW.LT.MOST+4) GOTO 58
      GOTO 60
420   IF (COL.LE.LEFT+2) GOTO 158
      GOTO 160
430   IF (COL.GE.RIGHT-2) GOTO 258
      GOTO 260
440   IF (WHITEY(1).EQ.BLACKY(1)+2)  GOTO 358
      IF (C.NE.1) GOTO 450
      IF (ROW.LT.LEAST+2) GOTO 358
      GOTO 360
450   IF (ROW.GT.LEAST-2) GOTO 358
      GOTO 360
      END
      SUBROUTINE READM(YOURM,OX,OY,NX,NY,MAN,T)
      LCGICAL*1 JJ(8),II(4)
      INTEGER OX,OY,T
```

```
      REAL*8 J,YOURM
      EQUIVALENCE (JJ,J),(II,I)
      I=0
      J=YOURM
      MAN=6
      II(4)=JJ(1)
      IF (I.NE.215) GOTO 20
      MAN=1
20    II(4)=JJ(2)
      OX=201-I
      II(4)=JJ(3)
      OY=I-240
      II(4)=JJ(5)
      NX=201-I
      II(4)=JJ(6)
      NY=I-240
      II(4)=JJ(4)
      IF (I.EQ.96) GOTO 60
      T=1
      II(4)=JJ(7)
      IF (I.EQ.64) RETURN
      T=2
      RETURN
60    T=0
      IF ((NY.NE.1).AND.(NY.NE.8)) RETURN
      IF (MAN.EQ.1) T=3
      RETURN
      END
      SUBROUTINE GETMOV(YOURM,OX,OY,NX,NY,MAN,T,*)
      LOGICAL*1 JJ(8),II(4),KK(4)
      INTEGER OX,OY,T
      REAL*8 J,YOURM
      EQUIVALENCE (JJ,J),(II,I),(KK,K)
      I=0
      K=0
      J=YOURM
      II(4)=JJ(1)
      IF (I.NE.215) GOTO 20
      MAN=1
      GOTO 30
20    IF (I.NE.210) RETURN1
      MAN=6
30    II(4)=JJ(2)
      OX=201-I
      IF ((OX.LT.1).OR.(OX.GT.8)) RETURN1
      II(4)=JJ(3)
      OY=I-240
      IF ((OY.LT.1).OR.(OY.GT.8)) RETURN1
      II(4)=JJ(5)
      NX=201-I
      IF ((NX.LT.1).OR.(NX.GT.8)) RETURN1
      II(4)=JJ(6)
      NY=I-240
      IF ((NY.LT.1).OR.(NY.GT.8)) RETURN1
      II(4)=JJ(4)
      IF (I.EQ.96) GOTO 60
      IF (I.NE.122) RETURN1
      T=1
```

```
      II(4)=JJ(7)
      IF (I.EQ.64) RETURN
      KK(4)=JJ(8)
      IF ((I.NE.197).OR.(K.NE.215)) RETURN1
      T=2
      RETURN
60    II(4)=JJ(7)
      IF (I.NE.64) RETURN1
      T=0
      IF ((NY.NE.1).AND.(NY.NE.8)) RETURN
      IF (MAN.EQ.1) T=3
      RETURN
      END
      SUBROUTINE PMOVE(MOVE,YOURM)
      COMMON /COLOR/ CCLOUR
      INTEGER COLOUR
      REAL*8 J,YOURM
      LOGICAL*1 JJ(8),X(8)/'A','B','C','D','E','F','G','H'/,DEL(6)/'P',
     C ',':','-','E','K'/,Y(8)/'1','2','3','4','5','6','7','8'/
      EQUIVALENCE (JJ,J)
      MOV=MOVE
      IF (MOV.LT.0) MOV=-MOVE
      JJ(7)=DEL(2)
      JJ(8)=DEL(2)
      I=0
5     I=I+1
      K=MOD(MOV,10)
      MOV=(MOV-K)/10
      GOTO (10,60,70,40,50,5,75),I
10    IF ((K.EQ.0).OR.(K.EQ.3)) GOTO 20
      JJ(4)=DEL(3)
      IF (K.EQ.1) GOTO 5
      JJ(7)=DEL(5)
      JJ(8)=DEL(1)
      GOTO 5
20    JJ(4)=DEL(4)
      GOTO 5
40    JJ(6)=Y(9-K+COLOUR*(2*K-9))
      GOTO 5
50    JJ(5)=X(9-K)
      GOTO 5
60    JJ(3)=Y(9-K+COLOUR*(2*K-9))
      GOTO 5
70    JJ(2)=X(9-K)
      GOTO 5
75    JJ(1)=DEL(MOV)
      YOURM=J
      WRITE (6,80) J
80    FORMAT (1H0,' MY MOVE IS   ',A8)
      RETURN
      END
      SUBROUTINE GETPOS
       COMMON /GAME/ WM(100),BM(100),STPOS(8,8),COUNT
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      INTEGER STPOS,COUNT
      INTEGER MP,SCORE,CAPTY,BOARD,MOVES(20)
       REAL*8 WM,BM
      LOGICAL*1 JJ(4),II(4)
```

```
      EQUIVALENCE (JJ,J),(II,I)
      DO 1 M=1,8
      DO 1 N=1,8
      BOARD(M,N)=0
1     CONTINUE
      KK=1
      LL=1
5     LP=0
      READ(9,10) (MOVES(K),K=1,20)
10    FORMAT(20A4)
30    LP=LP+1
      IF(MOVES(LP).GT.0) GOTO 50
      J=MOVES(LP)
      I=0
      II(4)=JJ(1)
      MAN=I-214
      IF (MAN.NE.1) GOTO 35
      GOTO 40
35    IF (MAN.NE.-4) GOTO 45
      MAN=6
40    II(4)=JJ(2)
      IX=201-I
      IF((IX.LT.1).OR.(IX.GT.8)) GOTO 45
      II(4)=JJ(3)
      IY=I-240
      IF((IY.LT.1).OR.(IY.GT.8)) GOTO 45
      GOTO 49
45    WRITE(6,46)
46    FORMAT (' *** INPUT ERROR RENTER')
      GOTO 5
49    BOARD(IX,IY)=MAN*KK
      GOTO 30
50    IF (LL.EQ.2) GOTO 100
      LL=LL+1
      KK=-1
      GOTO 5
100     DO 200 M=1,8
       DO 200 N=1,8
      STPOS(M,N)=BOARD(M,N)
200    CONTINUE
       CALL DUAL
      RETURN
      END
       SUBROUTINE UPDA(OX,OY,NX,NY,MAN,KIND)
       COMMON /DEEP/ DEPTH
       COMMON /DLIST/  MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
       COMMON/COLOR/ COLOUR
       INTEGER COLOUR
       INTEGER MP,SCORE,CAPTY,BOARD,DEPTH,OX,OY
       IF (DEPTH.EQ.1) GOTO 1
        MAN=-MAN
       NY=9-NY+COLOUR*(2*NY-9)
       OY=9-OY+COLOUR*(2*OY-9)
1       BOARD(OX,OY)=0
       BOARD(NX,NY)=MAN
       IF (KIND.EQ.2) BOARD(NX,NY-DEPTH)=0
       IF (KIND.EQ.3) BOARD(NX,NY)=5*DEPTH
       DEPTH=-DEPTH
```

```
      RETURN
      END
      SUBROUTINE USER(INPUT,*,*,*)
      REAL*8 MSGL
      REAL*8 INPUT,MASS,RES,PB,WM,BM
      INTEGER COUNT,DEPTH,WORD(2),OX,OY,T
      INTEGER MP,SCORE,CAPTY,BCARD,STPOS
      INTEGER PLY,SCOLOR,C,MOVEP,D,COLOUR,COL
      LOGICAL*1 LIST,RESIGN(8),MM(8),PBOARD(8),DRAW(4),END(4),
     1GOTO(4),II(4),JJ(4),DP(2),LAST,MSG(8)
      INTEGER*2 HWORD(4),KDP
      COMMON /COLOR/ CCLOUR
      COMMON /DEBUG/ MSGLVL
      COMMON /GAME/ WM(100),BM(100),STPOS(8,8),COUNT
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DEEP/ DEPTH
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      EQUIVALENCE (II,III),(JJ,JJJ),(END,KEND),(DRAW,KDRAW)
      EQUIVALENCE (RESIGN,RES),(PBOARD,PB),(GOTO,KGOTO)
      EQUIVALENCE (MASS,WORD,MM,HWORD),(DP,KDP)
      EQUIVALENCE (MSGL,MSG)
      DATA LIST/'L'/,RESIGN/'R','E','S','I','G','N',' ',' '/
     1,PBOARD/'P','B','O','A','R','D',' ',' '/,DRAW/'D','R'
     1,'A','W'/,END/'E','N','D',' '/,GOTO/'G','O','T','O'/
     1,DP/'D','='/,MSG/'M','S','G','L','V','L','=',' '/
      III=0
      JJJ=0
      PRINT19
19      FORMAT(1X,'*')
      READ(9,1) INPUT
      IF (MSGLVL.EQ.0) GOTO 2
      WRITE(6,11) INPUT
  11  FORMAT(1X,A8)
1     FORMAT(A8)
2     MASS=INPUT
      IF (WORD(1).EQ.KEND) STOP50
      IF (MASS.EQ.RES) RETURN1
      IF (WORD(1).EQ.KDRAW) RETURN2
      IF (MASS.NE.PB) GOTO 5
      CALL PRINTB
      RETURN3
5     IF (HWORD(1).NE.KDP) GOTO 10
      LP=3
      CALL NUMBER(INPUT,LP,NUM,&1000)
      D=NUM
      RETURN3
10    IF(WORD(1).NE.KGOTO) GOTO 50
      LP=6
      CALL NUMBER(INPUT,LP,NUM,&1000)
      IF ((NUM.EQ.0).OR.(NUM.GT.COUNT)) GOTO 1000
      DO 15 I=1,8
      DO 15 J=1,8
      BOARD(I,J)=STPOS(I,J)
15    CONTINUE
      DEPTH=1
      NUMM=NUM-COLOUR
      IF (NUMM.EQ.0) GOTO 25
      COL=COLOUR
```

```
       COLOUR=1
      DO 20 I=1,NUMM
      CALL READM(WM(I),OX,OY,NX,NY,MAN,T)
      CALL UPDA(OX,OY,NX,NY,MAN,T)
      CALL READM(BM(I),OX,OY,NX,NY,MAN,T)
      CALL UPDA(OX,OY,NX,NY,MAN,T)
20    CONTINUE
       COLOUR=COL
      IF (COLOUR.EQ.0) GOTO 26
25    CALL READM(WM(NUM),OX,OY,NX,NY,MAN,T)
      CALL UPDA(OX,OY,NX,NY,MAN,T)
26    CALL DUAL
      COUNT=NUM
       DEPTH=-1
      RETURN3
50     II(4)=LIST
      JJ(4)=MM(1)
      IF (III.NE.JJJ) GOTO 100
      LP=3
      II(4)=END(4)
      JJ(4)=MM(3)
      IF (III.NE.JJJ) GOTO 60
      LIMIT1=1
      LIMIT2=COUNT
      GOTO 70
60     CALL NUMBER(INPUT,LP,LIMIT1,&1000)
      LIMIT2=LIMIT1
      JJ(4)=MM(LP+1)
      IF (III.EQ.JJJ) GOTO 65
      LP=LP+1
      CALL NUMBER(INPUT,LP,LIMIT2,&1000)
65    IF ((LIMIT1.LT.1).OR.(LIMIT2.GT.COUNT).OR.(LIMIT2.LT.LIMIT1)) GOT
     C1000
70    DO 80 I=LIMIT1,LIMIT2
      WRITE(6,2001) I,WM(I),BM(I)
80    CONTINUE
      RETURN3
2001  FORMAT(1H0,I2,2X,A8,6X,A8)
100   II(4)=MM(8)
      MM(8)=END(4)
      IF (MASS.NE.MSGL) GOTO 200
      NUM=III-240
      IF ((NUM.LT.0).OR.(NUM.GT.10)) GOTO 1000
      MSGLVL=NUM
      RETURN3
200   MM(8)=II(4)
      CALL GETMOV(INPUT,OX,OY,NX,NY,MAN,T,&1000)
      CALL UPDA(OX,OY,NX,NY,MAN,T)
      RETURN
1000  PRINT1001
1001  FORMAT(1H0,'  INPUT ERROR REENTER')
      RETURN3
      END
      SUBROUTINE NUMBER(INPUT,LP,N,*)
      REAL*8 INPUT,J
      LOGICAL*1 JJ(8),II(4)
      EQUIVALENCE (J,JJ),(I,II)
      J=INPUT
```

```
      I=0
      N=0
      LPNEW=LP+2
      DO 10 K=LP,LPNEW
      II(4)=JJ(K)
      NUM=I-240
      IF (NUM.NE.-176) GOTO 5
      RETURN
5     IF((NUM.LT.0).OR.(NUM.GT.9)) RETURN1
      N=N*10+NUM
10    CONTINUE
      LP=LP+3
      RETURN
      END
      SUBROUTINE NEWCNE
      COMMON /COLOR/ CCLOUR
      INTEGER COLOUR
      LOGICAL*1 YES/'Y'/,NO/'N'/,INT(4)
      EQUIVALENCE (INT,IN)
      COLOUR=0
      IN=0
95    FORMAT(A1)
10    PRINT1000
1000  FORMAT (1H0,'*')
11    PRINT101
101   FORMAT(1H0,1X,'PICK YOUR COLOUR ( ''W'' OR ''B'')')
      READ(9,95) INT(4)
      IF (IN.EQ.230) GOTO 200
      IF (IN.NE.194) GOTO 11
      COLOUR=1
200    PRINT1000
       PRINT201
       PRINT1000
201    FORMAT(1H0,1X,' TYPE POSITION')
      RETURN
      END
      SUBROUTINE DUAL
      COMMON /COLOR/ COLOUR
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      INTEGER MP,SCORE,CAPTY,BOARD,CCLOUR
      IF (COLOUR.EQ.1) RETURN
      DO 10 J=1,4
      DO 10 I=1,8
      NEW=-BOARD(I,9-J)
      BOARD(I,9-J)=-BOARD(I,J)
      BOARD(I,J)=NEW
10    CONTINUE
      RETURN
      END
      SUBROUTINE PRINTB
      COMMON /COLOR/ COLOUR
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      INTEGER MP,SCORE,CAPTY,BOARD,CCLOUR
      IF (COLOUR.EQ.1) GOTO 10
      CALL DUAL
10     WRITE(6,101) ((BCARD(9-I,9-J),I=1,8),J=1,8)
      CALL DUAL
      RETURN
```

```
  101 FORMAT(1H1,8X,'BLACK SIDE',//,8(1X,8I3,/),//,8X,'WHITE SIDE')
      END
      INTEGER FUNCTION DIST(W,X,Y,Z)
      IMPLICIT INTEGER(A-Z)
      A=IABS(W-Y)
      B=IABS(X-Z)
      DIST=A
      IF (B.GT.A) DIST=B
      RETURN
      END
      SUBROUTINE ONEPAN
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /SKORE/ MATERW,MATERB
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /PLISTZ/ QC,KXO,KYO,KX,KY,PX,PY,PX1,PY1,
     1KX1,KY1,KX2,KY2
C
C     SUBROUTINE ONEPAN IS CALLED FROM TREE AND, WHEN GIVEN A
C     POSITION WITH ONLY ONE PAWN, IT SCORES IT AS A TERMINAL NODE.
C     TO DETERMINE WHETHER THE POSITION IS WON OR DRAWN ONEPAN USES
C     THE 'RULE OF THE SQUARE' TEST AND/OR CALLS SUBROUTINES RKPAWN
C     OR THPAWN, WHICHEVER IS APPROPRIATE. ONEPAN WILL ONLY TEST
C     POSITIONS WITH A WHITE PAWN, SO THE GIVEN POSITION MUST BE
C     'SET UP' ACCORDINGLY. THE SCORING FUNCTION MUST ALSO BE
C     ADJUSTED TO TAKE THIS FACT (IE. THE ORIGINAL MATERIAL
C     SITUATION) INTO CONSIDERATION. IN ONEPAN THE X-Y
C     COORDINATES OF THE BLACK KING ARE REPRESENTED BY KXO,KYO.
C     AS WELL THE WHITE PAWN IS REPRESENTED BY PX,PY AND THE
C     WHITE KING BY KX,KY. SINCE THESE VALUES ARE SOMETIMES
C     CHANGED IN RKPAWN OR THPAWN, A COPY OF THEM HAS BEEN
C     PRESERVED IN KX2,KY2,PX1,PY1 AND KX1,KY1 RESPECTIVELY.
C     QC REFERS TO THE COLOUR OF THE PLAYER TO MOVE. THE VARIABLE
C     ADD IN THE SCORING FUNCTION DISTINGUISHES BETWEEN POSITIONS
C     ON A MINOR SCALE (IE. BASED ON THE PROXIMITY OF EITHER
C     KING TO THE WHITE PAWN). THIS ENABLES THE TREE SEARCH TO SELECT
C     THE 'BEST' OF SEVERAL EQUALLY 'LIKELY' MOVES.
C
C TEST FOR ONE PAWN AND WHICH SIDE HAS PAWN
C
      IF ((MATERW.EQ.1).AND.(MATERB.EQ.0)) GOTO 10
      IF ((MATERB.EQ.-1).AND.(MATERW.EQ.0)) GOTO 20
      RETURN
C
C SETS UP THE PAWN'S AND BOTH KING'S POSITIONS FOR SUBROUTINES
C
10    QC=C
      J=2
11    IF (WHITEP(J).EQ.1) GOTO 12
      J=J+1
      GOTO 11
12    PX=WHITEX(J)
      PY=WHITEY(J)
      KX=WHITEX(1)
      KY=WHITEY(1)
```

```
      KXO=BLACKX(1)
      KYO=BLACKY(1)
      GO TO 50
C
C     IF BLACK HAS THE PAWN, THE POSITION IS REVERSED. THE PROGRAM
C     WILL ONLY TEST A POSITION WITH A WHITE PAWN.
C
20    QC=-C
      J=2
21    IF (BLACKP(J).EQ.-1) GOTO 22
      J=J+1
      GOTO 21
22    PX=9-BLACKX(J)
      PY=9-BLACKY(J)
      KX=9-BLACKX(1)
      KY=9-BLACKY(1)
      KXO=9-WHITEX(1)
      KYO=9-WHITEY(1)
50    DRAWC3=0
      PX1=PX
      PY1=PY
      KX1=KX
      KY1=KY
      KX2=KXO
      KY2=KYO
C
C BEGINS 'RULE OF THE SQUARE' TEST AND CONSIDERS
C EXCEPTIONS EG. ROOK PAWN STALEMATE.
C
      IF ((PY.EQ.2).AND.(QC.EQ.-1).AND.(DIST(PX,PY,KXO,KYO).EQ.1)
     1.AND.(DIST(PX,PY,KX,KY).NE.1)) GOTO 888
      IF ((PY.EQ.7).AND.(MOD(PX,7).EQ.1).AND.(KX.EQ.PX).AND.
     1(KY.EQ.8).AND.(((KYO.GE.7).AND.(IABS(KX-KXO).EQ.2)).OR.
     1((KYO.GE.6).AND.(IABS(KX-KXO).EQ.3).AND.(QC.EQ.-1)).OR.
     1((IABS(KX-KXO).EQ.1).AND.(KYO.EQ.6).AND.(QC.EQ.-1))))
     1GOTO 888
      DD=8-PY
      IF (PY.EQ.2) DD=5
      IF (QC.EQ.-1) DD=DD+1
      IF (IABS(KXO-PX).GT.DD) GOTO 777
      IF ((8-KYO).GT.DD) GOTO 777
C
C TESTS WHICH SUBRCUTINE SHOULD BE APPLIED
C
      IF ((KYO.EQ.PY-1).OR. (MOD(PX,7).EQ.1)) CALL RKPAWN
      IF ((KY2.NE.PY1-1).AND.(MOD(PX1,7).NE.1)) CALL THPAWN
      RETURN
C
C SCORES LOST POSITIONS
C
777   ADD=7+DIST(PX1,PY1,KX2,KY2)-DIST(PX1,8,KX1,KY1)
      IF (MATERW.NE.1)  GOTO 801
      SCORE(PLY+3)=6000-200*PLY-400*(8-PY1)-100*(1-C)+ADD
      DRAWC3=1
      RETURN
801   SCORE(PLY+3)=-6000+200*PLY+400*(8-PY1)+100*(1+C)-ADD
      DRAWC3=1
      RETURN
```

```
C
C SCORES DRAWN POSITIONS
C
888    ADD=7+DIST(PX1,PY1,KX2,KY2)-DIST(PX1,8,KX1,KY1)
       IF (MATERW.NE.1) GOTO 889
       SCORE(PLY+3)=ADD
       DRAWC3=1
       RETURN
889    SCORE(PLY+3)=-ADD
       DRAWC3=1
       RETURN
       END
       SUBROUTINE RKPAWN
       IMPLICIT INTEGER (A-Z)
       COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
       COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
       COMMON /SKORE/ MATERW,MATERB
       COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
       COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
       COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
       COMMON /PLISTZ/ QC,KXC,KYC,KX,KY,PX,PY,PX1,PY1,
      1KX1,KY1,KX2,KY2
C
C      SUBROUTINE RKPAWN IS CALLED FROM ONEPAN WHEN A) THE
C      BLACK KING IS ONE SQUARE BEHIND THE WHITE PAWN OR
C      B) THE PAWN IS A ROOK PAWN. THE PROGRAM TESTS,IN A
C      SEQUENTIAL MANNER,WHAT ARE THE POSITIONS OF THE BLACK KING
C      AND WHITE PAWN RELATIVE TO EACH OTHER . (IE. FIRST TEST FOR
C      WIDTH DISTANCE BETWEEN BLACK KING AND WHITE PAWN EG. 3
C      FILES ADJACENT. SECOND TEST FOR HEIGHT DISTANCE BETWEEN
C      THEM EG. D=4. NOTE: 0 FILES ADJACENT,D=0: BOTH PIECES
C      OCCUPY THE SAME SQUARE.)  IN EACH PARTICULAR CASE IT
C      THEN TESTS WHETHER THE WHITE KING OCCUPIES A SQUARE WITHIN
C      A SPECIFIED REGION AND, DEPENDING ON THE RESULT, SCORES THE
C      POSITION. THIS IS A CASE BY CASE APPROACH. IN (A) WE CONSIDER
C      PAWNS ON ANY FILE WITH D=-1 BUT IN (B) WE CONSIDER ONLY
C      ROOK PAWNS WITH D VARYING FROM ZERO TO SIX.
C      SUBROUTINE THPAWN IS CALLED FROM ONEPAN WHEN THE WHITE
C      PAWN IS NOT ON THE ROOK FILE AND D VARIES 0-6.
C      ITS STRUCTURE IS SIMILAR TO RKPAWN. IN THIS WAY
C      WE CONSIDER ALL CASES. THIS, TOGETHER WITH THE
C      'RULE OF THE SQUARE' TEST AND EXCEPTIONS, HANDLED IN ONEPAN,
C      WILL SCORE ANY GIVEN POSITION WITH ONE PAWN.
C
       IF (KYO.NE.PY-1)    GOTO 100
C
C
C
C       D=-1    BEGINS
C
C
C
C      THE BLACK KING IS ON THE SAME FILE OR ONE FILE ADJACENT TO
C      THE WHITE PAWN
C
       IF (IABS(KXO-PX).GT.1) GOTO 5
       IF (IABS(KX-PX).GT.1) GOTO 888
       IF (IABS(KY-PY).GT.1) GOTO 888
```

```
      GOTO 777
C
C     2 FILES ADJACENT:D=0
C
5     IF (IABS(PX-KXO).NE.2) GOTO 20
      IF (IABS(KX-PX).GT.2) GOTO 888
      IF (MOD(PX,7).NE.1) GOTO 15
      IF (PY.EQ.7) GOTO 10
      IF ((PY.EQ.6).AND.(KY.EQ.8).AND.(KX.EQ.PX)) GOTO 888
      IF (KY.LE.PY) GOTO 888
      IF (KY.GT.PY+2) GOTO 888
      IF ((KX.EQ.KXO).AND.(KY.EQ.PY+2)) GOTO 888
      GOTO 777
10    IF (KY.LT.6) GOTO 888
      IF ((KY.EQ.8).AND.(KX.EQ.PX)) GOTO 888
      GOTO 777
15     IF (KY.LT.KYC) GOTO 888
      IF (KY.GT.PY+2) GOTO 888
      IF ((KY.EQ.PY+2).AND.(KX.EQ.KXO)) GOTO 888
      GOTO 777
C
C     3 FILES ADJACENT
C
20    IF (IABS(KXO-PX).NE.3) GOTO 35
      IF (IABS(KX-PX).GT.3) GOTO 888
      IF (KY.GT.PY+3) GOTO 888
      IF ((KX.EQ.KXO).AND.(KY.GT.PY+1)) GOTO 888
      IF (MOD(PX,7).NE.1) GOTO 30
      IF (KY.LT.KYC) GOTO 888
      IF ((KX.EQ.PX).AND.(KY.EQ.KYO)) GOTO 888
      GOTO 777
30    IF (KY.LT.KYO-1) GOTO 888
      GOTO 777
C
C     4 FILES ADJACENT
C
35    IF (IABS(KXO-PX).NE.4) GOTO 45
      IF (KY.GT.PY+4) GOTO 888
      IF (IABS(KX-PX).GT.4) GOTO 888
38    IF ((KX.EQ.KXO).AND.(KY.GT.PY+1)) GOTO 888
      IF ((IABS(KX-KXO).EQ.1).AND.(KY.GT.PY+3)) GOTO 888
      IF (MOD(PX,7).NE.1) GOTO 40
      IF (KY.LT.KYO) GOTO 888
      GOTO 777
40    IF (KY.LT.KYO-1) GOTO 888
      IF ((KY.EQ.KYO-1).AND.(IABS(KX-KXO).LE.2))  GOTO 888
      GOTO 777
C
C     5 FILES ADJACENT
C
45    IF (IABS(KXO-PX).NE.5) GOTO 60
      IF (IABS(KX-PX).GT.5) GOTO 888
      IF (MOD(PX,7).EQ.1) GOTO 38
      KYO=0
      IF (PY.EQ.4) KYO=2
      GOTO 38
C
C     6 FILES ADJACENT
```

```
C
60    IF (MOD(PX,7).NE.1) GOTC 65
      IF (IABS(KX-PX).GT.6) GOTO 888
      GOTO 38
65    IF ((KX.EQ.KXO).AND.(KY.GT.PY+1)) GOTO 888
      IF ((KY.GT.6).AND.(IABS(KX-KXO).EQ.1)) GOTO 888
      GOTO 777
C
C
C
C     ROOK PAWN BEGINS
C
C
C
C     0 OR 1 FILES ADJACENT
C
100    QD=0
      IF (KXO.EQ.PX)  GOTO 888
      IF (IABS(KXO-PX).NE.1) GOTO 110
      IF (KYO.NE.PY) GOTO 888
      IF (QC.EQ.-1) GOTO 888
      IF (PY.EQ.6) GOTO 105
103   IF (KY.NE.PY+2) GOTO 888
      IF (IABS(KX-PX).GT.1) GOTO 888
      GOTO 777
105   IF (KY.NE.PY+2) GOTO 888
      IF (IABS(KX-PX).NE.1) GOTO 888
      GOTO 777
C
C     2 FILES ADJACENT
C
110   IF (IABS (KXO-PX).NE.2) GOTO 160
      IF (KYO.NE.PY) GOTO 125
      IF (PY.EQ.7) GOTO 120
      IF (PY.EQ.2) GOTO 115
      IF (QC.EQ.-1) GOTO 105
113   IF (KY.GT.PY+3) GOTO 888
114   IF (KY.LT.KYO) GOTO 888
      IF (IABS(KX-PX).GT.2+QD) GOTO 888
      GOTO 777
115   IF (KY.GT.PY+4) GOTO 888
      IF (KY.LT.KYC+2) GOTO 888
      IF (IABS(KX-PX).GT.3) GOTO 888
      GOTO 777
120   IF (KY.NE.6) GOTO 888
      IF (IABS(KX-PX).GT.2) GOTO 888
      GOTO 777
C
C     D=1
C
125   IF (KYO.NE.PY+1) GOTO 145
      IF (PY.EQ.7) GOTO 120
      IF (PY.NE.2) GOTO 135
      IF (QC.EQ.1) GOTO 130
      IF (KY.NE.PY+3) GOTO 888
      IF (KX.EQ.PX) GOTO 888
      IF (IABS(KX-PX).GT.2) GOTO 888
      GOTO 777
```

```
130   KYO=KYO-1
      GOTO 115
135   IF (QC.EQ.-1) GOTO 888
      PY=PY+1
      GOTO 105
C
C     D=2 OR GREATER
C
145   IF (KYO.NE.PY+2) GOTO 888
      IF (PY.EQ.6) GOTO 155
      IF (PY.NE.2) GOTO 888
      IF (QC.EQ.-1) GOTO 888
      PY=PY+2
      GOTO 105
155   IF (QC.EQ.-1) GOTO 888
      GOTO 120
C
C     3 FILES ADJACENT:D=0
C
160   IF (IABS(KXO-PX).NE.3) GOTO 250
      IF (KYO.NE.PY) GOTO 170
      IF (PY.NE.2) GOTO 165
      IF (KY.GT.PY+4) GOTO 888
      IF ((KX.EQ.PX).AND.(KY.EQ.PY+1)) GOTO 888
      GOTO 392
165   IF (QC.EQ.1) GOTO 167
166   IF ((IABS(KX-PX).EQ.2).AND.(KY.EQ.KYO+2)) GOTO 777
      GOTO 172
167   IF ((KX.EQ.KXO).AND.(KY.GT.KYO).AND.(KY.LT.KYO+4)) GOTO 777
      PY=PY+1
      KYO=KYO-1
      GOTO 113
C
C     D=1
C
170   IF (KYO.NE.PY+1) GOTO 190
      IF (PY.EQ.6) GOTO 185
      IF (PY.EQ.2) GOTO 180
      IF (QC.EQ.1) GOTO 175
172   IF (KY.GT.PY+3) GOTO 888
173   IF (IABS(KX-PX).NE.1) GOTO 888
      IF (KY.LT.KYO) GOTO 888
      GOTO 777
175   KYO=KYO-1
      PY=PY+1
      GOTO 113
180   IF (QC.EQ.1) GOTO 183
      IF ((KX.EQ.KXO).AND.(KY.EQ.KYO+2)) GOTO 777
      PY=PY+1
      GOTO 166
183   IF ((KY.GT.KYO).AND.(KY.LT.KYO+4).AND.
     1(IABS(KX-KXO).IE.1)) GOTO 777
      PY=PY+2
      KYO=KYO-1
      GOTO 113
185   KYO=KYO-1
      GOTO 173
C
```

```
C     D=2
C
190   IF (KYO.NE.PY+2) GOTO 215
      IF (PY.EQ.2) GOTO 205
      IF (PY.EQ.6) GOTO 200
      IF (QC.EQ.1) GOTO 175
      GOTO 172
200   IF ((KY.EQ.6).AND.(IABS(KX-PX).EQ.2)) GOTO 777
      KYO=KYO-2
      GOTO 173
205   PY=PY+1
      IF (QC.EQ.1) GOTO 175
      GOTO 172
C
C     D=3 OR GREATER
C
215   IF  (KYO.NE.PY+3) GOTO 245
      IF (PY.EQ.2) GOTO 235
217   IF (PY.EQ.5) KYC=KYO-1
      IF (QC.EQ.1) GOTO 175
      GOTO 172
235   IF (QC.EQ.1) GOTO 240
      PY=PY+1
      GOTO 172
240   PY=PY+2
      KYO=KYO-1
      GOTO 113
245   IF (KYO.EQ.8) PY=5
      IF (KYO.EQ.PY+4) PY=PY+1
      IF (KYO.EQ.PY+5) PY=PY+2
      GOTO 217
C
C     4 FILES ADJACENT:D=0
C
250   IF (IABS(KXO-PX).NE.4) GOTO 370
      IF (KYO.NE.PY) GOTO 265
      IF (PY.EQ.2) GOTO 260
      IF (QC.EQ.1) GOTO 255
253   IF ((IABS(KX-PX).EQ.3).AND.(KY.EQ.KYO+2)) GOTO 777
      PY=PY+1
      GOTO 113
255   QD=1
      PY=9
      GOTO 167
260   IF (MOD(KY,7).EQ.1) GOTO 888
      IF (IABS(KX-PX).GT.4) GOTO 888
      IF ((IABS(KX-PX).GT.2).AND.(KY.GT.6)) GOTO 888
      GOTO 777
C
C     D=1
C
265   IF (KYO.NE.PY+1) GOTO 285
      IF (PY.EQ.2) GOTO 275
      IF (QC.EQ.1) GOTO 270
      PY=PY+1
      GOTO 113
270   KYO=KYO-1
      QD=1
```

```
      GOTO 114
275   IF (QC.EQ.1) GOTO 280
      IF ((KX.EQ.KXO).AND.(KY.EQ.KYO+2)) GOTO 777
      PY=PY+1
      GOTO 253
280   QD=1
      PY=9
      GOTO 183
C
C     D=2
C
285   IF (KYO.NE.PY+2) GOTO 315
      IF (PY.EQ.2) GOTO 305
      IF (PY.EQ.5) GOTO 300
      IF (QC.EQ.1) GOTO 270
      PY=PY+1
      GOTO 113
300   KYO=KYO-1
      GOTO 114
305   IF (QC.EQ.1) GOTO 270
      PY=PY+2
      GOTO 113
C
C     D=3
C
315   IF (KYO.NE.PY+3)  GOTO 345
      IF (PY.EQ.5) GOTO 325
      IF (PY.EQ.4) GOTO 330
      IF (PY.EQ.2) GOTO 340
      IF (QC.EQ.1) GOTO 270
      PY=PY+1
      GOTO 113
325   KYO=KYO-2
      GOTO 114
330   KYO=KYO-1
      IF (QC.EQ.1) GOTO 270
      GOTO 114
340   IF (QC.EQ.1) GOTO 270
      PY=PY+2
      GOTO 113
C
C     D=4 OR GREATER
C
345   IF (KYO.EQ.8) GOTO 365
      IF (KYO.EQ.7) KYO=KYO-1
      IF (QC.EQ.1) GOTO 270
      IF (KY.EQ.8) GOTO 888
      GOTO 114
365   KYO=KYO-2
      IF (QC.EQ.1) GOTO 270
      GOTO 114
C
C     5 FILES ADJACENT:D=0
C
370   IF (IABS(KXO-PX).NE.5) GOTO 430
      IF (KYO.NE.PY) GOTO 390
      IF (PY.EQ.2) GOTO 380
      IF (QC.EQ.1) GOTO 375
```

```
371   QD=1
372   IF ((IABS(KX-PX).EQ.4) .AND.(KY.EQ.KYO+2)) GOTO 777
      GOTO 114
375   QD=2
      PY=9
      GOTO 167
380   QD=3
385   IF ((KY.GT.6).AND.(IABS(KX-PX).GT.3)) GOTO 888
      GOTO 114
C
C     D=1
C
390   IF (KYO.NE.PY+1) GOTO 405
      IF (PY.EQ.2) GOTO 401
      IF (QC.EQ.1) GOTO 395
392   QD=1
      GOTO 114
395   IF (IABS(KX-PX).GT.4) GOTO 888
      IF (KY.LT.KYO-1) GOTO 888
      GOTO 777
401   IF (QC.EQ.1) GOTO 435
      IF ((KX.EQ.KXO).AND.(KY.EQ.KYO+2)) GOTO 777
      GOTO 371
C
C     D=2
C
405   IF (KYO.NE.PY+2) GOTO 425
      IF (PY.EQ.4) KYC=KYO-1
      IF (QC.EQ.1) GOTO 395
      QD=1
      GOTO 114
C
C     D=3 OR GREATER
C
425   KYO=5
      IF (QC.EQ.1) GOTO 395
      QD=1
      GOTO 114
C
C     6 FILES ADJACENT:D=0
C
430   IF (KYO.NE.PY) GOTO  440
      IF (PY.EQ.2) GOTO 435
433   QD=2
      IF ((IABS(KX-PX).EQ.5).AND.(KY.EQ.KYO+2)) GOTO 777
      GOTO 114
435   QD=4
      IF ((KY.GT.6).AND.(IABS(KX-PX).GT.4)) GOTO 888
      KYO=2
      GOTO 114
C
C     D=1
C
440   IF (KYO.NE.PY+1) GOTO 460
      IF (PY.EQ.2) GOTO 450
445   QD=2
      GOTO 114
450   IF ((KX.EQ.KXO).AND.(KY.EQ.KYO+2)) GOTO 777
```

```
      GOTO 433
C
C     D=2 OR GREATER
C
460   KYO=4
      QD=2
      GOTO 114
C
C
C
C
C
C
777   ADD=7+DIST(PX1,PY1,KX2,KY2)-DIST(PX1,8,KX1,KY1)
      IF(MATERW.NE.1)  GOTO 801
      SCORE(PLY+3)=6000-200*PLY-400*(8-PY1)-100*(1-C)+ADD
      DRAWC3=1
      RETURN
801   SCORE(PLY+3)=-6000+200*PLY+400*(8-PY1)+100*(1+C)-ADD
      DRAWC3=1
      RETURN
888   ADD=7+DIST(PX1,PY1,KX2,KY2)-DIST(PX1,8,KX1,KY1)
      IF (MATERW.NE.1) GOTO 889
      SCORE(PLY+3)=ADD
      DRAWC3=1
      RETURN
889   SCORE(PLY+3)=-ADD
      DRAWC3=1
      RETURN
      END
      SUBROUTINE THPAWN
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /SKORE/ MATERW,MATERB
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /PLISTZ/ QC,KXC,KYC,KX,KY,PX,PY,PX1,PY1,
     1KX1,KY1,KX2,KY2
C
C     0 FILES ADJACENT
C
      IF (KXO.NE.PX) GOTO 300
      IF (KYO.NE.PY+1) GOTO 110
      IF (KYO.NE.8) GOTO 888
      IF (QC.EQ.1) GOTO 105
      IF (KY.NE.6) GOTO 888
      IF (IABS(KX-PX).NE.1) GOTO 888
      GOTO 777
105   IF ((KY.LT.5).OR.(KY.EQ.8)) GOTO 888
      IF (IABS(KX-PX).GT.2) GOTO 888
      IF ((KY.EQ.6).AND.(IABS(KX-PX).EQ.1)) GOTO 888
      GOTO 777
110   IF (KYO.NE.PY+2) GOTO 120
      IF (KYO.NE.8) GOTO 888
      IF (QC.NE.1) GOTO 888
      IF (KY.NE.PY) GOTO 888
```

```
      IF (IABS(KX-PX).NE.1) GOTO 888
      GOTO 777
120   IF (KYO.NE.PY+3)  GOTO 130
      IF (KYO.NE.8)      GOTO 125
      IF (QC.EQ.1) GOTO 123
      IF (KY.NE.6) GOTO 888
      IF (IABS(KX-PX).GT.1) GOTO 888
      GOTO 777
123   IF ((KY.LT.5).OR.(KY.EQ.8)) GOTO 888
      IF (IABS(KX-PX).GT.2)   GOTO 888
      GOTO 777
125   IF (QC.EQ.1)   GOTO 127
      IF (KY.NE.PY+1)  GOTO 888
      IF (KX.NE.PX) GOTO 888
      GOTO 777
127   IF (IABS(KX-PX).NE.1)  GOTO 888
      IF ((KY.LT.PY).OR.(KY.GT.PY+1)) GOTO 888
      GOTO 777
130   QD=KYO-PY
      IF (QC.EQ.1)  QD=QD+1
      IF (KY.LT.PY+6-QD)   GOTO 888
      IF (KY.GT.PY+QD-2)   GOTO 888
      IF (IABS(KX-PX).GT.QD-3) GOTO 888
      GOTO 777
C
C     1 FILE ADJACENT
C
300   IF (IABS(KXO-PX).NE.1) GOTO 410
310   IF (KYO.NE.PY)  GOTO 316
      IF (QC.EQ.1)  GOTO 315
      IF ((KY.LT.PY).OR.(KY.GT.PY+1)) GOTO 888
      IF (IABS(KX-PX).NE.1) GOTO 888
      GOTO 777
315   IF (IABS(KX-PX).GT.2) GOTO 888
      IF ((KY.LT.PY-1).OR.(KY.GT.PY+2)) GOTO 888
      IF ((IABS(KX-PX).EQ.2).AND.(IABS(KXO-KX).EQ.1)) GOTO 888
      GOTO 777
316   IF (KYO.NE.PY+1) GOTO 320
      IF (PY.EQ.6) GOTO 317
      IF (PY.EQ.7) GOTO 318
      IF (KY.NE.KYO) GOTO 888
      IF (IABS(KX-PX).NE.1) GOTO 888
      GOTO 777
317   IF (MOD(KY,8).LT.6) GOTO 888
      IF (IABS(KX-PX).NE.1) GOTO 888
      IF ((MOD(KXO,7).EQ.1).AND.(KY.EQ.6)) GOTO 888
      GOTO 777
318   IF ((KY.EQ.6).AND.(KX.EQ.PX)) GOTO 777
      IF (KY.LE.6) GOTO 888
      IF (IABS(KX-PX).NE.1) GOTO 888
      GOTO 777
320   IF (KYO.NE.PY+2) GOTO 330
      IF (KYO.NE.8)  GOTO 325
      IF ((MOD(PX,5).EQ.2).AND.(MOD(KXO,7).EQ.1)) GOTO 323
      IF (QC.EQ.1)  GOTO 322
      IF ((KY.GE.6).AND.(IABS(KX-PX).LE.1)) GOTO 777
      GOTO 888
322   IF ((KY.LT.5).OR.(IABS(KX-PX).GT.2)) GOTO 888
```

```
      IF ((KX.EQ.KXO).AND.(KY.EQ.PY)) GOTO 888
      GOTO 777
323   IF (QC.EQ.1) GOTO 324
      IF ((KY.EQ.6).AND.(IABS(KX-PX).EQ.1)) GOTO 777
      GOTO 888
324   IF ((KY.EQ.6).AND.(IABS(KX-PX).EQ.1)) GOTO 888
      IF ((KY.EQ.8).AND.(IABS(KX-PX).GT.1)) GOTO 888
      GOTO 322
325   IF (QC.EQ.1) GOTO 327
      IF ((KY.LE.PY).OR.(KY.GT.PY+2)) GOTO 888
      IF (IABS(KX-PX).NE.1) GOTO 888
      GOTO 777
327   IF ((KY.LT.PY).OR.(KY.GT.PY+3)) GOTO 888
      IF ((IABS(KX-PX).GT.2).OR.(IABS(KX-KXO).LT.2)) GOTO 888
      GOTO 777
330   IF (KYO.NE.PY+3)  GOTO 340
      IF (KYO.NE.8)  GOTO 335
      IF (QC.EQ.1)  GOTO 332
      IF ((KY.LT.6).OR.(IABS(KX-PX).GT.1)) GOTO 888
      GOTO 777
332   IF ((KY.LT.5).OR.(IABS(KX-PX).GT.2)) GOTO 888
      GOTO 777
335   IF (QC.EQ.1)  GOTO 337
      IF (IABS(KX-PX).NE.1)  GOTO 888
      IF ((KY.LE.PY).OR.(KY.GT.KYO)) GOTO 888
      GOTO 777
337   IF (IABS(KX-PX).GT.2)  GOTO 888
      IF ((KY.LT.PY).OR.(KY.GT.KYO+1)) GOTO 888
      IF ((KX.EQ.KXO).AND.(KY.EQ.KYO-2)) GOTO 888
      GOTO 777
340   IF (KYO.NE.PY+4)  GOTO 350
      IF (QC.EQ.1) GOTO 345
      IF ((IABS(KX-KXO).EQ.2).AND.(IABS(KX-PX).EQ.1).AND.
     1(KY.GT.PY).AND.(KY.LE.KYO)) GOTO 777
      IF ((KY.EQ.PY+2).AND.(IABS(KX-PX).LT.2)) GOTO 777
      GOTO 888
345   IF ((IABS(KX-KXO).LE.1).AND. (KY.EQ.PY)) GOTO 888
      IF (IABS(KX-PX).GT.2) GOTO 888
      IF (KY.LT.PY) GOTO 888
      IF (KY.GT.KYO+1) GOTO 888
      GOTO 777
350   IF (KYO.NE.PY+5) GOTO 360
      IF (QC.EQ.1)  GOTO 355
      IF ((KY.LE.PY).OR.(KY.GT.KYO)) GOTO 888
      IF (IABS(KX-PX).GT.2)  GOTO 888
      IF ((IABS(KX-PX).EQ.2).AND.(KY.GT.PY+3)) GOTO 888
      GOTO 777
355   IF (IABS(KX-PX).GT.3)  GOTO 888
      IF (KY.LT.PY) GOTO 888
      IF (( IABS(KX-PX).EQ.3).AND.(KY.GE.KYO)) GOTO 888
      GOTO 777
360   IF (QC.EQ.1) GOTO 365
      IF (IABS(KX-PX).GT.3) GOTO 888
      IF (KY.EQ.1)  GOTO 888
      IF ((IABS(KX-PX).GT.1).AND.(KY.GE.7)) GOTO 888
      GOTO 777
365   IF (IABS(KX-PX).GT.4) GOTO 888
      IF ((KY.EQ.8).AND.(IABS(KX-PX).GT.2)) GOTO 888
```

```
      GOTO 777
C
C     2 FILES ADJACENT
C
410   QD=0
      QK=0
      IF (IABS(KXO-PX).NE.2)  GOTO 525
      IF (KYO.NE.PY) GOTO 425
      IF (PY.EQ.2) GOTO 420
      IF (QC.EQ.1) GOTO 415
      IF (KX.EQ.KXO) GOTO 888
413   IF (KY.GT.PY+2) GOTO 888
414   IF (KY.LT.KYO-1+QK) GOTO 888
      IF (IABS(KX-PX).GT.2+QD) GOTO 888
      GOTO 777
415   IF ((IABS(KX-PX).EQ.3).AND.(IABS(KX-KXO).EQ.1)) GOTO 888
      IF ((KX.EQ.KXO).AND.(KY.EQ.KYO-2)) GOTO 888
417   QK=-1
      PY=PY+1
      QD=1
      GOTO 413
420   PY=3
      GOTO 417
425   IF (KYO.NE.PY+1) GOTO 460
      IF (PY.EQ.6) GOTO 450
      IF (PY.EQ.2) GOTO 440
      IF (PY.EQ.7) GOTO 430
      IF (QC.EQ.-1) GOTO 413
      IF ((IABS(KX-PX).EQ.1).AND.(KY.EQ.PY+3)) GOTO 777
      IF (IABS(KX-KXO).LE.1) GOTO 888
      GOTO 417
430   IF ((MOD(PX,3).NE.0).OR.(MOD(KXO,7).NE.1)) GOTO 435
      IF (IABS(KX-KXO).LE.1) GOTO 888
435   KYO=7
      GOTO 414
440   IF (QC.EQ.1) GOTO 445
      PY=3
      GOTO 413
445   IF ((KY.EQ.1).AND.(IABS(KX-KXO).LE.1)) GOTO 888
      PY=4
      KYO=1
      QD=1
      GOTO 413
450   IF (QC.EQ.1) GOTO 455
      KYO=6
      GOTO 414
455   KYO=5
      QD=1
      GOTO 414
460   IF (KYO.NE.PY+2) GOTO 480
      IF (PY.EQ.6) GOTO 470
      IF (QC.EQ.1) GOTO 465
      IF (IABS(KX-KXO).LE.1) GOTO 888
      PY=PY+1
      QK=-1
      GOTO 413
465   IF ((IABS(KY-KYO).EQ.2).AND.(IABS(KX-PX).EQ.1)) GOTO 777
      IF (IABS(KX-KXO).LE.1) GOTO 888
```

```
      PY=PY+2
      QK=-2
      QD=1
      GOTO 413
470   IF (QC.EQ.1) GOTO 475
      IF ((KY.EQ.6).AND.(IABS(KX-KXO).EQ.1)) GOTO 888
      KYO=6
      GOTO 414
475   KYO=5
      QD=1
      GOTO 414
480   IF (KYO.NE.PY+3) GOTO 495
      IF (PY.EQ.5) GOTO 490
      IF (QC.EQ.1) GOTO 485
482   IF ((KX.EQ.KXO).AND.(KY.EQ.KYO-2)) GOTO 777
      IF (IABS(KX-KXO).LE.1) GOTO 888
      PY=PY+1
      QK=-2
      GOTO 413
485   IF ((KX.EQ.KXO).AND.(KY.EQ.KYO-2)) GOTO 888
486   IF ((IABS(KX-KXO).LE.1).AND.(KY.EQ.PY-1)) GOTO 888
      PY=PY+2
      QK=-3
      QD=1
      GOTO 413
490   IF (QC.EQ.1) GOTO 486
      IF ((KY.EQ.5).AND.(IABS(KX-KXO).LE.1)) GOTO 888
      KYO=6
      GOTO 414
495   IF (KYO.NE.PY+4) GOTO 505
      IF (QC.EQ.1) GOTO 500
      IF ((IABS(KX-KXO).LE.1).AND.(KY.LT.KYO-2)) GOTO 888
497   PY=PY+2
      QK=-3
      GOTO 413
500   IF ((IABS(KX-PX).EQ.1).AND.(KY.EQ.PY)) GOTO 777
      IF ((IABS(KX-KXO).LE.1).AND.(KY.LE.PY)) GOTO 888
      PY=PY+3
      QK=-4
      QD=1
      GOTO 413
505   IF (KYO.NE.PY+5) GOTO 515
      IF (QC.EQ.1) GOTO 510
      IF ((IABS(KX-KXO).LE.1).AND.(KY.EQ.PY)) GOTO 888
      PY=PY+3
      QK=-4
      GOTO 413
510   IF ((IABS(KX-KXO).LE.1).AND.(KY.LT.PY)) GOTO 888
      QK=-5
      QD=1
      GOTO 414
515   IF (QC.EQ.1) GOTO 520
      IF ((IABS(KX-KXO).EQ.5).AND.(KY.GE.7)) GOTO 888
      KYO=3
      QD=1
      GOTO 414
520   IF (IABS(KX-PX).GT.4) GOTO 888
      IF ((KY.EQ.8).AND.(IABS(KX-PX).EQ.4)) GOTO 888
```

```
      GOTO 777
C
C
C
C     3 FILES ADJACENT
C
C
C
525   IF (IABS(KXO-PX).NE.3) GOTO 660
      IF (KYO.NE.PY) GOTO 545
      IF (PY.EQ.6) GOTO 540
      IF (PY.EQ.2) GOTO 535
      IF (QC.EQ.1) GOTO 530
526   IF ((KY.EQ.PY+2).AND.(IABS(KX-PX).EQ.2)) GOTO 777
527   IF ((KY.EQ.PY-1).AND.(IABS(KX-KXO).EQ.2)) GOTO 888
528   IF (IABS(KX-KXO).LE.1) GOTO 888
      PY=PY+1
      QD=1
      GOTO 413
530   IF ((KY.EQ.KYO+4).AND.(KX.EQ.KXO)) GOTO 888
      IF ((IABS(KX-KXO).LE.1).AND.(KY.EQ.KYO-2)) GOTO 888
      IF ((IABS(KX-KXC).EQ.1).AND.(IABS(KX-PX).EQ.4)) GOTO 888
      QD=2
      KYO=KYO-1
      PY=PY+2
      GOTO 413
535   IF ((IABS(KX-KXO).EQ.2).AND.(KY.EQ.PY-1)) GOTO 888
      PY=4
      QD=1
      GOTO 413
540   IF ((KY.EQ.8).AND.(IABS(KX-PX).EQ.2)) GOTO 777
      IF (IABS(KX-KXO).LE.1) GOTO 888
      KYO=KYO-1
      QD=1
      GOTO 414
545   IF (KYO.NE.PY+1) GOTO 570
      IF (PY.EQ.6) GOTO 565
      IF (PY.EQ.2) GOTO 555
547   IF (QC.EQ.1) GOTO 550
      KYO=KYO-1
      GOTO 527
550   IF ((KY.EQ.PY-2).AND.(IABS(KX-KXO).EQ.1)) GOTO 888
551   IF ((IABS(KX-KXO).LE.1).AND.(IABS(KX-PX).GT.2)) GOTO 888
      KYO=KYO-2
      PY=PY+2
      QD=2
      GOTO 413
555   IF (QC.EQ.1) GOTO 560
      IF ((KY.EQ.PY-1).AND.(IABS(KX-KXO).LE.2)) GOTO 888
      IF ((KY.EQ.KYO+3).AND.(IABS(KXO-KX).LE.1)) GOTO 888
      KYO=2
      PY=4
      QD=1
      GOTO 413
560   IF ((IABS(KX-KXO).LE.1).AND.(IABS(KX-PX).GT.2).AND.
     1(MOD(KY,6).EQ.1)) GOTO 888
      IF (KY.EQ.8) GOTO 888
      IF (IABS(KX-PX).GT.4) GOTO 888
```

```
      GOTO 777
565   IF ((KY.EQ.PY-2).AND.(IABS(KX-KXO).LE.1)) GOTO 888
      KYO=5
      QD=1
      GOTO 414
570   IF (KYO.NE.PY+2) GOTO 605
      IF (PY.EQ.6) GOTO 600
      IF (PY.EQ.5) GOTO 590
      IF (PY.EQ.2) GOTO 580
      KYO=KYO-1
      GOTO 547
580   KYO=0
      PY=3
      IF (QC.EQ.1) GOTO 551
      IF ((KY.EQ.PY-2).AND.(IABS(KX-KXO).EQ.2)) GOTO 888
      GOTO 528
590   IF (QC.EQ.1) GOTO 595
592   IF ((KX.EQ.KXO).AND.(KY.EQ.PY)) GOTO 888
      IF ((IABS(KX-KXO).LE.2).AND.(KY.EQ.4)) GOTO 888
      KYO=5
      QD=1
      GOTO 414
595   IF ((IABS(KX-KXO).LE.1).AND.(KY.EQ.3)) GOTO 888
      IF ((IABS(KX-PX).EQ.4).AND.(IABS(KX-KXO).EQ.1)) GOTO 888
      KYO=4
      QD=2
      GOTO 414
600   KYO=5
      QD=1
      GOTO 414
605   IF (KYO.NE.PY+3) GOTO 625
      IF (PY.EQ.5) GOTO 615
      IF (QC.EQ.1) GOTO 610
      IF ((KX.EQ.KXO).AND.(KY.EQ.KYO-2)) GOTO 777
      IF ((KY.EQ.PY-1).AND.(IABS(KX-KXO).EQ.2)) GOTO 888
      IF (IABS(KX-KXO).LE.1) GOTO 888
      PY=PY+2
      QK=-3
      QD=1
      GOTO 413
610   IF (((KY.EQ.PY-1).OR.(KY.EQ.KYO+2)).AND.(IABS(KX-PX).EQ.2))
     1GOTO 777
      IF ((IABS(KX-KXO).EQ.1).AND.(KY.EQ.PY+1)) GOTO 777
      IF ((IABS(KX-KXO).LE.1).AND.(KY.EQ.PY)) GOTO 777
      IF (IABS(KX-KXO).LE.1) GOTO 888
      PY=PY+3
      QK=-4
      QD=2
      GOTO 413
615   IF (QC.EQ.-1) GOTO 592
      IF ((KY.GE.5).AND.(IABS(KX-KXO).EQ.1)) GOTO 777
      GOTO 595
625   IF (KYO.NE.PY+4) GOTO 635
      IF (QC.EQ.1) GOTO 630
      IF ((IABS(KX-KXO).LE.1).AND.(KY.LE.PY+1)) GOTO 888
      IF ((IABS(KX-KXO).EQ.2).AND.(KY.EQ.PY-1)) GOTO 888
      PY=PY+2
      QK=-4
```

```
      QD=1
      GOTO 413
630   IF ((IABS(KX-PX).EQ.2).AND.(KY.GE.PY-1)
     1.AND.(KY.LE.KYO)) GOTO 777
      IF ((KY.LE.PY).AND.(IABS(KX-KXO).LE.1)) GOTO 888
      PY=PY+3
      QK=-5
      QD=2
      GOTO 413
635   IF (KYO.NE.PY+5) GOTO 645
      IF (QC.EQ.1) GOTO 640
      IF ((IABS(KX-PX).EQ.1).AND.(KY.EQ.PY)) GOTO 777
      IF ((IABS(KX-KXO).LE.2).AND.(KY.LE.PY)) GOTO 888
      PY=PY+3
      QK=-5
      QD=1
      GOTO 413
640   IF ((KY.EQ.PY-1).AND.(IABS(KX-PX).EQ.2)) GOTO 777
      IF ((IABS(KX-KXO).LE.1).AND.(KY.LT.PY)) GOTO 888
      QK=-6
      QD=2
      GOTO 414
645   IF (QC.EQ.1) GOTO 650
      IF ((IABS(KX-KXO).LE.2).AND.(KY.EQ.1)) GOTO 888
      QD=1
      QK=-6
      GOTO 414
650   IF (IABS(KX-PX).GT.4) GOTO 888
      GOTO 777
C
C
C
C     4 FILES ADJACENT
C
C
C
660   IF (IABS(KXO-PX).NE.4) GOTO 770
      IF (KYO.NE.PY) GOTO 675
      IF (PY.EQ.2) GOTO 670
      IF (QC.EQ.1) GOTO 664
      IF ((IABS(KX-PX).EQ.3).AND.(KY.EQ.PY+2)) GOTO 777
662   IF ((IABS(KX-KXO).EQ.2).AND.(KY.EQ.PY-2)) GOTO 888
663   IF (IABS(KX-KXO).LE.1) GOTO 888
      QD=1
      QK=QK-1
      PY=PY+2
      GOTO 413
664   IF ((KX.EQ.KXO).AND.((KY.EQ.KYO+2).OR.(KY.EQ.KYO+3)))
     1GOTO 777
665   IF ((IABS(KX-KXO).EQ.1).AND.(KY.EQ.PY-3)) GOTO 888
      QK=-6
      QD=1
      GOTO 414
670   IF ((KY.EQ.7).AND.(IABS(KX-KXO).LE.1)) GOTO 888
      QD=2
      PY=5
      GOTO 413
675   IF (KYO.NE.PY+1) GOTO 695
```

```
      IF (PY.EQ.2) GOTO 685
      IF (QC.EQ.1) GOTO 665
      QK=-1
      GOTO 662
685   IF (QC.EQ.1) GOTO 690
      IF ((IABS(KX-PX).LE.4).AND.(KY.EQ.KYO+2)) GOTO 777
      PY=3
      GOTO 663
690   IF ((IABS(KX-KXO).LE.1).AND.(MOD(KY,7).GE.5)) GOTO 777
      IF (IABS(KX-PX).GT.3) GOTO 888
      GOTO 777
695   IF (KYO.NE.PY+2) GOTO 715
      IF (PY.EQ.2) GOTO 705
      IF (PY.EQ.5) GOTO 700
      IF (QC.EQ.1) GOTO 665
      QK=-2
      GOTO 662
700   IF ((IABS(KX-KXO).EQ.1).AND.(KY.LE.4)) GOTO 888
      IF ((IABS(KX-KXO).EQ.2).AND.(KY.EQ.3)) GOTO 888
      KYO=4
      QD=1
      GOTO 414
705   IF (QC.EQ.1) GOTO 710
      PY=3
      QK=-2
      GOTO 663
710   QD=1
      QK=-2
      GOTO 414
715   IF (KYO.NE.PY+3) GOTO 740
      IF (PY.EQ.2) GOTO 730
      IF (PY.EQ.5) GOTO 725
      IF (QC.EQ.1) GOTO 720
717   IF ((KX.EQ.KXO).AND.(KY.EQ.KYO-2)) GOTO 777
      QK=-3
      GOTO 662
720   IF ((KX.EQ.KXO).AND.(KY.EQ.KYO-2)) GOTO 888
722   IF ((IABS(KX-KXO).LE.1).AND.(KY.GE.PY).AND.(KY.LT.KYO))
     1GOTO 777
      GOTO 665
725   IF ((KX.EQ.KXO).AND.(KY.EQ.6)) GOTO 777
      GOTO 700
730   IF (QC.EQ.1)  GOTO 720
      PY=3
      IF ((IABS(KX-PX).EQ.2).AND.(KY.EQ.PY-2)) GOTO 777
      GOTO 717
740   IF (KYO.NE.PY+4) GOTO 750
      IF (QC.EQ.1) GOTO 745
      IF ((IABS(KX-PX).LE.4).AND.(KY.EQ.KYO-2)) GOTO 777
      IF ((IABS(KX-KXO).EQ.2).AND.(KY.EQ.PY-2)) GOTO 888
      QK=-4
      PY=PY+1
      GOTO 663
745   IF ((IABS(KX-KXO).LE.1).AND.(KY.GT.PY).AND.
     1(KY.LT.KYO)) GOTO 777
      GOTO 665
750   IF (KYO.NE.PY+5) GOTO 760
      IF (QC.EQ.1) GOTO 722
```

```
      IF ((KX.EQ.KXO).AND.(KY.GT.PY)) GOTO 777
      IF ((KY.LE.PY).AND.(IABS(KX-KXC).EQ.1)) GOTO 888
      IF ((IABS(KX-KXO).EQ.2).AND.(KY.EQ.PY-2)) GOTO 888
      PY=PY+3
      QK=-6
      QD=1
      GOTO 413
760   IF (QC.EQ.1) GOTO 765
      IF (IABS(KX-PX).GT.4) GOTO 888
      IF ((KY.EQ.1).AND.(IABS(KX-KXO).LE.1)) GOTO 888
      GOTO 777
765   IF (IABS(KX-PX).GT.5) GOTO 888
      GOTO 777
C
C
C
C     5 FILES ADJACENT
C
C
C
770   IF (IABS(KXO-PX).NE.5) GOTO 850
      IF (KYO.NE.PY) GOTO 790
      IF (PY.EQ.2) GOTO 785
      IF (QC.EQ.1) GOTO 780
      IF ((IABS(KX-PX).EQ.4).AND.(KY.EQ.PY+2)) GOTO 777
775   IF ((IABS(KX-KXO).EQ.2).AND.(KY.EQ.PY-3)) GOTO 888
      GOTO 805
780   IF ((KX.EQ.KXO).AND.(MOD(KY,7).LE.1)) GOTO 888
      GOTO 765
785   IF ((KY.GE.7).AND.(IABS(KX-KXO).LE.1)) GOTO 888
      GOTO 765
790   IF (KYO.NE.PY+1) GOTO 815
      IF (PY.EQ.2) GOTO 800
795   IF (QC.EQ.-1) GOTO 775
797   IF (IABS(KX-PX).GT.4) GOTO 888
      GOTO 777
800   IF (QC.EQ.1) GOTO 810
      IF ((KY.EQ.5).AND.(IABS(KX-PX).LE.5)) GOTO 777
805   IF (IABS(KX-PX).GT.3) GOTO 888
      GOTO 777
810   IF ((IABS(KX-PX).GT.4).AND.(MOD(KY,7).LE.1)) GOTO 888
      GOTO 777
815   IF (KYO.EQ.PY+2) GOTO 795
      IF (KYO.NE.PY+3) GOTO 825
      IF (QC.EQ.1) GOTO 820
      IF ((KX.EQ.KXO).AND.(KY.EQ.PY+1)) GOTO 777
      GOTO 775
820   IF ((IABS(KX-KXO).EQ.1).AND.(KY.EQ.PY+1)) GOTO 777
      IF ((IABS(KX-KXO).LE.1).AND.(KY.EQ.PY))  GOTO 777
      GOTO 797
825   IF (KYO.NE.PY+4) GOTO 835
      IF (QC.EQ.1) GOTO 830
      IF ((IABS(KX-PX).LE.5).AND.(KY.EQ.PY+2)) GOTO 777
      GOTO 775
830   IF ((IABS(KX-KXO).LE.1).AND.(KY.GT.PY).AND.
     1 (KY.LT.KYO)) GOTO 777
      GOTO 797
835   IF (KYO.NE.PY+5) GOTO 845
```

```
      IF (QC.EQ.1) GOTO 840
      IF ((IABS(KX-PX).LE.5).AND.(KY.GT.PY)) GOTO 777
      GOTO 805
840   IF ((IABS(KX-PX).GT.4).AND.(KY.LT.PY)) GOTO 888
      GOTO 777
845   IF (QC.EQ.1) GOTO 777
      IF (IABS(KX-PX).GT.5) GOTO 888
      IF ((KY.EQ.1).AND.(IABS(KX-KXO).LE.1)) GOTO 888
      GOTO 777
C
C
C
C     6 FILES ADJACENT
C
C
C
850   IF (KYO.NE.PY) GOTO 860
      IF (PY.EQ.2) GOTO 855
      IF ((IABS(KX-PX).EQ.5).AND.(KY.EQ.PY+2)) GOTO 777
853    IF (IABS(KX-KXO).LE.1) GOTO 888
       GOTO 777
855   IF (KY.LE.6) GOTO 777
      IF (IABS(KX-KXO).LE.1) GOTO 888
860   IF (KYO.NE.PY+1) GOTO 870
      IF (PY.EQ.3) GOTO 853
      IF (KY.EQ.5) GOTO 777
      GOTO 853
870   IF (KYO.EQ.PY+2) GOTO 853
875   IF (KYO.NE.PY+3) GOTO 880
      IF ((KY.EQ.PY+1).AND.(KX.EQ.KXC)) GOTO 777
      GOTO 853
880   IF (KYO.NE.PY+4) GOTO 885
      IF (KY.EQ.PY+2) GOTO 777
      GOTO 853
885   IF (KYO.NE.PY+5) GOTO 890
      IF (KY.GT.PY) GOTO 777
      GOTO 853
890   IF ((KY.EQ.1).AND.(IABS(KX-KXO).LE.1)) GOTO 888
777   ADD=7+DIST(PX1,PY1,KX2,KY2)-DIST(PX1,8,KX1,KY1)
      IF (MATERW.NE.1)  GOTO 801
      SCORE(PLY+3)=6000-200*PLY-400*(8-PY1)-100*(1-C)+ADD
      DRAWC3=1
      RETURN
801   SCORE(PLY+3)=-6000+200*PLY+400*(8-PY1)+100*(1+C)-ADD
      DRAWC3=1
      RETURN
888   ADD=7+DIST(PX1,PY1,KX2,KY2)-DIST(PX1,8,KX1,KY1)
      IF (MATERW.NE.1) GOTO 889
      SCORE(PLY+3)=ADD
      DRAWC3=1
      RETURN
889   SCORE(PLY+3)=-ADD
      DRAWC3=1
      RETURN
      END
      SUBROUTINE TWOPAN
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
```

```
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /SKORE/ MATERW,MATERB
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /PLISTZ/ QC,KXO,KYO,KX,KY,PX,PY,PX1,PY1,
     1KX1,KY1,KX2,KY2
C
C     SUBROUTINE TWOPAN IS CALLED FROM TREE AND WHEN A POSITION
C     WITH ONLY TWO PAWNS OF THE SAME COLOUR IS SUBMITTED, IT
C     TRIES TO DETERMINE IF THE POSITION IS TERMINAL. IF
C     THE POSITION IS INDEED TERMINAL OR THE SEARCH PLY
C     LEVEL HAS REACHED MAXIMUM DEPTH, IT IS SCORED
C     ACCORDINGLY. OTHERWISE THE PARAMETER DRAWC3 IS NOT
C     ALTERED AND WHEN CONTROL RETURNS TO TREE THE SEARCH
C     CONTINUES. THE SETUP AND SCORING STRUCTURE OF TWOPAN
C     IS SIMILAR TO ONEPAN;EG. TWOPAN WILL ONLY TEST
C     POSITIONS WITH TWO WHITE PAWNS.
C
C
C     TEST FOR TWO PAWNS
C
      IF ((MATERW.EQ.2).AND.(MATERB.EQ.0)) GOTO 10
      IF ((MATERW.EQ.0).AND.(MATERB.EQ.-2)) GOTO 20
      RETURN
C
C     SETS UP THE PAWNS' AND BOTH KINGS' POSITIONS
C
10    CQ=C
      J=2
      JJ=1
11    IF (WHITEP(J).EQ.1) GOTO 12
      J=J+1
      GOTO 11
12    IF (JJ.EQ.2) GOTO 15
      XP1=WHITEX(J)
      YP1=WHITEY(J)
      J=J+1
      JJ=2
      GOTO 11
C
C     THIS MAKES XP1,YP1 THE MOST ADVANCED PAWN
C
15    IF (YP1.GE.WHITEY(J)) GOTO 17
      XP2=XP1
      YP2=YP1
      XP1=WHITEX(J)
      YP1=WHITEY(J)
      GOTO 19
17    XP2=WHITEX(J)
      YP2=WHITEY(J)
19    XK=WHITEX(1)
      YK=WHITEY(1)
      XKO=BLACKX(1)
      YKO=BLACKY(1)
      GOTO 30
20    CQ=-C
      J=2
```

```
      JJ=1
21    IF (BLACKP(J).EQ.-1)  GOTO 22
      J=J+1
      GOTO 21
22    IF (JJ.EQ.2) GOTO 25
      XP1=9-BLACKX(J)
      YP1=9-BLACKY(J)
      J=J+1
      JJ=2
      GOTO 21
25    IF (YP1.GE.(9-BLACKY(J))) GOTO 27
      XP2=XP1
      YP2=YP1
      XP1=9-BLACKX(J)
      YP1=9-BLACKY(J)
      GOTO 29
27    XP2=9-BLACKX(J)
      YP2=9-BLACKY(J)
29    XK=9-BLACKX(1)
      YK=9-BLACKY(1)
      XKO=9-WHITEX(1)
      YKO=9-WHITEY(1)
30    DRAWC3=0
      RKPW=0
      ABCD=0
      YZA=1
      BD1=DIST(XKO,YKO,XP1,YP1)
      BD2=DIST(XKC,YKO,XP2,YP2)
      DP=YP1-YP2
      IF (IABS(XP1-XP2).NE.2) GOTO 40
C
C     THIS TAKES CARE OF  STALEMATE POSSIBILITIES WITH BP AND RP
C
      IF ((MOD(XP1,7).NE.1).AND.(MOD(XP2,7).NE.1)) GOTO 40
32    IF ((MOD(XP1,7).EQ.1).AND.(XK.EQ.XP1).AND.(YK.EQ.8)
     1.AND.(YP1.EQ.7).AND.(((CQ.EQ.1).AND.(XKO.EQ.XP2).AND.
     1(YKO.GE.7)).OR.((CQ.EQ.-1).AND.(YKO.GE.6).AND.
     1(IABS(XKO-XP2).LE.1)))) GOTO  888
      IF ((MOD(XP1,7).EQ.1).AND.(XK.EQ.XP1).AND.(YK.EQ.7).AND.
     1(YP1.EQ.6).AND.(((YP2.EQ.7).AND.(CQ.EQ.-1).AND.(YKO.GE.6)
     1.AND.(IABS(XKO-XP2).LE.1)).OR.((YP2.EQ.6).AND.(CQ.EQ.1).AND.
     1(XKO.EQ.XP2).AND.(YKO.EQ.7)).OR.((YP2.EQ.5).AND.(CQ.EQ.-1)
     1.AND.((IABS(XKO-XP2).EQ.1).AND.(YKO.GE.6)).OR.((XKO.EQ.XP2)
     1.AND.(YKO.EQ.8))))) GOTO 888
      IF (ABCD.EQ.1) GOTO 34
C
C     WE CHECK WHETHER THE LEAST ADVANCED PAWN IS ON THE ROOK
C     FILE AND, IF SO, SWITCH THE TWO PAWNS' COORDINATES
C     IN ORDER TO TEST THE ABOVE FORMULA ONCE AGAIN.
C
      IF  (MOD(XP2,7).NE.1) GOTO 40
34    AAA=XP1
      BBB=YP1
      XP1=XP2
      YP1=YP2
      XP2=AAA
      YP2=BBB
      IF (ABCD.EQ.1) GOTO 40
```

```
      ABCD=1
      GOTO 32
C
C     'RULE OF THE SQUARE' FOR EACH PAWN
C
40    DD=8-YP1
      IF (YP1.EQ.2) DD=5
      IF (CQ.EQ.-1) DD=DD+1
      IF (IABS(XKO-XP1).GT.DD) GOTO  777
      IF ((8-YKO).GT.DD) GOTO 777
      IF (XP1.EQ.XP2) GOTO 200
      DD=8-YP2
      IF (YP2.EQ.2) DD=5
      IF (CQ.EQ.-1) DD=DD+1
      IF (IABS(XKO-XP2).GT.DD) GOTO 777
      IF ((8-YKO).GT.DD) GOTO 777
C
C     PAWNS WIN ALONE; HERE WE DETERMINE WHETHER THE PAWNS
C     WIN EVEN WITHOUT THEIR KING'S ASSISTANCE.DP IS THE
C     DIFFERENCE IN HEIGHT BETWEEN THE PAWNS. BD1 IS THE
C     DISTANCE OF THE BLACK KING  FROM THE MOST ADVANCED PAWN.
C     BD2 IS THE DISTANCE OF THE BLACK KING FROM THE OTHER PAWN.
C
      IF (YP1.EQ.2) GOTO 777
      IF ((YP2.EQ.2).AND.((CQ.EQ.1).OR.(BD1.NE.1))) DP=DP-1
      IF (CQ.EQ.1) DP=DP-1
C
C     FIRST WE CONSIDER PAWNS ONE FILE APART
C
      IF (IABS(XP1-XP2).NE.1) GOTO 50
      IF (YP1.EQ.YP2+1) GOTO 777
      IF ((YP1.EQ.YP2).AND.(CQ.EQ.1)) GOTO  777
      IF ((YP1.EQ.YP2).AND.(BD1.NE.1).AND.(BD2.NE.1)) GOTO 777
      IF ((BD1.GE.DP).AND.(YP1.GT.YP2+1)) GOTO 777
      GOTO 100
C
C     2 FILES APART
C
50    IF (IABS(XP1-XP2).NE.2) GOTO 60
      IF (YP1.NE.YP2) GOTO 52
      IF (YP1.EQ.7) GOTO 777
      IF ((BD1.EQ.1).AND.(BD1.EQ.BD2)) GOTO 100
      IF (CQ.EQ.1) GOTO 777
      IF ((CQ.EQ.-1).AND.(BD1.NE.1).AND.(BD2.NE.1)) GOTO 777
      GOTO 100
52    IF (YP1.NE.YP2+1) GOTO 55
      IF (XKO.EQ.XP2) GOTO 777
      IF ((CQ.EQ.1).AND.(BD1.GT.1)) GOTO 777
      IF ((CQ.EQ.-1).AND.(BD1.GT.2)) GOTO 777
      GOTO 100
55    IF (BD1.GE.DP+2) GOTO 777
      GOTO 100
C
C     3 FILES APART
C
60    IF (IABS(XP1-XP2).NE.3) GOTO 85
      IF (YP2.GE.6) GOTO 777
      IF ((CQ.EQ.1).AND.(YP2.GE.5)) GOTO 777
```

```
      IF ((YP1.EQ.3).AND.(YP2.EQ.2).AND.(BD1.NE.1)) GOTO 777
      IF ((YP1.EQ.3).AND.(YP2.EQ.3).AND.(BD1.NE.1).AND.
     1(BD2.NE.1)) GOTO 777
62    IF ((YP1.EQ.3).AND.(YP2.EQ.3).AND.(MOD(XP1,7).EQ.1).AND.
     1(XK.EQ.XP1).AND.(YK.EQ.1).AND.(IABS(XKO-XP1).EQ.1)
     1.AND.(YKO.EQ.3)) GOTO 888
      IF (ABCD.EQ.1) GOTO 64
C
C     SINCE BOTH PAWNS HERE ARE EQUALLY ADVANCED WE MUST
C     SWITCH THE COORDINATES TO RETEST THE ABOVE FORMULA.
C
      IF (MOD(XP2,7).NE.1) GOTO 65
64    AAA=XP1
      BBB=YP1
      XP1=XP2
      YP1=YP2
      XP2=AAA
      YP2=BBB
      IF (ABCD.EQ.1) GOTO 65
      ABCD=1
      GOTO 62
65    IF ((CQ.EQ.1).AND.(YP1.EQ.3))  GOTO 777
      IF (YP1.LT.6) GOTO 100
      AA=-1
80    AA=BD1+AA
      IF ((YP1.EQ.YP2).AND.(BD2.LT.BD1)) AA=AA+BD2-BD1
      IF (CQ.EQ.1) AA=AA+1
      IF (YP2.GT.3) AA=AA+YP2-3
      IF (AA.GE.3) GOTO  777
      GOTO 100
C
C     4 FILES APART
C
85    IF (IABS(XP1-XP2).NE.4) GOTO 90
      IF (YP2.GE.5) GOTO 777
      IF ((YP1.EQ.3).AND.(CQ.EQ.1)) GOTO 777
      IF ((YP1.GT.YP2).AND.(BD2.EQ.1)) GOTO 777
      IF ((BD1.GE.2).AND.(CQ.EQ.1)) GOTO 777
      AA=0
      GOTO 80
C
C     5 FILES APART
C
90    IF (IABS(XP1-XP2).NE.5) GOTO 777
      IF ((CQ.EQ.1).OR.(BD1.NE.1)) GOTO 777
      IF (YP2.GE.4) GOTO 777
C
C     HERE WE CHECK WHETHER THE WHITE KING WINS WITH ONLY ONE
C     PAWN BY CALLING ONEPAN, ONCE FOR EACH PAWN.
C
100   YZA=MATERW
      ZYA=MATERB
      MATERW=1
      MATERB=0
      ABC=WHITEX(1)
      DEF=WHITEY(1)
      GHI=WHITEP(2)
      JKL=WHITEX(2)
```

```
      MNO=WHITEY(2)
      PQR=BLACKX(1)
      STU=BLACKY(1)
      VWX=C
      YNOT=SCORE(PLY+3)
      WHITEX(1)=XK
      WHITEY(1)=YK
      WHITEP(2)=1
      WHITEX(2)=XP1
      WHITEY(2)=YP1
      BLACKX(1)=XKO
      BLACKY(1)=YKO
      C=CQ
      CALL ONEPAN
      IF (IABS(SCORE(PLY+3)).GT.20) GOTO 777
      SCORE(PLY+3)=YNOT
      IF (RKPW.EQ.1) GOTO 910
      WHITEX(2)=XP2
      WHITEY(2)=YP2
      CALL ONEPAN
      IF (IABS(SCORE(PLY+3)).GT.20) GOTO 777
      SCORE(PLY+3)=YNOT
      GOTO 300
C
C     WE CONSIDER DOUBLED PAWNS IN GENERAL. IF RKPW IS SET
C     EQUAL TO 1 WE DEAL WITH THE SPECIAL CASE OF DOUBLED
C     ROOKPAWNS.
C
200   IF (MOD(XP1,7).EQ.1) RKPW=1
      IF ((YP1.EQ.6).AND.(YP2.EQ.5).AND.(((XKO.EQ.XP1).AND.
     1(YKO.EQ.7)).OR.((YK.EQ.6).AND.(CQ.EQ.-1).AND.(YKO.EQ.8)
     1.AND.(IABS(XK-XP1).EQ.1).AND.(XKO.EQ.XP1)))) GOTO 888
      IF ((YP1.EQ.YP2+1).AND.(DIST(XK,YK,XP1,YP1).EQ.1).AND.
     1(YK.LT.YP1).AND.(IABS(XKO-XP1).LE.1).AND.(((YKO.EQ.YP1+1)
     1.AND.(CQ.EQ.1)).OR.((CQ.EQ.-1).AND.(YKO.EQ.YP1+2)))) GOTO 888
      GOTO 100
C
C     SPECIAL TESTS TO DETERMINE WHETHER WHITE WILL SUCCEED
C     IN PROTECTING HIS MOST ADVANCED PAWN WITH HIS KING
C     AND THUS WIN.
C
300   IF (DIST(XK,YK,XP1,YP1).EQ.1) GOTO 777
      IF (CQ.EQ.-1) BD1=BD1-1
      IF (DIST(XK,YK,XP1,YP1).LE.BD1-1) GOTO 777
C
C     IF A POSITION CANNOT BE EVALUATED AND THE PLY LEVEL IS
C     LESS THAN D, THE MAXIMUM DEPTH, WE RETURN WITHOUT
C     CHANGING THE SCORE VECTOR.
C
      IF (PLY.GE.D) GOTO 850
      DRAWC3=0
      GOTO 889
C
C     SCORES WON AND LOST POSITIONS
C
777   ADD=7- DIST(XK,YK,XP1,YP1+1)+ DIST(XKO,YKO,XP1,YP1+1)
      IF ((MATERW.EQ.0).OR.(YZA.EQ.0)) GOTO 801
      SCORE(PLY+3)=6000-200*PLY-400*(8-YP1)-100*(1-C)+ADD
```

```
      DRAWC3=1
      GOTO 889
801   SCORE(PLY+3)=-6000+200*PLY+400*(8-YP1)+100*(1+C)-ADD
      DRAWC3=1
      GOTO 889
850   ADD1=7-DIST(XK,YK,XP1,YP1+1)+DIST(XKC,YKO,XP1,YP1+1)
C
C     SCORES UNDECIDED POSITIONS AT MAXIMUM DEPTH
C
      IF ((MATERW.EQ.0).OR.(YZA.EQ.0)) GOTO 860
      SCORE(PLY+3)=ADD1+25
      DRAWC3=1
      GOTO 889
860   SCORE(PLY+3)=-25-ADD1
      DRAWC3=1
      GOTO 889
C
C     SCORES DRAWN POSITIONS
C
878   SCORE(PLY+3)=-ADD1
      DRAWC3=1
      GOTO 889
888   IF ((MATERW.EQ.0).OR.(YZA.EQ.0)) GOTO 878
      SCORE(PLY+3)=ADD1
      DRAWC3=1
889   IF (MATERW.EQ.1) GOTO 900
      RETURN
900   MATERW=YZA
      MATERB=ZYA
      WHITEX(1)=ABC
      WHITEY(1)=DEF
      WHITEP(2)=GHI
      WHITEX(2)=JKL
      WHITEY(2)=MNO
      BLACKX(1)=PQR
      BLACKY(1)=STU
      C=VWX
      RETURN
910   IF ((YK.LE.YP1+1).OR.(YKO.GE.YP1+2)) GOTO 888
      DRAWC3=0
      GOTO 889
      END
      SUBROUTINE PAWN2
      IMPLICIT INTEGER (A-Z)
      COMMON /COLOR/ COLOUR
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /SKORE/ MATERW,MATERB
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /PLISTZ/ QC,KXO,KYO,KX,KY,PX,PY,PX1,PY1,
     1KX1,KY1,KX2,KY2
      COMMON /PLISTX/ XP1,YP1,XP2,YP2,XK,YK,XKO,YKO,NLOCK,CP
C
C     SUBROUTINE PAWN2 IS CALLED FROM TREE AND IT TRIES TO
C     DETERMINE WHETHER A GIVEN POSITION, WHERE EACH SIDE HAS
C     BUT ONE PAWN, IS TERMINAL. THE SETUP HERE IS SOMEWHAT
```

```
C     DIFFERENT FROM ONEPAN. THE ORIGINAL POSITION SUBMITTED
C     FROM TREE IS SETUP TO START WITH. THERE ARE THREE PAWN
C     VS. PAWN SITUATIONS TO BE CONSIDERED A) PASSED PAWNS:
C     EACH PAWN IS FREE TO ADVANCE TO THE QUEENING SQUARE
C     WITHOUT OBSTRUCTION OR THREAT OF CAPTURE BY THE
C     OPPOSING PAWN, B) BLOCKED PAWNS: BOTH PAWNS ARE ON
C     THE SAME FILE AND CANNOT ADAVNCE TO QUEEN UNLESS THE
C     OPPOSING PAWN IS CAPTURED BY THE KING, C) ADJACENT
C     PAWNS: PAWNS ARE SEPARATED BY ONE FILE AND IF EITHER
C     SIDE ADVANCES TO THE QUEENING SQUARE THE OPPOSING PAWN
C     MAY CAPTURE IT WHILE IT PASSES. IF THE PAWNS ARE PASSED
C     A CALL IS MADE TO PASPON; OTHERWISE PAWN2 TESTS THE
C     POSITION.IN EITHER CASE, IF THE POSITION CANNOT BE EVALUATED
C     AS A CLEAR WIN,LOSS OR DRAW, A CALL IS MADE TO REVERS
C     (WHICH IN TURN CALLS DUAL) TO EFFECTIVELY REVERSE THE
C     GIVEN POSITION CHESSWISE. CONTROL IS RETURNED TO THE
C     CALLING PROGRAM, EITHER PAWN2 OR PASPON AND THE
C     REVERSED POSITION IS SUBJECTED TO THE SAME SERIES
C     OF TESTS. THE SCORING FOLLOWS ALONG SIMILAR LINES TO
C     TWOPAN.FINALLY ONE MORE CALL IS MADE TO REVERS OR DUAL
C     TO SET UP THE ORIGINAL POSITION AND RETURN. THE
C     VARIABLE NLOCK IS SET TO A) 0 FOR BLOCKED POSITIONS WHERE
C     PAWNS ARE ALSO LOCKED IE. ONE RANK APART, B) 1 FOR
C     BLOCKED BUT NOT LOCKED PAWN POSITIONS, C) 2  FOR
C     ADJACENT PAWNS, D) 3 FOR PASSED PAWNS, AND E) 4 FOR
C     BLOCKED ROOK PAWNS, AN EXCEPTIONAL CASE.
C
      IF ((MATERW.EQ.1).AND.(MATERB.EQ.-1)) GOTO 1
      RETURN
C
C     SETUP
C
1     REV=0
      PCHK=0
      CP=C
2     QC=CP
      J=2
5     IF (WHITEP(J).EQ.1) GOTO 10
      J=J+1
      GOTO 5
10    XP1=WHITEX(J)
      YP1=WHITEY(J)
      J=2
15    IF (BLACKP(J).EQ.-1) GOTO 16
      J=J+1
      GOTO 15
16    XP2=BLACKX(J)
      YP2=BLACKY(J)
      XK=WHITEX(1)
      YK=WHITEY(1)
      XKO=BLACKX(1)
      YKO=BLACKY(1)
C
C     AS A DEPARTURE FROM THE NORM THE VARIABLE YP11 TAKES BOTH
C     PAWNS' HEIGHT INTO CONSIDERATION WHEN CALCULATING THE SCORE
C     OF A WON OR LOST POSITION (SEE 777 AND 1111).
C
      YP11=16-YP1-YP2
```

```
      MOVE(699)=COLOUR
      IF (((MOD(PLY,2).EQ.1).AND.(C.EQ.1)).OR.
     1((MOD(PLY,2).EQ.0).AND.(C.EQ.-1))) YP11=16-YP11
      DRAWC3=0
      NLOCK=0
C
C     THIS TEST ENABLES SUBPROGRAM TREE TO CONTINUE GENERATING
C     MOVES AFTER MAXIMUM DEPTH HAS BEEN REACHED. IF EITHER KING
C     MIGHT CAPTURE AN UNDEFENDED PAWN ON THE NEXT ONE OR TWO
C     PLIES, THE PARAMETER MOVE(700) IS SET EQUAL TO 1 AND THIS
C     CAUSES THE SEARCH TO CONTINUE WHEN CONTROL RETURNS TO TREE.
C
      IF ((PLY.LT.D).OR.(PLY.GT.D+1)) GOTO 19
      IF (((DIST(XK,YK,XP2,YP2).EQ.1).AND.
     1(DIST(XKO,YKO,XP2,YP2).NE.1)).OR.((DIST(XKO,YKO,XP1,YP1).EQ.1)
     1.AND.(DIST(XK,YK,XP1,YP1).NE.1))) MOVE(700)=1
19    IF ((IABS(XP1-XP2).GT.1).OR.(YP1.GT.YP2)) CALL PASPON
      IF (NLOCK.EQ.3) RETURN
      ADD1=DIST(XK,YK,XP2,YP2)+DIST(XKO,YKO,XP1,YP1)
18    TEMP=0
      IF (XP1.NE.XP2) GOTO 330
C
C     BLOCKED PAWNS
C
      IF (MOD(XP1,7).EQ.1) GOTO 300
      IF (YP2.GT.YP1+1) GOTO 220
      IF (YP1.GE.5) GOTO 20
C
C     DOES BLACK HAVE THE OPPOSITION AFTER WHITE WINS THE PAWN?
C
      PR=0
      IF (QC.EQ.-1) PR=1
      IF (DIST(XK,YK,XP2,YP2)+PR.LE.
     1DIST(XKO,YKO,XP2,YP2+2)-1) GOTO 20
      IF (DIST(XKO,YKO,XP2,YP2+2).EQ.1) GOTO 170
      IF ((QC.EQ.1).AND.(DIST(XK,YK,XP2,YP2).EQ.1)
     1.AND.(XKO.EQ.XP2).AND.(YKO.EQ.YP2+2)) GOTO 777
      GOTO 50
C
C     CAN WHITE CAPTURE BLACK'S PAWN?
C
20    IF (YK.GE.YP1) GOTO 40
      RC=DIST(XK,YK,XP2+2,YP1)
      IF (XP2.EQ.2) GOTO 25
      PT=DIST(XK,YK,XP2-2,YP1)
      IF (XP2.EQ.7) GOTO 30
      IF (PT.GT.RC) PT=RC
      GOTO 30
25    PT=RC
30    IF (QC.EQ.-1) PT=PT+1
      IF (PT+2.LE.DIST(XKO,YKO,XP2,YP2)-1) GOTO 777
      GOTO 50
40    PT=0
      IF (QC.EQ.-1) PT=1
      IF (DIST(XK,YK,XP2,YP2)+PT.LE.
     1DIST(XKO,YKO,XP2,YP2)-1) GOTO 777
C
C
```

```
C     SPECIAL CASES
C
C
50    IF ((YP1.LE.4).AND.(YKO.LE.YP1)) GOTO 80
      IF (YP1.LE.4) GOTO 110
      IF ((YK.EQ.YP2).AND.(IABS(XK-XP2).LE.3)
     1.AND.(YKO.GT.YP2)) GOTO 60
      GOTO 70
60    IF ((MOD(XP2,5).NE.2).OR.(YP2.NE.7).OR.
     1(MOD(XK,7).EQ.1)) GOTO 777
      GOTO 888
C
C     CHECK FOR OPPOSITION BETWEEN KINGS IE. TWO RANKS APART
C
70    IF ((YK.EQ.YP1).AND.(XK.EQ.XKO).AND.(YKO.EQ.YP2+1)) GOTO 100
C
C     MOST POSITIONS WHERE THE KING IS CLOSE TO THE ENEMY PAWN
C     ARE EVALUATED HERE.
C
80    IF ((YK.EQ.YP2).AND.(IABS(XK-XP2).EQ.2).AND.
     1(QC.EQ.1).AND.(DIST(XKO,YKO,XP1,YP1).NE.1)) GOTO 777
      GOTO 110
100   IF ((QC.EQ.1).OR.((MOD(XP2,5).EQ.2).AND.
     1(YP2.EQ.7))) GOTO 888
      GOTO 777
110   IF ((IABS(XK-XP2).EQ.1).AND.(YK.EQ.YP2)) GOTO 120
      GOTO 130
120   IF ((IABS(XKO-XP1).EQ.1).AND.(YKO.EQ.YP1).AND.
     1(QC.EQ.-1)) GOTO 777
      XPR=XP1+1
      IF (XK.GT.XP1) XPR=XP1-1
      IF (DIST(XKO,YKO,XPR,YP1).EQ.1) GOTO 999
      IF ((IABS(XK-XKO).LE.1).AND.(YKO.EQ.YP1-1)) GOTO 777
C
C     CHECK FOR ANOTHER FORM OF OPPOSITION BETWEEN KINGS
C     IE. TWO FILES APART (SIDEWAYS).
C
130   IF ((YK.EQ.YKO).AND.(IABS(XK-XKO).EQ.2)) GOTO 150
      GOTO 170
150   IF (IABS(XK-XP1).LT.IABS(XKO-XP2)) GOTO 160
      GOTO 170
160   IF ((IABS(XK-XP1).LE.1).AND.(YK.LE.YP1)) GOTO 170
      IF (YP1.GT.4) GOTO 777
      RT=2
      IF ((NLOCK.EQ.1).AND.(QC.EQ.-1)) RT=1
      IF (YK.EQ.8) RT=RT+1
      IF (YK.GE.YP2+IABS(XK-XP2)+RT) GOTO 888
      GOTO 777
170   IF (REV.EQ.1) GOTO 180
      CALL REVERS
      REV=1
      QC=-CP
      GOTO 18
180   IF (NLOCK.NE.0) GOTO 1111
      TEMP=2
      GOTO 1333
C
C     BLOCKED AND LOCKED PAWNS ARE SUBJECTED TO AN UNUSUAL TEST.
```

```
C     AFTER THE SECOND CALL TO REVERS HAS ALREADY BEEN MADE,
C     CONTROL RETURNS HERE TO TEST FOR A PARTICULAR POSITION OF
C     THE ATTACKING KING.
C
185    TEMP=0
      REV=0
      QC=CP
      IF ((XK.EQ.XP2).AND.(YK.EQ.YP2+1)) GOTO 190
      GOTO 200
190   IF (YKO.LE.YK) GOTO 777
      DT=0
      IF (QC.EQ.-1) DT=1
      IF (DIST(XKO,YKO,XP2,YP2+2).GT.DT+1) GOTO 777
      GOTO 888
200   IF ((XKO.EQ.XP1).AND.(YKO.EQ.YP1-1)) GOTO 210
      GOTO 1111
210   IF (YK.GE.YKO) GOTO 999
      DT=0
      IF (QC.EQ.1) DT=1
      IF (DIST(XK,YK,XP1,YP1-2).GT.DT+1) GOTO 999
      GOTO 888
C
C     PAWNS ARE ON THE SAME FILE BUT NOT LOCKED
C
220   NLOCK=1
      IF ((YKO.GT.YP1).AND.(YKO.LT.YP2).AND.
     1(IABS(XKO-XP1).LE.1)) GOTO 470
      IF ((DIST(XKO,YKO,XP1,YP1).EQ.1).AND.
     1(YK.LT.YP2)) GOTO 470
      IF ((XK.EQ.XP2).AND.(YK.EQ.YP2-1).AND.
     1(DIST(XKO,YKO,XP2,YP2).EQ.1)) GOTO 888
      IF ((YP1.EQ.5).AND.(YK.EQ.7).AND.(YKO.EQ.7).AND.
     1(IABS(XK-XP2).EQ.1).AND.(IABS(XKO-XP2).EQ.1)) GOTO 230
      GOTO 240
230   IF (QC.EQ.1) GOTO 888
      IF (QC.EQ.-1) GOTO 60
240   PT=DIST(XKO,YKO,XP1,YP1)
      IF (QC.EQ.-1) PT=PT-1
      IF ((YP1.EQ.2).AND.(YKO.EQ.1)) PT=PT+1
      IF (PT.LT.DIST(XK,YK,XP2,YP2)) GOTO 470
C
C     DISTANCE TEST
C
250   DT=0
      IF (QC.EQ.1) DT=-1
      IF ((YP1-YKO.EQ.3).AND.(NLOCK.EQ.1)) DT=DT-1
      IF (YK.GE.YP2-1) GOTO 260
      DT=DT+DIST(XK,YK,XP2,YP2-1)+1
      GOTO 270
260   DT=DT+DIST(XK,YK,XP2,YP2)
270   IF (DT.LE.DIST(XKO,YKO,XP2,YP2)-2) GOTO 280
      GOTO 290
280   DT=DT+YP2-YP1-2
      IF (DT.LE.DIST(XKO,YKO,XP1,YP1)-2) GOTO 777
290   IF (NLOCK.EQ.2) GOTO 460
      IF ((IABS(XK-XP1).GE.2).OR.(YK.GE.YP2)) GOTO 130
      GOTO 170
C
```

```
C     ROOKPAWNS,BOTH LOCKED AND NOT LOCKED
C
300   NLOCK=4
      DT=DIST(XK,YK,XP2,YP2)-1
      IF (QC.EQ.-1) DT=DT+1
      PR=DT
      IF ((YP1.EQ.2).AND.(YP2.NE.3)) PR=PR-1
      IF (DT.EQ.0) GOTO 310
      IF ((IABS(XKO-XP1).LE.DT).AND.(YKO.EQ.YP1-PR)) GOTO 170
310   IF ((IABS(XKO-XP1).LE.DT+3).AND.(YKO.GE.YP1-PR)) GOTO 470
      GOTO 170
C
C     PAWNS ARE ON ADJACENT FILES
C
330   IF (YP2.EQ.YP1) GOTO 340
      IF (YP2.EQ.YP1+1) GOTO 350
      GOTO 360
C
C     CHECKING FOR EN PASSANT; IF EP. IS POSSIBLE
C     THE PAWN MOVE IS REJECTED
C
340   IF ((IABS(MOVE(MP(PLY)))-60000000).GE.0) CALL PASPON
      IF (NLOCK.EQ.3) RETURN
      XTEMP=IABS(MOVE(MP(PLY)))/100
      XTEMP=MOD(XTEMP,10)
      IF ((YP1.EQ.5).AND.(QC.EQ.1).AND.(XTEMP.EQ.7)) GOTO 777
      IF ((YP2.EQ.4).AND.(QC.EQ.-1).AND.(XTEMP.EQ.2)) GOTO 999
      CALL PASPON
      RETURN
350   IF ((IABS(MOVE(MP(PLY)))-60000000).GE.0) GOTO 355
C
C     IF PAWN MAY CAPTURE PAWN, CONTROL RETURNS IMMEDIATELY TO
C     SUBROUTINE TREE AND THE SEARCH CONTINUES (EVEN IF THE
C     PLY LEVEL IS EQUAL TO OR GREATER THAN THE MAXIMUM DEPTH).
C
      IF (PLY.GE.D) MOVE(700)=1
      GOTO 1111
355   IF (MOD(PLY,2).EQ.1) GOTO 999
      GOTO 777
360   NLOCK=2
      IF ((MOD(XP2,7).EQ.1).AND.(YP2.EQ.7).AND.(XKO.EQ.XP2)
     1.AND.(YP1.EQ.5).AND.(YKO.EQ.8)) GOTO 370
      GOTO 390
370   IF ((YK.GE.7).AND.(IABS(XK-XKO).EQ.2)) GOTO 777
      IF ((QC.EQ.1).AND.(YK.GE.6).AND.(IABS(XK-XKO).LE.3)
     1.AND.(IABS(XK-XKO).GT.1)) GOTO 777
390   IF ((DIST(XKO,YKO,XP2,YP2).EQ.1).OR.
     1(DIST(XKO,YKO,XP1,YP1).EQ.1)) GOTO 470
C     EXCEPTIONAL CASES WITH ROCKPAWNS FOR EITHER SIDE.
      IF (MOD(XP1,7).EQ.1) GOTO 450
      IF (MOD(XP2,7).EQ.1) GOTO 170
400   IF ((YKO.EQ.YK).AND.(IABS(XK-XKO).EQ.2)) GOTO 410
      GOTO 250
C
C     CHECK FOR SIDEWAYS  OPPOSITION BETWEEN KINGS IE. TWO
C     FILES APART
C
410   DT=0
```

```
      PR=0
      IF (IABS(XK-XP2).LT.IABS(XKO-XP2)) GOTO 420
      GOTO 250
420   IF (IABS(XKO-XP1).GT.IABS(XKO-XP2)) GOTO 440
      IF ((YP2.EQ.YP1+2).OR.((QC.EQ.1).AND.((YP2.EQ.YP1+3).OR.
     1((YP1.EQ.2).AND.(YP2.EQ.YP1+4))))) GOTO 430
      GOTO 250
430   IF (XP2-XK.GT.0) DT=2*(XP2-XK)
      IF (XK+YK+DT.GE.XP2+YP2+PR+1) GOTO 777
      GOTO 250
440   IF ((MOD(XP1,7).EQ.1).AND.(((YP1.LT.4).AND.(QC.EQ.-1)).OR.
     1(YP2.LE.5))) GOTO 450
      IF ((YP2.EQ.YP1+2).OR.((QC.EQ.1).AND.((YP2.EQ.YP1+3).OR.
     1((YP1.EQ.2).AND.(YP2.EQ.YP1+4))))) PR=-3
      IF (PR.EQ.-3) GOTO 430
      IF ((YP2.EQ.YP1+3).OR.((QC.EQ.1).AND.((YP2.EQ.YP1+4).OR.
     1((YP1.EQ.2).AND.(YP2.EQ.YP1+5))))) PR=-2
      GOTO 430
450   DT=0
      IF (QC.EQ.1) DT=-1
      XPR=3
      IF (XP1.EQ.8) XPR=6
      IF (DIST(XKO,YKO,XPR,8).LE.
     1DIST(XK,YK,XP1,8)+DT-1) GOTO 470
      IF (YP1.GE.4) GOTO 400
      GOTO 170
460   DT=0
      IF (QC.EQ.1) DT=-1
      IF (DIST(XK,YK,XP2,YP2)+DT.GE.
     1DIST(XKO,YKO,XP1,YP1)) GOTO 470
      GOTO 170
C
C     THE VARIABLE PCHK IS SET EQUAL TO A) 1, IF BLACK HAS AT
C     LEAST A DRAW, B) 2, IF WHITE HAS AT LEAST A DRAW, C) 0, IF
C     NEITHER SIDE HAS A DRAW, AND D) 2, IF BOTH SIDES HAVE AT
C     LEAST A DRAW. IF THE PLY LEVEL IS GREATER THAN OR EQUAL
C     TO MAXIMUM DEPTH AND CONTRCL SWITCHES TO 1222 THE POSITION
C     IS SCORED ACCORDINGLY.
C
470   IF ((REV.EQ.1).AND.(PCHK.EQ.0)) PCHK=3
      IF (PCHK.EQ.1) PCHK=2
      IF (PCHK.EQ.0) PCHK=1
      GOTO 170
777   IF (REV.EQ.1) GOTO 1200
      DRAWC3=1
      SCORE(PLY+3)=6000-100*PLY-200*YP11-50*(1-C)-ADD1
      GOTO 1333
888   DRAWC3=1
      SCORE(PLY+3)=0
      GOTO 1333
999   IF (REV.EQ.1) REV=2
      IF (REV.EQ.2) GOTO 1200
      DRAWC3=1
      SCORE(PLY+3)=-6000+100*PLY+200*(16-YP11)+50*(1+C)-ADD1
      GOTO 1333
1111  IF (MOVE(700).EQ.1) GOTO 1112
      IF (PLY.GE.D) GOTO 1222
1112  DRAWC3=0
```

```
      GOTO 1333
1200  TEMP=1
      IF (REV.EQ.2) GOTO 1210
      REV=0
      GOTO 999
1210  REV=0
      GOTO 777
1222  IF (PCHK.EQ.0) SCORE(PLY+3)=-ADD1
      IF (PCHK.EQ.1) SCORE(PLY+3)=-5-ADD1
      IF (PCHK.EQ.3) SCORE(PLY+3)=5-ADD1
      IF (PCHK.EQ.2) SCORE(PLY+3)=5-ADD1
      DRAWC3=1
1333  IF ((REV.EQ.0).AND.(TEMP.EQ.0)) RETURN
      CALL REVERS
      COLOUR=MOVE(699)
      IF (TEMP.EQ.2) GOTO 185
      RETURN
      END
      SUBROUTINE REVERS
      IMPLICIT INTEGER (A-Z)
      COMMON /COLOR/ CCLOUR
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /SKORE/ MATERW,MATERB
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /PLISTZ/ QC,KXO,KYO,KX,KY,PX,PY,PX1,PY1,
     1KX1,KY1,KX2,KY2
      COMMON /PLISTX/ XP1,YP1,XP2,YP2,XK,YK,XKO,YKO,NLOCK,CP
      COLOUR=0
      CALL DUAL
      DO 30 I=1,8
      DO 30 J=1,8
      IF (BOARD(I,J).NE.6) GOTO 15
      XK=I
      YK=J
      GOTO 30
15    IF (BOARD(I,J).NE.1) GOTO 20
      XP1=I
      YP1=J
      GOTO 30
20    IF (BOARD(I,J).NE.-6) GOTO 25
      XKO=I
      YKO=J
      GOTO 30
25    IF (BOARD(I,J).NE.-1) GOTO 30
      XP2=I
      YP2=J
30    CONTINUE
      RETURN
      END
      SUBROUTINE PASPON
      IMPLICIT INTEGER (A-Z)
      COMMON /COLOR/ COLOUR
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /SKORE/ MATERW,MATERB
```

```
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /PLISTZ/ QC,KXO,KYO,KX,KY,PX,PY,PX1,PY1,
     1KX1,KY1,KX2,KY2
      COMMON /PLISTX/ XP1,YP1,XP2,YP2,XK,YK,XKO,YKO,NLOCK,CP
      NLOCK=3
      REV=0
      PCHK=0
      YP11=16-YP1-YP2
      IF (((MOD(PLY,2).EQ.1).AND.(C.EQ.1)).OR.
     1((MOD(PLY,2).EQ.0).AND.(C.EQ.-1))) YP11=16-YP11
      ADD2=DIST(XK,YK,XP2,YP2)+DIST(XKO,YKO,XP1,YP1)
500   SWAT=0
      TRICK=0
      TEMP=0
C
C     PASPON CHECKS WHETHER THE WHITE KING LIES IN FRONT OF THE
C     BLACK PAWN AND ONE FILE ADJACENT TO IT, AND THUS SUBJECT
C     TO A CHECK BY THE ENEMY PAWN. IF THE KING IS ACTUALLY
C     IN CHECK AT ANY PLY LEVEI, CCNTROL RETURNS TO SUBROUTINE
C     TREE AND THE SEARCH CONTINUES (GENERATING MOVES FROM THE
C     'CHECKING' POSITION). OTHERWISE REVERS IS CALLED AND,
C     UNLESS THE NODE IS FOUND TO EE TERMINAL, ONCE CONTROL RE
C     TURNS TO TREE THE SEARCH CONTINUES, EVEN AT MAXIMUM DEPTH.
C
      IF ((IABS(XK-XP2).NE.1).OR.(YK.GE.YP2)) GOTO 505
      IF ((PLY.EQ.D).OR.(YK.EQ.YP2-1)) MOVE(700)=1
      IF (YK.EQ.YP2-1) GOTO 170
      IF (IABS(XKO-XP2).GT.IABS(XKO-XK)) GOTO 505
      GOTO 170
C
C     IS BLACK'S KING IN THE 'SQUARE' OF WHITE'S PAWN?
C
505   PTO=0
      DT=9-YP1
      IF (YP1.EQ.2) DT=6
      IF (QC.EQ.1) DT=DT-1
      IF (DIST(XKO,YKO,XP1,8).LE.DT) GOTO 1000
      IF ((YP1.EQ.2).AND.(DIST(XKO,YKO,XP1,YP1).EQ.1)
     1.AND.(QC.EQ.-1)) GOTO 170
      IF ((XK.EQ.XP2).AND.(YK.LT.YP2)) GOTO 777
C
C     IS WHITE'S KING WITHIN THE SQUARE OF BLACK'S PAWN?
C
510   PR=YP2
      IF (YP2.EQ.7) PR=6
      IF (QC.EQ.-1) PR=PR-1
      IF (PTO.GE.1) GOTO 540
C
C
C     IS WHITE'S KING CIOSE ENOUGH TO BLACK'S PAWN TO PREVENT
C     BLACK'S KING ASSISTING IT TO QUEEN?
C
C
      IF ((DIST(XK,YK,XP2,1).GT.PR).OR.
     1(IABS(XKO-XK).GE.IABS(XKO-XP2))) GOTO 540
      RC=1
```

```
      IF (YK-YKO.EQ.2) RC=0
      TF=0
      IF (DIST(XK,YK,XP2,1).EQ.PR) TF=1
      IF ((TF.EQ.1).AND.(QC.EQ.1)) GOTO 515
      IF ((QC.EQ.-1).AND.(YK.EQ.5).AND.(YP1.LE.3)) RC=2
      GOTO 520
515   IF (DIST(XKO,YKO,XP1,8).EQ.DT+1) GOTO 540
520   IF (IABS(XP2-XK).GE.IABS(XP2-XKO)-RC) GOTO 540
      IF ((QC.EQ.1).AND.(DIST(XKO,YKO,XP1,8).NE.DT+1)) GOTO 530
      IF ((XP1.EQ.XK).AND.(YP1.LT.YK)) GOTO 540
530   IF ((TF.EQ.1).AND.(YP1.EQ.2).AND.
     1(DIST(XKO,YKO,XP1,YP1).EQ.1)) GOTO 540
C
C     WE MUST ALSO CHECK WHETHER WHITE'S OWN PAWN PREVENTS HIM
C     FROM CATCHING BLACK'S PAWN AND ADVANCING HIS OWN.IE.
C     WHEN WHITE'S KING MUST CROSS HIS OWN PAWN'S FILE TO
C     INTERCEPT BLACK'S PAWN (IF IT ADVANCES).
C
      IF (IABS(XP2-XK).LE.IABS(XP2-XP1))  GOTO 777
      RC=0
      IF (XK-XP1.GT.0) RC=2*(XK-XP1)
      IF (XP1+YP1+RC.LE.XK+YK) GOTO 540
      GOTO 777
C
C     WHO WINS RACE TO QUEEN PAWN?
C
540   IF ((XK.EQ.XP1).AND.(YK.GT.YP1)) DT=DT+1
      IF ((XKO.EQ.XP2).AND.(YKO.LT.YP2)) PR=PR+1
      IF (QC.EQ.-1) PR=PR+1
      IF (PTO.GE.2) DT=DT+1-PTO
      IF (PTO.GE.2) GOTO 1030
      IF (PR-DT.LT.0) GOTO 470
      IF ((PR-DT.EQ.3).AND.(XKO.EQ.XP2).AND.(YKO.EQ.1)) GOTO 1170
      IF (PR-DT.GT.2) GOTO 777
      IF (PR-DT.LT.2) GOTO 550
      IF ((MOD(XP2,7).NE.1).AND.(MOD(XP2,3).NE.0)) GOTO 777
550   SR=0
C
C     CALL REVERS IF EITHER SIDE BLOCKS THEIR OWN PAWN.
C
      IF ((XK.EQ.XP1).AND.(YK.GT.YP1)) GOTO 170
      IF ((XKO.EQ.XP2).AND.(YKO.LT.YP2)) GOTO 170
C
C     DOES PAWN QUEEN WITH CHECK?
C
      IF (YKO.EQ.8) SR=1
      IF (XKO.EQ.XP1) SR=2
      RC=0
      IF (XP1-XKO.GT.0) RC=2*(XP1-XKO)
      IF(XKO+YKO+RC.EQ.XP1+8) SR=3
      IF (SR.EQ.0) GOTO 570
C
C     DOES WHITE'S KING BLOCK HIS OWN PAWN'S CHECK?
C
      IF (DIST(XK,YK,XP1,8).GE.DIST(XKO,YKO,XP1,8)-1) GOTO 560
      IF ((SR.EQ.1).AND.(YK.EQ.8)) SR=-1
      IF ((SR.EQ.2).AND.(XK.EQ.XP1)) SR=-2
      IF (SR.NE.3) GOTO 560
```

```
      RC=0
      IF (XP1-XK.GT.0) RC=2*(XP1-XK)
      IF (XK+YK+RC.EQ.XP1+8) SR=-3
C
C     IF WHITE QUEENS HIS PAWN AFTER BLACK QUEENS HIS OWN,
C     WITHOUT CHECKING BLACK'S KING IN THE PROCESS, WE CALL
C     REVERS AND NOTE THAT BLACK HAS AT LEAST A DRAW.
C
560   IF ((SR.LT.0).AND.(PR.EQ.DT)) GOTO 470
      IF (PR.NE.DT) GOTO 570
C
C     CAN BLACK'S QUEEN CAPTURE WHITE'S IF BLACK QUEENS FIRST?
C
      IF ((XP1.EQ.XP2).AND.(SR.NE.2)) GOTO 170
      IF ((IABS(XP1-XP2).EQ.7).AND.(SR.NE.3)) GOTO 170
      GOTO 580
570   IF (PR.EQ.DT) GOTO 470
      IF (PR-DT.EQ.2) GOTO 1100
      IF (SR.GT.0) GOTO 780
      TRICK=1
C
C     DOES BLACK QUEEN WITH CHECK?
C
580   MT=0
      IF (YK.EQ.1) MT=1
      IF (XK.EQ.XP2) MT=2
      RC=0
      IF (XK-XP2.GT.0) RC=2*(XK-XP2)
      IF (XK+YK.EQ.XP2+1+RC) MT=3
C
C     DOES BLACK'S KING BLOCK HIS OWN PAWN'S CHECK?
C
      IF ((MT.EQ.0).OR.
     1(DIST(XKO,YKO,XP2,1).GE.DIST(XK,YK,XP2,1)-1)) GOTO 590
      IF ((MT.EQ.1).AND.(YKO.EQ.1)) MT=-1
      IF ((MT.EQ.2).AND.(XKO.EQ.XP2)) MT=-2
      IF (MT.NE.3) GOTO 590
      RC=0
      IF (XKO-XP2.GT.0) RC=2*(XKO-XP2)
      IF (XKO+YKO.EQ.XP2+1+RC) MT=-3
590   IF (PR.EQ.DT) GOTO 610
C
C     CAN  WHITE'S QUEEN CAPTURE BLACK'S IF WHITE QUEENS FIRST?
C
      IF ((XP1.EQ.XP2).AND.(MT.NE.2)) GOTO 600
      IF ((IABS(XP1-XP2).EQ.7).AND.(MT.NE.3)) GOTO 600
      GOTO 610
C
C     PERHAPS ALL FOUR PIECES ARE LINED UP ON CNE FILE OR THE
C     LONG DIAGONAL
C
600   IF ((SR.EQ.-2).AND.(SR.EQ.MT)) SWAT=1
      IF (SWAT.EQ.1) GOTO 620
      IF ((SR.EQ.-3).AND.(SR.EQ.MT)) SWAT=1
      IF (SWAT.EQ.1) GOTO 630
C
C     CAN BLACK PROTECT HIS QUEEN FROM CAPTURE AFTER QUEENING?
C
```

```
      IF ((DIST(XKO,YKO,XP2,1).EQ.1).AND.(MOD(XP2,7).EQ.1))GOTO 610
      IF (DIST(XKO,YKO,XP2,1).EQ.1) GOTO 760
      IF (DIST(XKO,YKO,XP2,2).GT.2) GOTO 777
      GOTO 780
610   IF (MT.GT.0) GOTO 470
C
C
C     TRAPS!! WINNING POSITIONS AFTER BOTH SIDES QUEEN.
C     IF TRICK IS EQUAL TO 1 WHITE QUEENS FIRST.
C
C
620   IF ((XKO.EQ.XP2).AND.(YKO.GT.3).AND.(YKO.LT.7)) GOTO 770
      IF (SWAT.EQ.1) GOTO 760
630   IF (MOD(XP2,7).NE.1) GOTO 650
640   IF ((MOD(XP1,7).EQ.1).AND.((SR.EQ.3).OR.(SWAT.EQ.1))
     1.AND.(IABS(XKO-XP2).GT.1)) GOTO 780
      IF (SWAT.EQ.1) GOTO 760
      IF ((DIST(XK,YK,XKO,YKO).EQ.2).AND.
     1(DIST(XKO,YKO,XP2,1).EQ.1)) GOTO 645
      GOTO 650
645   IF ((XK.NE.XKO).AND.(YK.NE.2)) GOTO 650
      IF ((TRICK.EQ.1).OR.((SR.GT.0).AND.(YKO.NE.1))) GOTO 777
650   IF ((SR.NE.2).AND.(TRICK.EQ.0)) GOTO 700
660   IF ((MOD(XKO,7).NE.1).OR.(IABS(XKO-XP2).NE.1)) GOTO 670
      IF ((YKO.EQ.4).AND.(DIST(XK,YK,XKO,2).EQ.2)) GOTO 777
      IF ((YKO.GT.4).AND.(YKO.LT.7)) GOTO 770
670   IF ((YK.EQ.YKO).AND.(IABS(XK-XKO).EQ.2).AND.
     1(IABS(XKO-XP2).EQ.1).AND.(IABS(XK-XP2).EQ.3)) GOTO 675
      GOTO 680
675   IF ((YKO.GT.4).AND.(YKO.LT.7)) GOTO 780
      IF ((YKO.LE.3).AND.(MOD(XP2,7).EQ.1).AND.(YKO.GE.2)) GOTO 777
680   IF ((YK.EQ.3).AND.(IABS(XK-XKO).EQ.2).AND.
     1(YKO.EQ.2).AND.(MOD(XKO,7).EQ.1)) GOTO 777
690   IF ((YK.EQ.3).AND.(IABS(XK-XKO).EQ.1).AND.(YKO.EQ.1)
     1.AND.(MOD(XK,7).NE.1).AND.(MOD(XKO,5).EQ.2)) GOTO 777
700   IF ((YK.EQ.3).AND.(IABS(XK-XKO).EQ.2).AND.
     1(YKO.EQ.3).AND.(MOD(XKO,7).EQ.1)) GOTO 705
      GOTO 710
705   IF ((SR.EQ.3).OR.(TRICK.EQ.1)) GOTO 777
C
C     THE FOLLOWING TESTS ARE ONLY FOR POSITIONS WHERE WHITE
C     QUEENS FIRST.
C
710   IF (TRICK.EQ.0) GOTO 760
      IF ((YKO.EQ.1).AND.(IABS(XKO-XP2).GT.2).AND.
     1(IABS(XP1-XKO).GT.1).AND.(MOD(XKO,7).GT.2)) GOTO 720
      GOTO 730
720   IF ((XK.NE.XP1).AND.(IABS(XP1-XKO).LT.IABS(XP1-XP2))
     1.AND.(MOD(XP1,6).LE.3)) GOTO 780
      RC=0
      IF (XP1-XK.GT.0) RC=2*(XP1-XK)
      IF ((MOD(XP1,7).EQ.1).AND.(IABS(XP1-XP2).LT.IABS(XP1-XKO))
     1.AND.(XK+YK+RC.NE.XP1+8)) GOTO 780
730   RC=0
      IF (XKO-XP2.GT.0) RC=2*(XKO-XP2)
      IF ((XKO+YKO.EQ.XP2+1+RC).AND.(IABS(XKO-XP2).GT.1)) GOTO 740
      GOTO 760
740   IF ((IABS(XP1-XKO).LT.IABS(XK-XKO)).OR.
```

```
     1(MOD(XP2,7).EQ.1)) GOTO 750
745   RC=0
      IF (XK-XP2.GT.0) RC=2*(XK-XP2)
      IF (XK+YK.EQ.XP2+1+RC) GOTO 760
      IF (MOD(XP2,7).EQ.1) GOTO 780
      IF ((XK.EQ.XP1).AND.(XK+YK.GT.XP2+1+RC)) GOTO 760
750   IF (MOD(XKO,7).LE.2) GOTO 760
      IF ((IABS(XP1-XKO).LT.IABS(XP1-XP2)).AND.
     1(IABS(XP1-XKO).GT.1)) GOTO 780
      IF ((MOD(XP2,7).EQ.1).AND.((YK.NE.8).OR.
     1(IABS(XP1-XP2).GT.IABS(XK-XP2)))) GOTO 745
C
C     WHITE HAS AT LEAST A DRAW.
C
760   PCHK=1
      IF (REV.EQ.0) PCHK=3
      GOTO 1111
770   IF (SWAT.EQ.1) GOTO 810
      IF (SR.EQ.2) GOTO 780
      IF (TRICK.NE.1) GOTO 775
      IF (XK.EQ.XKO)  GOTO 775
      IF ((YK.NE.8).OR.(IABS(XP1-XP2).LT.IABS(XK-XP1)).OR.
     1(IABS(XP1-XP2).LT.IABS(XK-XP2))) GOTO 780
775   IF (XKO.EQ.XP2) GOTO 630
      GOTO 670
C
C
C     IF BLACK HAS A BP OR RP HE MAY STILL BE ABLE  TO DRAW,
C     EVEN  THOUGH QUEENING THE PAWN LOSES.
C
C
780   IF (SWAT.EQ.1) GOTO 810
      IF ((MOD (XP2,7).EQ.1).OR.(MOD(XP2,3).EQ.0)) GOTO 790
      GOTO 777
C
C     FIRST WE CONSIDER POSITIONS WHERE WHITE QUEENS AFTER BLACK
C     BUT WITH CHECK!
C
790   IF (PR.NE.DT) GOTO 820
      IF ((SR.EQ.3).AND.(IABS(XKO-XP2).GT.2)) GOTO 777
      IF (SR.EQ.3) GOTO 910
      IF (MOD(XP2,3).EQ.0) GOTO 800
      IF ((XKO.EQ.XP2).AND.(YKO.EQ.4))  GOTO 910
      GOTO 777
800   IF (XKO.NE.XP2) GOTO 760
      IF (YKO.EQ.6) GOTO 777
      RC=2
      IF (YKO.EQ.5) RC=3
      IF (DIST(XK,YK,XP2,1).LE.RC) GOTO 777
      IF ((YK.EQ.4).AND.(MOD(XK,7).EQ.1).AND.
     1(IABS(XK-XKO).EQ.2)) GOTO 777
      GOTO 1111
810   IF (MOD(XP2,3).EQ.0) GOTO  760
      IF ((SR.EQ.-3).AND.(YKO.EQ.3).AND.(YK.NE.5))  GOTO 760
      GOTO 777
C
C     HERE WHITE QUEENS FIRST!
C
```

```
820    IF (MOD(XP2,3).EQ.0) GOTO  830
       IF (DIST(XKO,YKO,XP2,1).GT.3) GOTO 777
       IF (YKO.EQ.4) GOTO 905
       GOTO 910
830     RC=0
        IF ((SR.EQ.0).AND.(XK.EQ.XP1)) RC=1
       IF ((XKO.EQ.XP2).AND.(YKC.EQ.5).AND.(SR.LE.0)) GOTO 840
       IF ((SR.EQ.3).AND.(YKO.EQ.5).AND.(MOD(XKO,5).EQ.2)
      1.AND.(XK.EQ.XP1).AND.(YK.GE.5)) GOTO 760
840    IF (DIST(XKO,YKO,XP2,1).GT.3+RC) GOTO 777
       IF (DIST(XK,YK,XP2,1).LE.2) GOTO  777
       IF ((MOD(XKO,6).LE.3).AND.(IABS(XKO-XP2).LE.2)) GOTO 935
       IF (((XP1.EQ.XP2).OR.(XP1.EQ.XKO)).AND.(YK.EQ.5).AND.
      1(IABS(XKC-XP2).EQ.2).AND.(IABS(XK-XKO).EQ.1)) GOTO 760
       GOTO 940
C
C      IF WHITE'S KING IS CLOSE ENOUGH TO BLACK'S ROOKPAWN OR
C      BISHOP PAWN, AFTER BOTH SIDES QUEEN WHITE WILL MATE.
C
900    IF (MOD(XP2,3).EQ.0)  GOTO 920
       IF (DIST(XKO,YKO,XP2,1).GT.2) GOTO 777
       IF (YKO.NE.3) GOTO 910
905    RC=0
       IF (XK-XP2.GT.0) RC=2*(XK-XP2)
       IF (XK+YK.EQ.XP2+1+RC) GOTO 910
       GOTO 777
910    IF ((YK.EQ.5).AND.(IABS(XK-XP2).EQ.4)) GOTO 760
       IF (DIST(XK,YK,XP2,1).LE.4) GOTO 777
       IF ((IABS(XP1-XP2).EQ.7).AND.(PR-DT.EQ.1)) GOTO 1180
       GOTO 760
920    RC=1
       IF (XK.EQ.XP1) RC=2
       IF (DIST(XKO,YKO,XP2,RC).GT.2) GOTO 777
930    IF (DIST(XK,YK,XP2,1).LE.2) GOTO 777
       IF (MOD(XKO,7).GT.2) GOTO 940
935    XPR=1
       IF (XP2.EQ.6) XPR=8
       IF (DIST(XK,YK,XPR,2).LE.2) GOTO 777
       GOTO 760
940    XPR=5
       IF (XP2.EQ.6) XPR=4
       IF (DIST(XK,YK,XPR,2).LE.2) GOTO 777
       IF  ((YKO.EQ.2).AND.(YK.EQ.5).AND.(XK.EQ.XKO).AND.
      1(IABS(XP1-XKO).EQ.1).AND.(IABS(XKO-XP2).EQ.1)) GOTO 760
       IF (XPR.EQ.4) XPR=8
       IF (XPR.EQ.5) XPR=1
       IF (DIST(XK,YK,XPR,2).LE.3) GOTO 777
       GOTO 760
C
C
C      BLACK'S KING IS WITHIN THE SQUARE OF WHITE'S PAWN.
C      IF WHITE'S KING CAN SUCCESSFULLY ASSIST HIS PAWN TO QUEEN,
C      WE CHECK WHICH PAWN QUEENS  FIRST.
C
C
1000   IF ((YK.EQ.YKO).AND.(IABS(XK-XKO).EQ.2).AND.(YKO.GT.YP1).AND.
      1(IABS(XKO-XP1).EQ.1).AND.(YP1.LE.4).AND.(QC.EQ.-1)) GOTO 470
       IF ((XKO.EQ.XP1).AND.(YKO.GT.YP1)) GOTO 470
```

```
C
C     IF BOTH KINGS ARE ON DIFFERENT SIDES OF THE WHITE PAWN WE CAN
C     SIMPLY TEST EACH'S DISTANCE FROM THE PAWN AND FROM ITS QUEENING
C     SQUARE. OTHERWISE WE SETUP LOCAL VARIABLES TO CALL ONEPAN. IF
C     WHITE CANNOT WIN WITH KING AND PAWN VS. LONE KING, BLACK HAS AT
C     LEAST A DRAW. BUT A WINNING SCORE IS DECEPTIVE SINCE THE CONCEPT
C     OF 'ZUGSWANG' HAS NO VALUE WHEN BLACK CAN SIMPLY PUSH HIS OWN
C     PAWN TO QUEEN!
C
      YNOT=SCORE(PLY+3)
      EX=0
      RC=0
      IF (QC.EQ.1) RC=1
      IF (((XP1-XK)*(XP1-XKO).LE.0).OR.(YP1.EQ.7)) GOTO 1010
      IF ((YK.LE.5).OR.(YP1.LE.4)) GOTO 1010
      ABC=WHITEX(1)
      DEF=WHITEY(1)
      GHI=WHITEP(2)
      JKL=WHITEX(2)
      MNO=WHITEY(2)
      PQR=BLACKX(1)
      STU=BLACKY(1)
      VWX=C
      YZA=QC
      WHITEX(1)=XK
      WHITEY(1)=YK
      WHITEP(2)=1
      WHITEX(2)=XP1
      WHITEY(2)=YP1
      BLACKX(1)=XKO
      BLACKY(1)=YKO
      C=QC
      MATERB=0
      CALL ONEPAN
      MATERB=-1
      WHITEX(1)=ABC
      WHITEY(1)=DEF
      WHITEP(2)=GHI
      WHITEX(2)=JKL
      WHITEY(2)=MNO
      BLACKX(1)=PQR
      BLACKY(1)=STU
      C=VWX
      QC=YZA
      IF (IABS(SCORE(PLY+3)).LT.20)  EX=1
      SCORE(PLY+3)=YNOT
      IF (EX.EQ.1) GOTO 470
      IF ((YK.GE.7).OR.(YKO.NE.8)) GOTO 1005
      IF ((XK.EQ.XKO).AND.(YP1.EQ.5)) GOTO 470
      TF=0
      IF (QC.EQ.1) TF=1
      IF (IABS(XK-XP1).LT.IABS(XKO-XP1)+TF-1) GOTO 1005
      MOVE(700)=1
      GOTO 470
1005  IF ((DIST(XKO,YKO,XP1,10)+RC.NE.3).OR.
     1(YP1.NE.5).OR.(YK.NE.6)) GOTO 1010
      IF (QC.EQ.1) DT=DT+2
      IF ((QC.EQ.-1).AND.(YKO.EQ.8)) DT=DT+1
```

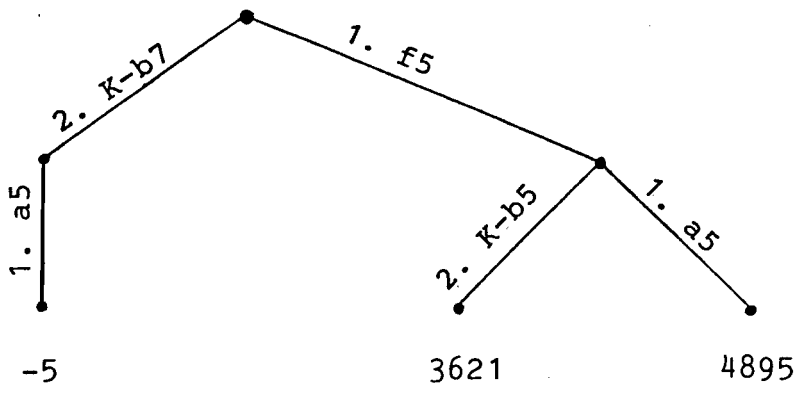
```
C
      DRAWC3=0
      IF (MATERW+IABS(MATERB).GT.2) RETURN
      CALL GENERT
      IF (MOVE(50*PLY+1).NE.0) RETURN
      DRAWC3=1
      IF ((MATERW.NE.2).AND.(MATERB.NE.-2)) GOTO 50
      IF (C.EQ.1) GOTO 30
      IF (BLACKY(1).NE.8) GOTO 50
      DO 10 ABC=1,8
      IF (BOARD(ABC,7).NE.1) GOTO 10
      IF (IABS(ABC-BLACKX(1)).EQ.1) GOTO 60
10    CONTINUE
      GOTO 50
30    IF (WHITEY(1).NE.1) GOTO 50
      DO 40 ABC=1,8
      IF (BOARD(ABC,2).NE.-1) GOTO 40
      IF (IABS(ABC-WHITEX(1)).EQ.1) GOTO 70
40    CONTINUE
50    SCORE(PLY+3)=0
      RETURN
60    SCORE(PLY+3)=6000-200*PLY-100*(1-C)
      RETURN
70    SCORE(PLY+3)=-6000+200*PLY+100*(1+C)
      RETURN
      END
```

APPENDIX 2

Here we present two sample trees from problem nos. 2 and 25 of Chapter III. The chess pieces are internally represented as follows: White king = 6, Black king = -6, White pawn = 1, Black pawn = -1. A chess move is stored as a six digit number where the first digit represents the chess piece that moved, the next four digits, in pairs, represent the x-y coordinates of the new and old squares respectively of the piece that moved, and the last digit represents the type of move (0 for regular, 1 for capture, 2 for en passant and 3 for queening). We now present an example of a small tree to explain how to interpret the data.

4895	1	135340	1	-185860
3621	1	135340	2	-674750
-5	2	677880	1	-185860

In each line the number located in the first column is the score at the terminal node; in the second column it refers to the number of the move in the move list at the first ply depth; in the third column it is the move itself (in this case it is White's move); in the fourth it is the number of the move in the list at the second ply depth; and finally, in the fifth column it is Black's move. The conventional format for the tree above follows:



In the sample trees that follow D refers to the maximum ply depth; the same position was tested at different D levels. The real job time and the principal continuation are also displayed after each tree.

```
BLACK SIDE
 0  0  0  0 -6  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0 -1  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  1  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0

     WHITE SIDE


   -5  1 145440
   -1  2 665640
   -1  3 654640

THE PRINCIPAL CONTINUATION IS
60165640-

                                                    SCORE=     -1

D= 1
JOB TIME=   0.03SECONDS


   -4  1 145440  1-637480
   -4  1 145440  2-657480
   -4  1 145440  3-647480
    0  2 665640  1-637480
   -6  2 665640  2-145460
    8  3 654640  1-145460  1 645541
   -6  3 654640  1-145460  2 665540
   -4  3 654640  1-145460  3 656540  1-637480
   -6  3 654640  1-145460  4 664540
   -6  3 654540  1-145460  5 663540
   -6  3 654640  1-145460  6 643540
   -6  3 654640  1-145460  7 653540
    0  3 654640  2-637480
    0  3 654640  3-657480
    0  3 654640  4-647480

THE PRINCIPAL CONTINUATION IS
60154640--60137480-

                                                    SCORE=      0

D= 2
JOB TIME=   0.17SECONDS


   -4  1 145440  1-637480  1 665540
   -4  1 145440  1-637480  2 654640
   -4  1 145440  2-657480  1 665640
   -4  1 145440  3-647480  1 655640
   -4  2 665640  1-637480  1 145440
    1  2 665640  1-637480  2 656650
    0  2 665640  1-637480  3 666650
    0  2 665640  1-637480  4 654650
    0  2 665640  1-637480  5 664650
    8  2 665640  2-145460  1 656650  1-637480  1 645561
    7  2 665640  2-145460  1 656650  2-657480  1 645561
    8  2 665640  2-145460  1 656650  3-658480  1 645561
   -4  2 665640  3-657480  1 145440
  -11  2 665640  3-657480  2 676650
  -11  2 665640  3-657480  3 675650
  -11  2 665640  3-657480  4 674650
    0  2 665640  3-637480  5 654650
    0  2 665640  3-657480  6 664650
    0  2 665640  4-647480  1 654650
   -4  3 654540  1-657480  1 145440
    0  3 654640  1-657480  2 645540
    0  3 654540  1-657480  3 665540
    0  3 654640  1-657480  4 664340

THE PRINCIPAL CONTINUATION IS
60165640--60157480- 60154650-

                                                    SCORE=      0

D= 3
JOB TIME=   0.33SECONDS


   -4  1 145440  1-637480  1 655640  1-626370
   -4  1 145440  1-637480  1-665640  2-627370
   -4  1 145440  1-637480  1 665640  3-647370
   -4  1 145440  1-637480  2 654640  1-635370
   -4  1 145440  2-657480  1 665640  1-647570
    0  1 145440  2-657480  1 665640  2-657570
   -4  1 145440  3-647480  1 665640  1-637470
   -4  1 145440  3-647480  1 665640  2-657470
   -4  2 665640  1-637480  1 145440  1-626770
 4397  2 665540  1-637480  2 656650  1-626770
    7  2 665640  1-637480  2 656650  2-145460  1 645561
   -5  2 665640  1-637480  2 656650  2-145460  2 657560
   -5  2 665640  1-637480  2 656650  2-145460  3 666560
   -5  2 665640  1-637480  2 656650  2-145460  4 665560
   -5  2 665640  1-637480  2 656650  2-145460  5 667560
 3997  2 665640  1-637480  2 656650  2-145460  5 655560  1-626370
   -3  2 665640  1-637480  2 656650  2-145460  5 655560  2-626370
    2  2 665640  1-637480  2 656650  3-636370
 4396  2 665340  1-637480  2 655650  4-627370
 4395  2 665640  1-637480  2 656650  5-664370
 4395  2 665640  1-637480  2 656650  6-624370
 4395  2 665640  1-637480  2 656650  7-638370
    1  2 665640  1-637480  3 666650  1-634370
    1  2  65640  1-637480  4 654650  1-636370
    0  2 665640  1-637480  5 664650
    8  2 665640  2-145460  1 656650  1-637480  1 645561
    7  2 665640  2-145460  1 656650  2-657480  1 645561
    8  2 665640  2-145460  1 656650  3-658480  1 645561
   -4  2 665640  3-657480  1 145440  1-647570
   -1  2 665640  3-647480  2 676650  1-647570
  -10  2 665640  3-647480  2 676650  2-657570
  -10  2 665640  3-647480  3 675650  1-647470
  -10  2 665640  3-647480  4 674650  1-647570
    1  2 665640  3-647480  5 654650  1-657570
    1  2 665640  3-647480  5 654650  2-657470
    7  2 665640  3-647480  5 654650  3-145460  1 645541
    0  2 665640  3-647480  5 664650
 4196  2 665640  4-647480  1 654650  1-145460
    1  2 665640  4-647480  1 654650  2-647470
    1  2 665340  4-647480  1 654650  3-657470
   -4  3 654640  1-657480  1 145440  1-647570
    0  3 654640  1-657480  2 645540
```

```
THE PRINCIPAL CONTINUATION IS
60165640--60157440- 60154650--60150570-

                                                    SCORE=      1

D= 4
JOB TIME=   0.055SECONDS


-3303   1  145440   1-637480   1  665640   1-626370   1  665650
   -4   1  145440   1-637480   1  665640   1-626370   2  665650
   -4   1  145440   1-637480   1  665640   1-626370   3  654650
   -4   1  145440   1-637480   1  665640   1-626370   4  664650
 4396   1  145440   1-637480   1  665640   2-627370   1  665650
 4396   1  145440   1-637480   1  665640   3-647370   1  665650
   -4   1  145440   1-637480   2  654640   1-626370   1  645540
   -4   1  145440   1-637480   2  654640   1-626370   2  644540
   -4   1  145440   1-637480   2  654640   1-626370   3  664340
 4396   1  145440   2-657440   1  665640   1-647570
    0   1  145440   2-657480   1  665640   2-667570
 4396   1  145440   3-647480   1  665640   1-637470   1  665650
 4395   1  145440   3-647480   1  665640   2-657470   1  675650
   -4   2  665640   1-637480   1  665650   1-626370   1  145440
   -4   2  665640   1-637480   1  665650   1-626370   2  657560   1-145460
   -9   2  665640   1-637480   1  665650   1-626370   3  667560
 3997   2  665640   1-637480   1  665650   1-626370   4  656660   1-145460
 4197   2  665640   1-637480   1  665650   1-626370   4  656660   2-625250
    2   2  665640   1-637480   1  665650   1-626370   4  656660   3-636260
    1   2  665640   1-637480   1  665650   1-626370   4  656660   4-637260
 4196   2  665640   1-637480   1  665650   1-626370   4  656650   5-627260
   -9   2  665640   1-637480   1  665650   1-626370   5  665660
 3997   2  665640   1-637480   1  665650   2-145460   1  656660   1-626370
   -3   2  665640   1-637480   1  665650   2-145460   1  656660   2-636370
 3996   2  665640   1-637480   1  665650   2-145460   1  656660   3-627370
    8   2  665640   1-637480   1  665650   2-145460   1  656660   4-648370   1  645561
 3995   2  665640   1-637480   1  665650   2-145460   1  656660   5-628370
    8   2  665640   1-637480   1  665650   2-145460   1  656660   6-638370   1  645561
   -5   2  665640   1-637480   1  665650   2-145460   2  667560
   -5   2  665640   1-637480   1  665650   2-145460   3  657560
   -3   2  665640   1-637480   1  665650   2-145460   4  656660   1-636370
   -5   2  665640   1-637480   1  665650   2-145460   5  665660
    2   2  665640   1-637480   1  665650   3-636370   1  666660
 3996   2  665640   1-637480   1  665650   4-627370   1  656650   1-145460
    1   2  665640   1-637480   1  665650   4-627370   1  656660   2-637270
    2   2  665640   1-637480   1  656650   4-627370   1  656660   3-636270
 4197   2  665640   1-637480   1  665650   4-627370   1  656660   4-626270
 4396   2  665640   1-637480   1  665650   5-647370   1  145440
    0   2  665640   1-637480   1  665650   5-648370
 4395   2  665640   1-637480   1  656650   7-628370   1  145440
 4395   2  665640   1-637480   1  665650   8-638270   1  145440
    7   2  665640   1-637480   2  656650   1-145460   1  645561
   -5   2  665640   1-637480   2  656650   1-145460   2  657560
   -5   2  665640   1-637480   2  656650   1-145460   3  666560
   -5   2  665640   1-637480   2  656650   1-145460   4  665560
   -5   2  665640   1-637480   2  656650   1-145460   5  667560
   -3   2  665640   1-637480   2  656650   1-145460   6  656660   1-636370
 4397   2  665640   1-637480   2  656650   2-626370
-3302   2  665640   1-637480   2  656650   2-636370   1  145440
   -9   2  665640   1-637480   2  656650   3-636370   2  667560
    2   2  665640   1-637480   2  656650   3-636370   3  657560
   -9   2  665640   1-637480   2  656650   3-636370   4  666560
   -9   2  665640   1-637480   2  656650   3-636370   5  665560
 4396   2  665640   1-637480   2  656650   4-627370
 4395   2  665640   1-637480   2  656650   5-648370
 4395   2  665640   1-637480   2  665650   6-628370
 4395   2  665640   1-637480   2  656650   7-638370
-3303   2  665640   1-637480   3  145440   1-626370   1  656650
   -4   2  665640   1-637480   3  145440   1-626370   2  666650
   -4   2  665640   1-637480   3  145440   1-626370   3  654650
   -4   2  665640   1-637480   3  145440   1-626370   4  664650
   -4   2  665640   1-637480   4  654650   1-626370   1  145440
 4197   2  665640   1-637480   4  654650   1-626370   2  645540   1-625250
    0   2  665640   1-637480   4  654650   1-626370   2  643540   2-637260
   -9   2  665640   1-637480   4  654650   1-626370   3  665540
   -9   2  665640   1-637480   4  654650   1-626370   4  664540
    0   2  665640   1-637480   5  664650
    8   2  665640   2-145460   1  656650   1-638430   1  645561
    7   2  665640   2-145460   1  656650   2-637480   1  645561
    8   2  665640   2-145460   1  654650   3-654480   1  645561
 4396   2  665640   3-657480   1  145440   1-647570   1  656650
    0   2  665640   3-657480   1  145440   2-667570
   -6   2  665640   3-657480   2  675650   1-145460   1  677750
   -5   2  665640   3-657480   2  676650   1-145460   2  665740
   -6   2  665640   3-657480   2  675650   1-145460   3  675760
   -6   2  665640   3-657480   3  675650   1-145460   1  674750
   -5   2  665640   3-657480   3  675650   1-145460   2  665750
   -5   2  665640   3-657480   3  675650   1-145460   3  664750
   -6   2  665640   3-657480   3  675650   1-145460   4  674760
   -5   2  665640   3-657480   4  674650   1-145460   1  665740
   -6   2  665640   3-657480   4  674650   1-145460   2  673740
   -6   2  665640   3-657480   4  674650   1-145460   3  675740
   -5   2  665640   3-657480   4  674650   1-145460   4  664740
   -5   2  665640   3-657480   4  674650   1-145460   5  663740
    7   2  665640   3-657480   5  654650   1-145460   1  645541
   -4   2  665640   3-657480   5  654650   2-667570   1  145440
 4197   2  665640   3-657480   5  654650   2-667570   2  645540   1-665660
    0   2  665640   3-657480   5  654650   2-666570   2  645540   2-657660
   -9   2  665640   3-657480   5  654650   2-666570   3  664540
    0   2  665640   3-657480   6  664650
 4397   2  665640   4-647480   1  145440   1-637470   1  656650
 4395   2  665640   4-647480   1  145440   2-657470   1  676650
   -4   3  654640   1-657480   1  145440   1-666570   1  664540
   -4   3  654640   1-657480   1  145440   1-666570   2  664540
    0   3  654640   1-657480   2  645540
   -6   3  654640   1-657480   3  665540   1-145460   1  676650
 3996   3  654640   1-657480   3  665540   1-145460   2  656650   1-647570
 3996   3  654640   1-657480   3  665540   1-145460   2  656660   2-667570
   -6   3  654640   1-657480   3  665540   1-145460   3  676650
   -6   3  654640   1-657480   3  665540   1-145460   4  674650
   -5   3  654640   1-657480   3  665540   1-145460   5  664650
   -4   3  654640   1-657480   3  665540   2-647570   1  145440
    0   3  654640   1-657480   3  665540   2-647570   2  664650
    0   3  654640   1-657480   3  665540   2-647570   3  654650
    0   3  654640   1-657480   3  665540   2-647570   4  664650
    0   3  654640   1-657480   4  664540

THE PRINCIPAL CONTINUATION IS
60165640--60157440- 1024 5440--60167570-

                                                    SCORE=      0

D= 5
JOB TIME=   1.045SECONDS
```

BLACK SIDE

```
 0  0  0  0 -6  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0 -1  0  0  0
 0  0  0  0  0  0  0  0
 0  0  6  0  1  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0
```

WHITE SIDE

```
-3303  1 145440  1-6374H0  1 665640  1-526370  1 656650
 4397  1 145440  1-637480  1 665640  1-626370  2 666650  1-626260
 4396  1 145440  1-6374H0  1 665640  1-626370  2 666650  2-626260
 4396  1 145440  1-6374H0  1 665640  1-626370  2 666650  3-627760
 4396  1 145440  1-6374H0  1 665640  1-526370  2 666650  4-627260
   -4  1 145440  1-6374H0  1 665640  1-626370  3 654650  1-626260
    0  1 145440  1-637480  1 665640  1-626370  4 664650
 4396  1 145440  1-6374H0  1 665640  2-627370  1 666650
 4396  1 145440  1-6374H0  1 665640  3-547370  1 666650
   -4  1 145440  1-6374H0  2 654640  1-626370  1 665540  1-626260
   -4  1 145440  1-637480  2 654640  1-626370  2 664540  1-626260
    0  1 145440  1-6374H0  2 654640  1-626371  3 664540
 4396  1 145440  2-657480  1 665640  1-647570
    0  1 145440  2-6574H0  1 665640  2-647570
-3306  1 145440  2-657480  2 675640  1-647570  1 686750
 4396  1 145440  2-6574H0  2 675640  1-647570  2 665750
   -4  1 145440  2-6574H0  2 675640  2-647570  1 665750  1-637470
 4396  1 145440  2-6574H0  2 675640  2-647570  2 666750
 4397  1 145440  2-6474H0  1 665640  1-637470  1 656650
 4395  1 145440  3-6474H0  1 665640  2-657470  1 676650
   -5  1 145440  3-6474H0  1 665640  2-657470  2 675650  1-647570
   -5  1 145440  3-6474H0  1 665640  2-657470  3 674650  1-647570
   -4  1 145440  3-6474H0  1 665640  2-657470  4 654650  1-647570
    0  1 145440  3-6474H0  1 665640  2-657470  5 664650
   -4  1 145440  3-6474H0  2 654640  1-657470  1 665540  1-647570
   -4  1 145440  3-6474H0  2 654640  1-657470  2 664540  1-647570
    0  1 145440  3-6474H0  2 654640  1-657470  3 664540
 4395  2 665640  1-6474H0  1 145440  1-657470  1 676650
   -5  2 665640  1-6474H0  1 145440  1-657470  2 675650  1-647570
   -5  2 665640  1-6474H0  1 145440  1-657470  3 674650  1-647570
   -4  2 665640  1-6474H0  1 145440  1-657470  4 654650  1-647570
   -4  2 665640  1-6474H0  1 145440  1-657470  5 664650  1-647570
   -4  2 665640  1-6474H0  2 666650  1-145460  1 667660  1-636470
   -3  2 665640  1-6474H0  2 666650  1-145460  2 665660  1-636470
   -4  2 665640  1-6474H0  2 666650  1-145460  3 665660  1-636470
 4196  2 665640  1-647480  3 654650  1-145460
    0  2 665640  1-647480  4 664650
    8  3 654640  1-145460  1 645541
 3997  3 654640  1-145460  2 665540  1-537480  1 656650  1-626370
   -4  3 654640  1-145460  2 665540  1-637480  2 666650  1-626370
 3997  3 654640  1-145460  2 665540  1-637480  3 656650  1-626370
   -4  3 654640  1-145460  2 665540  1-637480  4 664650  1-626370
    7  3 654540  1-145460  3 655540  1-637480  1 645551
 3997  3 654640  1-145460  3 655540  1-637480  2 656550
   -4  3 654640  1-145460  3 655540  1-637480  3 666550
   -4  3 654640  1-145460  3 655540  1-637480  4 665550  1-626370
   -4  3 654540  1-145460  3 655540  1-537480  5 664550
 3997  3 654640  1-145440  4 664540  1-637480  1 655540
   -4  3 654640  1-145460  4 664540  1-637480  2 665640
   -4  3 654640  1-145460  4 664540  1-637480  3 653640  1-626370
   -4  3 654640  1-145460  4 664540  1-537480  4 663640  1-626370
   -4  3 654640  1-145460  5 663540  1-637480  1 664630
   -4  3 654640  1-145460  5 663540  1-537480  2 653630
   -4  3 654640  1-145460  6 643540  1-637480  1 633430  1-626370
   -4  3 654640  1-145460  6 643540  1-637480  2 653430
   -4  3 654640  1-145460  7 653540  1-637480  1 664530
   -4  3 654640  1-145460  7 653540  1-637480  2 643530  1-626370
   -4  3 654640  1-145460  7 653540  1-537480  3 663530
```

THE PRINCIPAL CONTINUATION IS
10245440--60147440- 60165640--60157470- 60176650-

SCORE= 4395

D= 6
JOB TIME= 1.135SECONDS

BLACK SIDE

6	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	-6	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

WHITE SIDE

4593	1	135340	1-185860											
4795	1	135340	2-664750											
4793	1	135340	3-684750											
4794	1	135340	4-674750											
-3505	1	135340	5-665750											
4793	1	135340	6-684750											
-3505	1	135340	7-664750											
4794	1	135340	8-676750											
5	2	677880	1-665750	1	686771									
3994	2	677880	1-665750	2	667770	1-185860	1	135340	1-184850					
4196	2	677880	1-665750	2	667770	1-185860	1	135340	2-654650					
4194	2	677880	1-665750	2	667770	1-185860	1	135340	3-674650					
4195	2	677880	1-665750	2	667770	1-185960	1	135340	4-664650					
-3304	2	677880	1-665750	2	667770	1-185960	1	135340	5-655650					
0	2	677880	1-665750	2	667770	1-185860	2	657670	1-655650	1	135340	1-184850		
-11	2	677880	1-665750	2	667770	1-185860	2	657670	1-655650	2	647570	1-184850		
-3504	2	677880	1-665750	2	667770	1-185860	2	657670	1-655650	3	667570			
-3705	2	677880	1-665750	2	667770	1-185860	3	677670						
-3704	2	677880	1-665750	3	687770									
-3504	2	677880	1-665750	4	135340									
5	2	677880	2-664750	1	686771									
4594	2	677680	3-684750											
6	2	677880	4-674750	1	686771									
-2	2	677880	5-185860	1	135340	1-184850								
4395	2	677880	5-185860	1	135340	2-664750								
4393	2	677880	5-185860	1	135340	3-684750								
4394	2	677880	5-185860	1	135340	4-674750								
-3505	2	677880	5-185860	1	135340	5-665750								
-2	2	677980	5-185860	2	667770	1-184850	1	135340	1-183840	1	136350			
-3708	2	677980	5-185860	2	667770	1-184850	1	135340	1-183840	2	657570			
-3708	2	677880	5-185860	2	667770	1-184850	1	135340	1-183840	3	677570			
-3707	2	677880	5-185860	2	667770	1-184850	1	135340	1-183840	4	656670			
-4	2	677980	5-185860	2	667770	1-184850	1	135340	2-664750					
3993	2	677880	5-185860	2	667770	1-184850	1	135340	3-674750					
3994	2	677880	5-185860	2	667770	1-184850	1	135340	4-665750					
3992	2	677880	5-185860	2	667770	1-184850	1	135340	5-685750					
3992	2	677880	5-185860	2	667770	1-184850	1	135340	6-686750					
-3907	2	677880	5-185960	2	667770	1-184850	2	657670						
-3907	2	677880	5-185860	2	667770	1-184850	3	677670						
-3907	2	677880	5-185860	2	667770	1-184850	4	656670						
-4	2	677880	5-185860	2	667770	2-664750	1	135340	1-184850					
4196	2	677880	5-185860	2	667770	2-664750	1	135340	2-654640					
4194	2	677880	5-185860	2	667770	2-664750	1	135340	3-674640					
4194	2	677880	5-185860	2	667770	2-664750	1	135340	4-675640					
-3304	2	677880	5-185860	2	667770	2-664750	1	135340	5-655640					
4195	2	677880	5-185860	2	667770	2-664750	1	135340	6-665640					
-11	2	677880	5-185860	2	667770	2-664750	2	666670	1-184850	1	135340	1-183840		
3796	2	677880	5-185860	2	667770	2-664750	2	666670	1-184850	1	135340	2-554540		
3794	2	677880	5-185860	2	667770	2-664750	2	666670	1-184850	1	135340	3-674640		
-11	2	677880	5-185860	2	667770	2-664750	2	666670	1-184850	2	655650	1-183840		
-3705	2	677880	5-185860	2	667770	2-664750	2	666670	1-184850	3	676660			
1	2	677880	5-185860	2	667770	2-664750	2	666670	2-653640	1	135340	1-184850		
3995	2	677880	5-185860	2	667770	2-664750	2	666670	2-653640	1	135340	2-664530		
3997	2	677880	5-185860	2	667770	2-664750	2	666670	2-653640	1	135340	3-644530		
3996	2	677880	5-185860	2	667770	2-664750	2	666670	2-653640	1	135340	4-554530		
-1	2	677980	5-185860	2	667770	2-664750	2	666670	3-673640	1	135340	1-184850		
3994	2	677880	5-185860	2	667770	2-664750	2	666670	3-673640	1	135340	2-674730		
3995	2	677880	5-185960	2	667770	2-664750	2	666670	3-673640	1	135340	3-564730		
0	2	677880	5-185960	2	667770	2-664750	2	656670	4-663640	1	135340	1-184850		
3994	2	677880	5-185860	2	667770	2-664750	2	666670	4-653640	1	135340	2-574530		
3996	2	677880	5-185960	2	667770	2-664750	2	666670	4-663640	1	135340	3-654630		
3995	2	677880	5-185860	2	667770	2-664750	2	666670	4-653640	1	135340	4-664630		
3796	2	677880	5-185960	2	667770	2-664750	2	666670	5-654640	1	135340	1-184850		
3997	2	677880	5-185960	2	667770	2-664750	2	656670	5-654640	1	135340	2-644540		
3995	2	677880	5-185860	2	667770	2-664750	2	666670	5-654640	1	135340	3-564540		
-4	2	677880	5-185860	2	667770	2-664750	2	666670	5-654640	1	135340	4-645540	1	136350 1-536451
3997	2	677880	5-185860	2	667770	2-664750	2	656670	5-654640	1	135340	4-645540	1	136350 2-635450
2	2	677880	5-185860	2	667770	2-664750	2	666670	5-654640	1	135340	4-645540	1	136350 3-546450
2994	2	677880	5-185960	2	667770	2-664750	2	666670	6-674640	1	135340			
-11	2	677880	5-185860	2	667770	2-664750	3	656670	1-184850	1	135340	1-183840		
-12	2	677880	5-185860	2	667770	2-664750	3	656670	1-184850	2	646560	1-183840		
-3705	2	677880	5-185860	2	667770	2-664750	3	656670	1-184850	3	565560			
-3707	2	677880	5-185860	2	667770	2-664750	3	656670	1-184850	4	645550			
-3704	2	677880	5-185860	2	667770	2-664750	4	676670						
4194	2	677880	5-185860	2	667770	3-684750	1	135340	1-674840					
4194	2	677880	5-185860	2	667770	3-684750	1	135340	2-674840					
4194	2	677880	5-185860	2	667770	3-684750	1	135340	3-673840					
4193	2	677880	5-185860	2	667770	3-684750	1	135340	4-683840					
4194	2	677880	5-185860	2	667770	4-674750	1	135340						
3994	2	677680	5-185860	2	667770	5-665750	1	135340	1-184850					
4195	2	677880	5-185860	2	667770	5-665750	1	135340	2-654650					
4194	2	677880	5-185960	2	667770	5-665750	1	135340	3-674650					
4195	2	677880	5-185860	2	667770	5-665750	1	135340	4-664650					
-3304	2	677880	5-185960	2	667770	5-665750	1	135340	5-665650					
4194	2	677880	5-185860	2	667770	5-665750	1	135340	6-675650					
0	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	1	135340	1-184850		
3996	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	1	135340	2-644550		
3994	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	1	135340	3-564550		
3995	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	1	135340	4-654550		
-3104	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	1	135340	5-645550		
3994	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	1	135340	5-565550		
-11	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	1-184850		
-2206	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	1	135340 1-535441
3795	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	1	135340 2-634440
3794	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	1	135340 3-654440
3794	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	1	135340 4-655440
0	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	1	135340 5-184850
3795	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	1	135340 6-645440
-6	2	67788	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	2	635470 1-534441
3595	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	2	635470 2-633440
-1	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	2	635470 3-553440
0	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	2	635470 4-643440
-10	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	2	635470 5-184850
-6	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550	2	635470 6-554440
-2	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	2-644550		
-1	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	4-554550		
-2	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	5-665550		
-2	2	677880	5-185860	2	667770	5-665750	2	657670	1-655650	2	647570	6-665550		
3795	2	677880	5-185860	2	667770	5-665750	2	657670	2-654650	1	135340	1-184850		
3996	2	677880	5-185860	2	667770	5-665750	2	657670	2-654650	1	135340	2-644540		
3994	2	677880	5-185860	2	667770	5-665750	2	657670	2-654650	1	135340	3-564540		
3994	2	677880	5-185860	2	667770	5-665750	2	657670	2-654650	1	135340	4-665540		
-3104	2	677880	5-185860	2	667770	5-665750	2	657670	2-654650	1	135340	5-645540		
3995	2	677880	5-185860	2	667770	5-665750	2	657670	2-654650	1	135340	6-655540		
-10	2	677880	5-185860	2	667770	5-665750	2	657670	2-654650	2	555570	1-184850		

```
     0  2 677880   5-185860   2 667770   5-665750   2 657670   2-654650   2 555570   2-644540   1 135340
    -1  2 677880   5-185860   2 667770   5-665750   2 657670   2-654650   2 555570   3-663540
     0  2 677880   5-185860   2 667770   5-665750   2 657670   2-654650   2 555570   4-653540
    -5  2 577880   5-185860   2 667770   5-665750   2 657670   2-654650   2 555570   5-644540   1 135340   1-535441
  3796  2 677880   5-185860   2 667770   5-665750   2 657670   2-654650   2 555570   5-644540   1 135340   2-654440
  3795  2 677880   5-185860   2 667770   5-665750   2 657670   2-654650   2 555570   5-536540   1 135340   3-654440
     1  2 677880   5-185860   2 667770   5-665750   2 657670   2-654650   2 555570   5-644540   1 135340   4-184850
    -6  2 677880   5-185860   2 667770   5-665750   2 657670   2-654650   2 555570   4-664540
   -12  2 677880   5-185860   2 667770   5-665750   2 657670   3-674650   1 656570   1-184850
    -1  2 677880   5-185860   2 667770   5-665750   2 657670   3-674650   1 555570   2-663740
    -8  2 677880   5-185860   2 667770   5-665750   2 657670   3-674650   1 555570   3-593740
    -2  2 677880   5-185860   2 667770   5-665750   2 657670   3-674650   1 656570   4-673740
    -6  2 677880   5-185860   2 667770   5-665750   2 657670   3-674650   1 556570   5-664740
    -8  2 677880   5-185860   2 667770   5-665750   2 657670   3-674650   1 656570   6-594740
    -7  2 677880   5-185860   2 667770   5-665750   2 657670   3-674650   1 556570   7-675740
  3993  2 677880   5-185860   2 667770   5-665750   2 657670   3-674650   2 135340
    -1  2 677880   5-185860   2 667770   5-665750   2 657670   4-654650   1 135340   1-184850
  3995  2 677880   5-185860   2 667770   5-665750   2 657670   4-654650   1 135340   2-654640
  3993  2 577880   5-185860   2 667770   5-665750   2 657670   4-654650   1 135340   3-674540
  3993  2 677880   5-185860   2 667770   5-665750   2 657670   4-654650   1 135340   4-675540
  3995  2 677880   5-185860   2 667770   5-665750   2 657670   4-654650   1 135340   5-555540
  3994  2 677880   5-185860   2 667770   5-665750   2 657670   4-654650   1 135340   6-665640
    -2  2 677880   5-185860   2 667770   5-665750   2 657670   5-194850   1 135340   1-193840
  3795  2 577880   5-185860   2 667770   5-665750   2 657670   5-194850   1 135340   2-654650
  3793  2 677880   5-185860   2 667770   5-665750   2 657670   5-194850   1 135340   3-674650
    -1  2 677880   5-185860   2 667770   5-665750   2 657670   5-194850   1 135340   4-554550
     0  2 677880   5-185860   2 667770   5-665750   2 657670   5-194850   1 135340   5-655650
    -2  2 677880   5-185860   2 667770   5-665750   2 657670   5-194850   1 135340   6-675650
    -2  2 677880   5-185860   2 667770   5-665750   2 657670   5-194850   1 135340   7-576650
     0  2 677880   5-185860   2 667770   5-665750   2 657670   6-675650
    -2  2 677880   5-185860   2 667770   5-665750   2 657670   7-676650   1 135340   1-194850
  3994  2 677880   5-185860   2 667770   5-665750   2 657670   7-676650   1 135340   2-665760
  3993  2 677880   5-185860   2 667770   5-665750   2 657670   7-676650   1 135340   3-675760
  3992  2 577880   5-185860   2 667770   5-665750   2 657670   7-676650   1 135340   4-695760
  3992  2 677880   5-185860   2 667770   5-665750   2 657670   7-676650   1 135340   5-687760
  3993  2 677880   5-185860   2 667770   5-665750   2 657670   7-676650   1 135340   6-677750
  4193  2 677880   5-185860   2 667770   6-686750
 -3906  2 677880   5-185860   3 687770
  4594  2 577880   5-685750
 -3905  3 687880

THE PRINCIPAL CONTINUATION IS
60177880--10295860- 60167770--60164750- 60156570--10284850- 10235340--10283840-

                                        SCORE=   -11

D= 9
JOB TIME=  1.92SECONDS


  4593  1 135340   1-185860
  4795  1 135340   2-664750
  4793  1 135340   3-684750
  4794  1 135340   4-674750
 -3505  1 135340   5-665750
  4793  1 135340   6-685750
 -3505  1 135340   7-666750
  4794  1 135340   8-676750
     5  2 677880   1-665750   1 686771
  3994  2 577880   1-665750   2 667770   1-185860   1 135340   1-184850
  4196  2 677880   1-665750   2 667770   1-185860   1 135340   2-654650
  4194  2 677880   1-665750   2 667770   1-185860   1 135340   3-674650
  4195  2 677880   1-665750   2 667770   1-185860   1 135340   4-654650
 -3304  2 677880   1-665750   2 667770   1-185860   1 135340   5-555550
     0  2 577880   1-665750   2 667770   1-185860   2 657670   1-655650   1 135340   1-184850   1 136350
   -11  2 677880   1-665750   2 667770   1-185860   2 657670   1-655650   1 135340   1-184850   2 647570
 -3305  2 677880   1-665750   2 667770   1-185860   2 657670   1-655650   1 135340   1-184850   3 657570
   -11  2 677880   1-665750   2 667770   1-185860   2 657670   1-655650   2 647570   1-184850   1 135340
 -3507  2 677880   1-665750   2 667770   1-185860   2 657670   1-655650   2 647570   1-184850   2 637470
 -3505  2 577880   1-665750   2 667770   1-185860   2 657670   1-655650   2 647570   1-184850   3 557470
 -3507  2 677880   1-665750   2 667770   1-185860   2 657670   1-655650   2 647570   1-184850   4 636470
 -3504  2 677880   1-665750   2 667770   1-185860   2 657670   1-655650   3 557570
 -3705  2 677880   1-665750   2 667770   1-185860   3 677670
 -3704  2 677880   1-665750   3 687770
 -3504  2 677880   1-665750   4 135340
     5  2 677880   2-664750   1 686771
  4594  2 677880   3-684750
     5  2 577880   4-674750   1 686771
    -2  2 677880   5-185860   1 135340   1-184850
  4395  2 677880   5-185860   1 135340   2-664750
  4393  2 677880   5-185860   1 135340   3-684750
  4394  2 677880   5-185860   1 135340   4-674750
 -3505  2 677880   5-185860   1 135340   5-665750
    -2  2 677880   5-185860   2 667770   1-194850   1 135340   1-193840   1 136350
 -3708  2 677880   5-185860   2 667770   1-184850   1 135340   1-193840   2 557570
 -3708  2 677880   5-185860   2 667770   1-184850   1 135340   1-193840   3 677670
 -3707  2 677880   5-185860   2 667770   1-184850   1 135340   1-193840   4 555670
    -4  2 577880   5-185860   2 667770   1-184850   1 135340   2-664750
  3993  2 677880   5-185860   2 667770   1-184850   1 135340   3-674750
  3994  2 677880   5-185860   2 667770   1-184850   1 135340   4-655750
  3992  2 577880   5-185860   2 667770   1-184850   1 135340   5-685750
  3992  2 677880   5-185860   2 667770   1-184850   1 135340   6-686750
 -3907  2 677880   5-185860   2 667770   1-184850   2 657670
 -3907  2 677880   5-185860   2 667770   1-184850   3 677670
 -3907  2 677880   5-185860   2 657770   1-184850   4 656670
    -4  2 677880   5-185860   2 667770   2-664750   1 135340   1-184850
  4196  2 677880   5-185860   2 667770   2-664750   1 135340   2-654640
  4194  2 677880   5-185860   2 667770   2-664750   1 135340   3-674640
  4195  2 577880   5-185860   2 667770   2-664750   1 135340   4-675640
 -3704  2 677880   5-185860   2 667770   2-664750   1 135340   5-655640
  4195  2 677880   5-185860   2 667770   2-664750   1 135340   6-665640
   -11  2 677880   5-185860   2 667770   2-664750   2 555670   1-184850   1 135340   1-183840   1 136350
   -11  2 677880   5-185860   2 667770   2-664750   2 666670   1-184850   1 135340   1-183840   2 655650
 -3506  2 677880   5-185860   2 667770   2-664750   2 666670   1-184850   1 135340   1-183840   3 676650
  3796  2 677880   5-185860   2 667770   2-664750   2 666670   1-184850   1 135340   2-654640
  3794  2 677880   5-185860   2 667770   2-664750   2 666670   1-184850   1 135340   3-674640
   -11  2 677880   5-185860   2 667770   2-664750   2 666670   1-184850   2 656660   1-183840   1 135340
 -3707  2 677880   5-185860   2 667770   2-664750   2 666670   1-184850   2 656660   1-183840   2 646550
 -3706  2 677880   5-185860   2 667770   2-664750   2 666670   1-184850   2 656660   1-183840   3 656550
 -3707  2 677880   5-185860   2 667770   2-664750   2 666670   1-184850   2 555560   1-183840   4 665560
 -3705  2 677880   5-185860   2 667770   2-664750   2 666670   1-184850   3 575550
     0  2 677880   5-185860   2 667770   2-664750   2 666670   2-653640   1 135340   1-184850
  3995  2 677880   5-185860   2 667770   2-664750   2 666670   2-653640   1 135340   2-664530
  3997  2 677880   5-185860   2 667770   2-664750   2 666670   2-653640   1 135340   3-644530
  3996  2 677880   5-185860   2 667770   2-664750   2 666670   2-653640   1 135340   4-654530
     0  2 677880   5-185860   2 667770   2-664750   2 666670   2-653640   1 135340   1-194850
  3994  2 677880   5-185860   2 667770   2-664750   2 666670   3-673640   1 135340   2-674730
  3995  2 677880   5-185860   2 667770   2-664750   2 666670   3-673640   1 135340   3-664730
     0  2 677880   5-185860   2 667770   2-664750   2 666670   4-663640   1 135340   1-194850
  3994  2 677880   5-185860   2 667770   2-664750   2 666670   4-663640   1 135340   2-674630
  3996  2 677880   5-185860   2 667770   2-664750   2 666670   4-663640   1 135340   3-534530
  3995  2 677880   5-185860   2 667770   2-664750   2 666670   4-663640   1 135340   4-644630
  3795  2 677880   5-185860   2 667770   2-664750   2 666670   5-654640   1 135340   1-194850
  3997  2 577880   5-185860   2 667770   2-664750   2 666670   5-654640   1 135340   2-644340
  3995  2 677880   5-185860   2 667770   2-664750   2 666670   5-654640   1 135340   3-654540
    -4  2 677880   5-185860   2 667770   2-664750   2 666670   5-654640   1 135340   4-645540   1 136350   1-536451
  3997  2 677880   5-185860   2 667770   2-664750   2 666670   5-654640   1 135340   4-645540   1 136350   2-535450
     2  2 677880   5-185860   2 667770   2-664750   2 666670   5-654640   1 135340   4-645540   1 136350   3-546450
 1 137360
  3994  2 677880   5-185860   2 667770   2-664750   2 666670   6-674640   1 135340
   -11  2 677880   5-185860   2 667770   2-664750   3 656670   1-194850   1 135340   1-193840   1 136350
   -12  2 677880   5-185860   2 667770   2-664750   3 656670   1-194850   1 135340   1-193840   2 666560
 -3506  2 677880   5-185860   2 667770   2-664750   3 656670   1-194850   1 135340   1-193840   3 666560
```

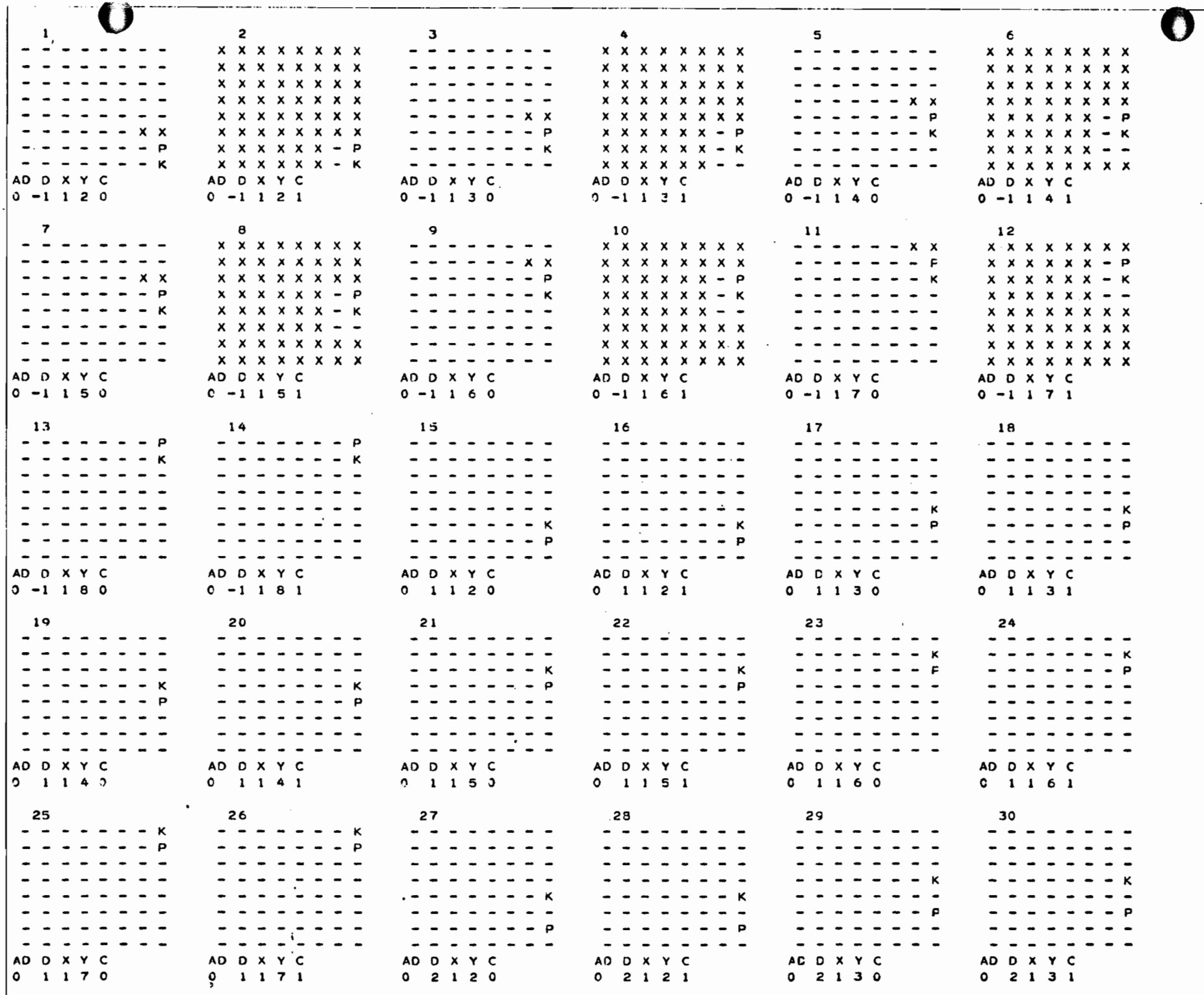
```
-1507  2 577480  5-185860  2 667770  2-664750  3 656670  1-184850  1 135340  1-183840  4 635560
  -12  2 677880  5-185860  2 667770  2-664750  3 656670  1-184850  2 646560  1-183840  1 135340
-3708  2 677880  5-185860  2 667770  2-664750  3 656670  1-184850  2 646560  1-183840  2 636460
-3706  2 577880  5-185860  2 667770  2-664750  3 656670  1-184850  2 646560  1-183840  3 636460
-3708  2 677880  5-185860  2 667770  2-664750  3 656670  1-184850  2 646560  1-183840  4 635460
-3707  2 577880  5-185860  2 667770  2-664750  3 656670  1-184850  2 546560  1-183840  5 545460
-3705  2 677880  5-185860  2 667770  2-664750  3 656670  1-184850  3 566560
-3707  2 677880  5-185860  2 667770  2-664750  3 656670  1-184850  4 535350
-3704  2 577880  5-185860  2 667770  2-664750  4 676670
  4194  2 677880  5-185860  2 667770  3-674750  1 135340  1-675840
  4194  2 677880  5-185860  2 667770  3-674750  1 135340  2-674840
  4194  2 677880  5-185860  2 667770  3-674750  1 135340  3-673840
  4193  2 677880  5-185860  2 667770  3-674750  1 135340  4-673840
  4194  2 577880  5-185860  2 667770  4-674750  1 135340
  3994  2 677880  5-185860  2 667770  5-665750  1 135340  1-134850
  4195  2 577880  5-185860  2 667770  5-665750  1 135340  2-654650
  4194  2 677880  5-185860  2 667770  5-665750  1 135340  3-674650
  4195  2 677880  5-185860  2 667770  5-665750  1 135340  4-664650
-3304  2 577880  5-185860  2 667770  5-665750  1 135340  5-655650
  4194  2 677880  5-185860  2 667770  5-665750  1 135340  6-675650
     0  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  1 135340  1-184850  1 136350
  3996  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  1 135340  2-644550
  3994  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  1 135340  3-664550
  3995  2 577880  5-185860  2 667770  5-665750  2 657670  1-655650  1 135340  4-654550
-3104  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  1 135340  5-645550
  3994  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  1 135340  6-655550
   -11  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  1-184850  1 135340
-2206  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  1 135340  1-635441
  3795  2 577880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  1 135340  2-634440
  3794  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  1 135340  3-654440
  3794  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  1 135340  4-655440
    -2  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  1 135340  5-184850
  3795  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  1 135340  6-645440
    -6  2 577880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  2 646470  1-634441
  3595  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  2 646470  2-633440
    -1  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  2 646470  3-653440
    -1  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  2 646470  4-643440
1 135340
   -10  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  2 646470  5-184850
1 135340
    -6  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  2-644550  2 646470  6-654440
   -12  2 577880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  3-664550  1 636470
    -2  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  3-664550  2 135340
  3794  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  4-654550  1 135340
     0  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  5-665550
    -2  2 677880  5-185860  2 667770  5-665750  2 657670  1-655650  2 647570  6-656550  1 135340
  3795  2 577880  5-185860  2 667770  5-665750  2 657670  2-654650  1 135340  1-184850
  3996  2 677880  5-185860  2 667770  5-665750  2 657670  2-654650  1 135340  2-644540
  3994  2 677880  5-185860  2 667770  5-665750  2 657670  2-654650  1 135340  3-664540
  3994  2 577880  5-185860  2 667770  5-665750  2 657670  2-654650  1 135340  4-665540
-3104  2 677880  5-185860  2 667770  5-665750  2 657670  2-654650  1 135340  5-645540
  3995  2 677880  5-185860  2 667770  5-665750  2 657670  2-654650  1 135340  6-655540
   -10  2 677880  5-185860  2 667770  5-665750  2 657670  2-654650  2 656570  1-184850  1 135340
     0  2 677880  5-185860  2 667770  5-665750  2 657670  2-654650  2 555570  2-643540  1 135340
    -1  2 577880  5-185860  2 667770  5-665750  2 657670  2-654650  2 656570  3-663540  1 135340
     0  2 677880  5-185860  2 667770  5-665750  2 657670  2-654650  2 555570  4-653540  1 135340
    -5  2 577880  5-185860  2 667770  5-665750  2 657670  2-654650  2 555570  5-644540  1 135340  1-535441
  3796  2 677880  5-185860  2 667770  5-665750  2 657670  2-654650  2 654570  5-644540  1 135340  2-634440
  3795  2 677880  5-185860  2 667770  5-665750  2 657670  2-654650  2 555570  5-644540  1 135340  3-654440
     0  2 677880  5-185860  2 667770  5-665750  2 657670  2-654650  2 655570  5-644540  1 135340  4-184850
    -1  2 677880  5-185860  2 667770  5-665750  2 657670  2-654650  2 555570  6-664540  1 135340
  3993  2 577880  5-185860  2 667770  5-665750  2 657670  3-674650  1 135340
    -4  2 677880  5-185860  2 667770  5-665750  2 657670  4-664650  1 135340  1-184850
  3995  2 677880  5-185860  2 667770  5-665750  2 657670  4-664650  1 135340  2-654540
  3993  2 577880  5-185860  2 667770  5-665750  2 657670  4-664650  1 135340  3-674640
  3993  2 677880  5-185860  2 667770  5-665750  2 657670  4-664650  1 135340  4-675640
  3995  2 677880  5-185860  2 667770  5-665750  2 657670  4-664650  1 135340  5-655640
  3994  2 677880  5-185860  2 667770  5-665750  2 657670  4-664650  1 135340  6-665640
    -2  2 677880  5-185860  2 667770  5-665750  2 657670  5-184850  1 135340  1-183840  1 136350
  3795  2 577880  5-185860  2 667770  5-665750  2 657670  5-184850  1 135340  2-654650
  3793  2 677880  5-185860  2 667770  5-665750  2 657670  5-184850  1 135340  3-674650
    -4  2 677880  5-185860  2 667770  5-665750  2 657670  5-184850  1 135340  4-654650
     0  2 677880  5-185860  2 667770  5-665750  2 657670  5-184850  1 135340  5-655650  1 136350
    -2  2 677880  5-185860  2 667770  5-665750  2 657670  5-184850  1 135340  6-675650
    -2  2 677880  5-185860  2 667770  5-665750  2 657670  5-184850  1 135340  7-676650
     0  2 677880  5-185860  2 667770  5-665750  2 657670  6-675650
    -2  2 677880  5-185860  2 667770  5-665750  2 657670  7-676650  1 135340  1-184850
  3994  2 677880  5-185860  2 667770  5-665750  2 657670  7-676650  1 135340  2-665750
  3993  2 677880  5-185860  2 667770  5-665750  2 657670  7-676650  1 135340  3-675760
  3992  2 577880  5-185860  2 667770  5-665750  2 657670  7-676650  1 135340  4-686750
  3992  2 677880  5-185860  2 667770  5-665750  2 657670  7-676650  1 135340  5-687760
  3993  2 677880  5-185860  2 667770  5-665750  2 657670  7-676650  1 135340  6-677750
  4193  2 677880  5-185860  2 667770  6-686750
-3906  2 677880  5-185860  3 687770
  4594  2 677880  6-685750
-3905  3 687880

THE PRINCIPAL CONTINUATION IS
60177880--10285850- 60167770--60164750- 60155570--10284850- 10235340--10283840- 10236350-

                                    SCORE=   -11

D= 9
JOB TIME=  2.10SECONDS
```

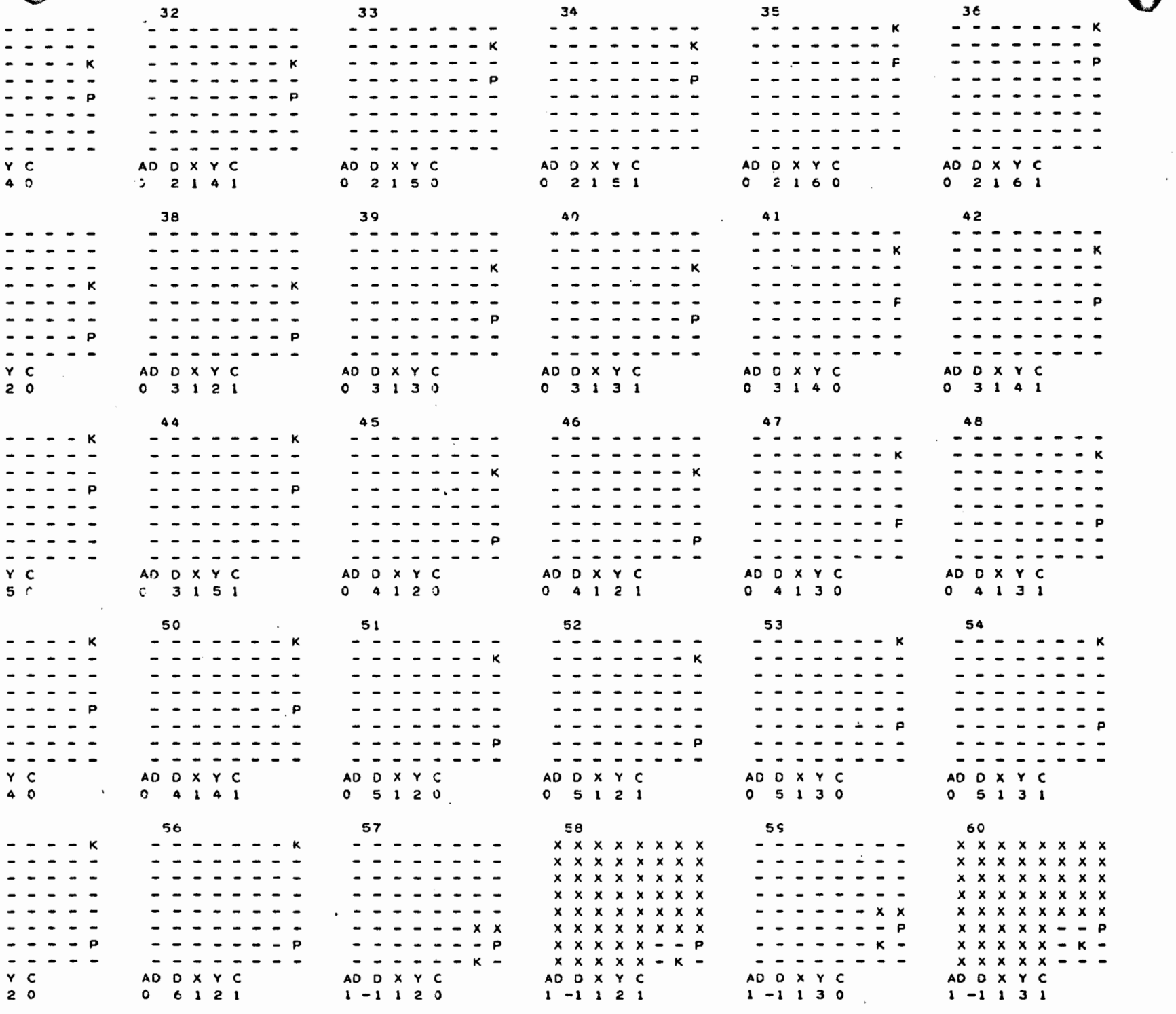
APPENDIX 3

We wrote a 'drive' program to double-check subroutine ONEPAN. This was done by setting up a position with the Black king and the White pawn on certain squares and then testing each of the remaining squares with subroutine ONEPAN to determine whether or not the position would be won for White with the White king occupying that square. If the position was won the 'drive' program would print an x on the square; otherwise it would be left empty. Finally we tested the program in every possible situation with the Black king relative to the White pawn (except those cases easily disposed of by the 'rule of the square' test). The 'drive' program and a sample of the positions generated by it are listed here. It was much simpler to verify subroutine ONEPAN's accuracy by checking these diagrams. The masks discussed in Chapter II represent the chess information locked in these diagrams more concisely; one mask 'covers' several different diagrams exhibiting the same structure ie. a many-one relationship (see 2.1). If all the diagrams are correct then ONEPAN is correct. In the sample figures that follow some additional information is supplied. On top of each is a number indicating its position in the list of diagrams. On the bottom, X and Y are the coordinates of the White pawn P in the figure (with the origin in the bottom right-hand corner), AD and D, defined as W and H respectively in Chapter II, describe the position of the Black king K in relation to the pawn, and finally C refers to the colour of the player on the move (1 for White and 0 for Black).

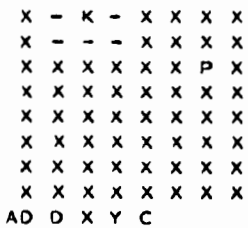
```
      IMPLICIT INTEGER (A-Z)
      COMMON /MLIST/ MOVE(700),PLY,SCOLOR,C,MOVEP,D
      COMMON /DLIST/ MP(20),SCORE(20),CAPTY(20),BOARD(8,8)
      COMMON /SKORE/ MATERW,MATERB
      COMMON /DRAW/ DRAWT,DRAWC1,DRAWC2,DRAWC3,KONTA,KONTS
      COMMON /PLIST/ WHITEP(11),WHITEX(11),WHITEY(11)
      COMMON /PLISTB/ BLACKP(11),BLACKX(11),BLACKY(11)
      COMMON /PLISTZ/ QC,KXO,KYO,KX,KY,PX,PY,PX1,PY1,
     1KX1,KY1,KX2,KY2
      INTEGER*2 B(8,8,6),KING,PAWN,EX,SLASH,BLANK
      DIMENSION E(6,6)
      DATA  E/36*0/
      DATA KING,PAWN,EX,SLASH,BLANK/1HK,1HP,1HX,1H-,1H /
C
C     THIS PROGRAM CHECKS WHETHER ONEPAN IS CORRECT, BY
C     PRINTING BOARD POSITIONS CASE BY CASE. ON EACH BOARD
C     THE BLACK KING IS REPRESENTED BY 'K', THE WHITE PAWN
C     BY 'P', EMPTY SQUARES BY '-' AND WINNING SQUARES FOR
C     THE WHITE KING BY 'X'. THE PROGRAM GOES THROUGH 6 DO LOOPS
C     WHICH INCREMENT THE WHITE PAWN'S, THE BLACK KING'S AND
C     THE WHITE KING'S X-Y COORDINATES.IT CONSIDERS BOTH BLACK
C     AND WHITE TO PLAY IN EVALUATING EACH POSITION. (C IS
C     ALTERNATELY PLUS OR MINUS ONE FOR WHITE OR BLACK TO PLAY
C     RESPECTIVELY). B(8,8,6) STORES SIX BOARDS AT A TIME FOR
C     PRINTING SIMULTANEOUSLY. E(6,6) STORES INCREMENTED VALUES
C     OF THE PIECES FOR EACH OF THE SIX BOARDS.
C
      MATERW=1
      T=1
      PLY=0
      MATERB=0
      TT=1
      BS=0
      WRITE(6,5000)
      DO 1000 I=1,8
      DO 900 J=1,8
      DO 800 K=1,8
      DO 700 L=2,8
      IF ((IABS(J-I).EQ.0).AND.(K.EQ.2)) GOTO 800
      WHITEP(2)=1
      WHITEX(2)=I
      WHITEY(2)=L
      BLACKX(1)=J
      IF (L+K-2.GT.8) GOTO 800
      BLACKY(1)=L+K-2
      C=-1
      GOTO 100
50    T=T+1
      TT=MOD(T,6)
      IF (TT.EQ.0) TT=6
      IF (TT.EQ.1) GOTO 650
75    C=C+2
      IF (C.GT.1) GOTO 700
100   DO 150 D=1,8
      DO 150 G=1,8
```

```
      B(D,G,TT)=BLANK
150   CONTINUE
      DO 600 M=1,8
      DO 400 N=1,8
      IF ((M.EQ.WHITEX(2)).AND.(N.EQ.WHITEY(2))) GOTO 260
      IF ((M.EQ.BLACKX(1)).AND.(N.EQ.BLACKY(1))) GOTO 300
      IF (DIST(M,N,BLACKX(1),BLACKY(1)).EQ.1) GOTO 240
      WHITEX(1)=M
      WHITEY(1)=N
200   CALL ONEPAN
      IF (SCORE(3).LT.20) B(M,N,TT)=SLASH
      IF (SCORE(3).GT.20) B(M,N,TT)=EX
      GOTO 400
240   B(M,N,TT)=SLASH
      GOTO 400
260   B(M,N,TT)=PAWN
      GOTO 400
300   B(M,N,TT)=KING
400   CONTINUE
600   CONTINUE
      E(1,TT)=T
      E(2,TT)=IABS(J-I)
      E(3,TT)=K-2
      E(4,TT)=I
      E(5,TT)=L
      E(6,TT)=C
      IF (C.EQ.-1) E(6,TT)=0
      GOTO 50
650   BS=BS+1
      WRITE(6,2500) (E(1,P),P=1,6)
      WRITE(6,2000) (((B(9-P,9-Q,R),P=1,8),R=1,6),Q=1,8)
      WRITE(6,3000)
      WRITE(6,4000) ((E(P,Q),P=2,6),Q=1,6)
      DO 675 P=1,6
      DO 675 Q=1,6
      E(P,Q)=0
675   CONTINUE
      GOTO 75
700   CONTINUE
800   CONTINUE
900   CONTINUE
1000  CONTINUE
2000  FORMAT(6(2X,8A2,2X))
2500  FORMAT(1H ,6(I4,16X))
3000  FORMAT(1H ,6('AD D X Y C  ',8X))
4000  FORMAT(1H ,6(I1,1X,I2,1X,I1,1X,I1,1X,I1,10X)/)
5000  FORMAT(1H1)
      STOP
      END
```


1
AD D X Y C
0 -1 1 2 0
2
AD D X Y C
0 -1 1 2 1
3
AD D X Y C
0 -1 1 3 0
4
AD D X Y C
0 -1 1 3 1
5
AD D X Y C
0 -1 1 4 0
6
AD D X Y C
0 -1 1 4 1
7
AD D X Y C
0 -1 1 5 0
8
AD D X Y C
0 -1 1 5 1
9
AD D X Y C
0 -1 1 6 0
10
AD D X Y C
0 -1 1 6 1
11
AD D X Y C
0 -1 1 7 0
12
AD D X Y C
0 -1 1 7 1
13
AD D X Y C
0 -1 1 8 0
14
AD D X Y C
0 -1 1 8 1
15
AD D X Y C
0 1 1 2 0
16
AD D X Y C
0 1 1 2 1
17
AD D X Y C
0 1 1 3 0
18
AD D X Y C
0 1 1 3 1
19
AD D X Y C
0 1 1 4 0
20
AD D X Y C
0 1 1 4 1
21
AD D X Y C
0 1 1 5 0
22
AD D X Y C
0 1 1 5 1
23
AD D X Y C
0 1 1 6 0
24
AD D X Y C
0 1 1 6 1
25
AD D X Y C
0 1 1 7 0
26
AD D X Y C
0 1 1 7 1
27
AD D X Y C
0 2 1 2 0
28
AD D X Y C
0 2 1 2 1
29
AD D X Y C
0 2 1 3 0
30
AD D X Y C
0 2 1 3 1

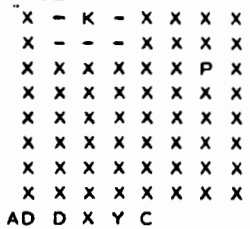

31
AD D X Y C
0 2 1 4 0
32
AD D X Y C
0 2 1 4 1
33
AD D X Y C
0 2 1 5 0
34
AD D X Y C
0 2 1 5 1
35
AD D X Y C
0 2 1 6 0
36
AD D X Y C
0 2 1 6 1
37
AD D X Y C
0 3 1 2 0
38
AD D X Y C
0 3 1 2 1
39
AD D X Y C
0 3 1 3 0
40
AD D X Y C
0 3 1 3 1
41
AD D X Y C
0 3 1 4 0
42
AD D X Y C
0 3 1 4 1
43
AD D X Y C
0 3 1 5 0
44
AD D X Y C
0 3 1 5 1
45
AD D X Y C
0 4 1 2 0
46
AD D X Y C
0 4 1 2 1
47
AD D X Y C
0 4 1 3 0
48
AD D X Y C
0 4 1 3 1
49
AD D X Y C
0 4 1 4 0
50
AD D X Y C
0 4 1 4 1
51
AD D X Y C
0 5 1 2 0
52
AD D X Y C
0 5 1 2 1
53
AD D X Y C
0 5 1 3 0
54
AD D X Y C
0 5 1 3 1
55
AD D X Y C
0 6 1 2 0
56
AD D X Y C
0 6 1 2 1
57
AD D X Y C
1 -1 1 2 0
58
AD D X Y C
1 -1 1 2 1
59
AD D X Y C
1 -1 1 3 0
60
AD D X Y C
1 -1 1 3 1

931



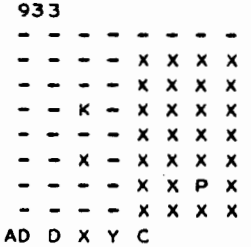
AD	D	X	Y	C
4	2	2	6	0

932



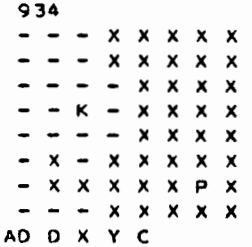
AD	D	X	Y	C
4	2	2	6	1

933



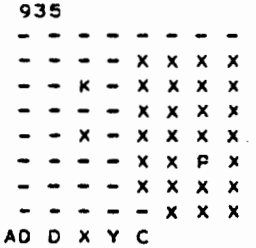
AD	D	X	Y	C
4	3	2	2	0

934



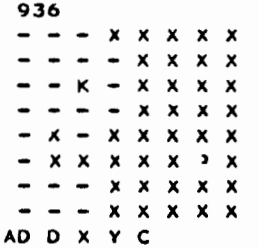
AD	D	X	Y	C
4	3	2	2	1

935



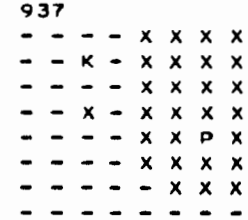
AD	D	X	Y	C
4	3	2	3	0

936



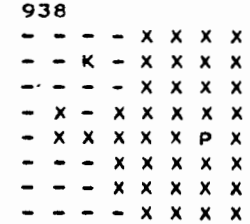
AD	D	X	Y	C
4	3	2	3	1

937



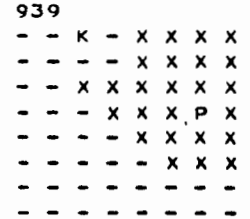
AD	D	X	Y	C
4	3	2	4	0

938



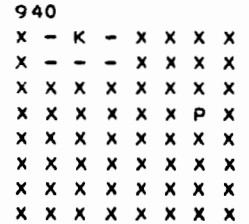
AD	D	X	Y	C
4	3	2	4	1

939



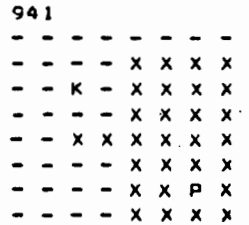
AD	D	X	Y	C
4	3	2	5	0

940



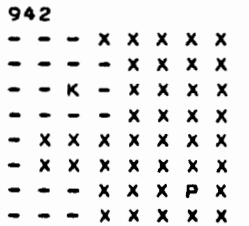
AD	D	X	Y	C
4	3	2	5	1

941



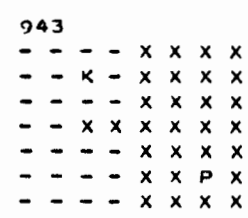
AD	D	X	Y	C
4	4	2	2	0

942



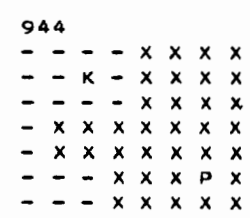
AD	D	X	Y	C
4	4	2	2	1

943



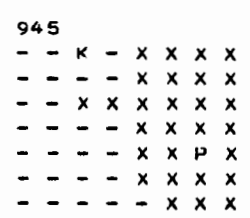
AD	D	X	Y	C
4	4	2	3	0

944



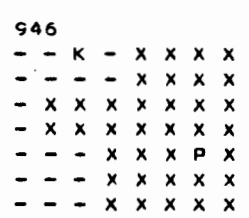
AD	D	X	Y	C
4	4	2	3	1

945



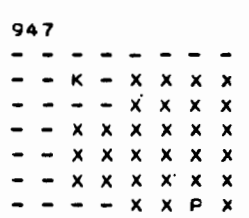
AD	D	X	Y	C
4	4	2	4	0

946



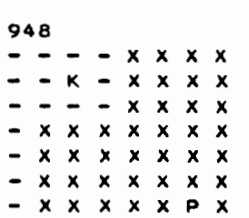
AD	D	X	Y	C
4	4	2	4	1

947



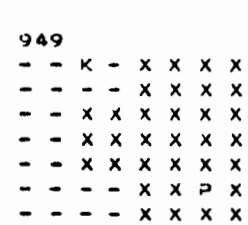
AD	D	X	Y	C
4	5	2	2	0

948



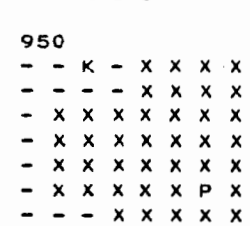
AD	D	X	Y	C
4	5	2	2	1

949



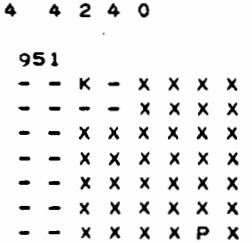
AD	D	X	Y	C
4	5	2	3	0

950



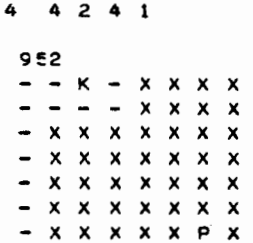
AD	D	X	Y	C
4	5	2	3	1

951



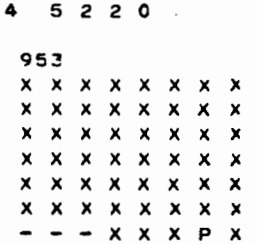
AD	D	X	Y	C
4	6	2	2	0

952



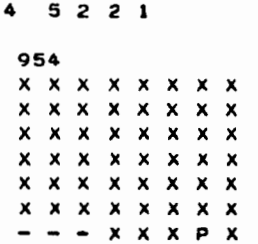
AD	D	X	Y	C
4	6	2	2	1

953



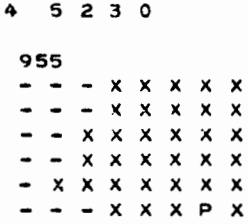
AD	D	X	Y	C
5	-1	2	2	0

954



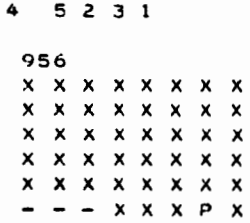
AD	D	X	Y	C
5	-1	2	2	1

955



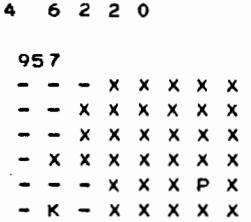
AD	D	X	Y	C
5	-1	2	3	0

956



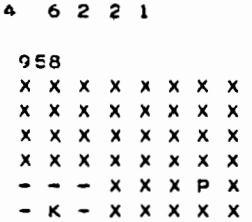
AD	D	X	Y	C
5	-1	2	3	1

957



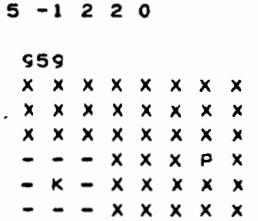
AD	D	X	Y	C
5	-1	2	4	0

958



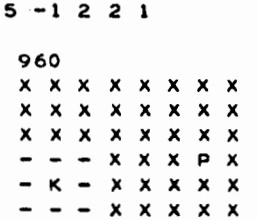
AD	D	X	Y	C
5	-1	2	4	1

959



AD	D	X	Y	C
5	-1	2	5	0

960



AD	D	X	Y	C
5	-1	2	5	1

961

```
X X X X X X X X
X X X X X X X X
- - - X X X P X
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5 -1 2 6 0
```

962

```
X X X X X X X X
X X X X X X X X
- - - X X X P X
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5 -1 2 6 1
```

963

```
X X X X X X X X
X X X X X X X X
- - - X X X P X
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5 -1 2 7 0
```

964

```
X X X X X X X X
- - - X X X P X
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5 -1 2 7 1
```

965

```
- - - X X X P X
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5 -1 2 8 0
```

966

```
- - - X X X P X
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5 -1 2 8 1
```

967

```
- - - X X X X X
- - - X X X X X
- X X X X X X X
- X X X X X X X
- X X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
AD D X Y C
5  0 2 2 0
```

968

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
AD D X Y C
5  0 2 2 1
```

969

```
- - - X X X X X
- - - X X X X X
- - - X X X X X
- - X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
- - - X X X X X
AD D X Y C
5  0 2 3 0
```

970

```
- - X X X X X X
- - X X X X X X
- X X X X X X X
- X X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
- - X X X X X X
AD D X Y C
5  0 2 3 1
```

971

```
- - - X X X X X
- - - X X X X X
- - X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
- - - X X X X X
- - - - X X X X
AD D X Y C
5  0 2 4 0
```

972

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  0 2 4 1
```

973

```
X X X X X X X X
X X X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  0 2 5 0
```

974

```
X X X X X X X X
X X X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  0 2 5 1
```

975

```
X X X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  0 2 6 0
```

976

```
X X X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  0 2 6 1
```

977

```
- - - X X X X X
- K - X X X P X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  0 2 7 0
```

978

```
- - - X X X X X
- K - X X X P X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  0 2 7 1
```

979

```
- K - X X X P X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  0 2 8 0
```

980

```
- K - X X X P X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  0 2 8 1
```

981

```
- - - X X X X X
- - - X X X X X
- - - X X X X X
- X X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
- - - X X X X X
AD D X Y C
5  1 2 2 0
```

982

```
- - X X X X X X
- - X X X X X X
X X X X X X X X
X X X X X X X X
- - - X X X X X
- K - X X X P X
- - - X X X X X
- - X X X X X X
AD D X Y C
5  1 2 2 1
```

983

```
- - - X X X X X
- - - X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X P X
- - - X X X X X
- - - X X X X X
AD D X Y C
5  1 2 3 0
```

984

```
- - X X X X X X
- - X X X X X X
- - X X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X P X
- - - X X X X X
- - X X X X X X
AD D X Y C
5  1 2 3 1
```

985

```
- - - X X X X X
- - - X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X P X
- - - X X X X X
- - - X X X X X
- - - - X X X X
AD D X Y C
5  1 2 4 0
```

986

```
X X X X X X X X
X X X X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  1 2 4 1
```

987

```
X X X X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  1 2 5 0
```

988

```
X X X X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  1 2 5 1
```

989

```
- - - X X X X X
- K - X X X X X
- - - X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  1 2 6 0
```

990

```
- - - X X X X X
- K - X X X X X
- - - X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
5  1 2 6 1
```

991

```
- K - X X X X X
- - - X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
5 1 2 7 0

992

```
- K - X X X X X
- - - X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
5 1 2 7 1

993

```
- - - X X X X X
- - - X X X X X
- - - X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X X X
- - - X X X P X
- - - X X X X X
```

AD D X Y C
5 2 2 2 0

994

```
- - X X X X X X
- - X X X X X X
- - X X X X X X
- - X X X X X X
- K - X X X X X
- - - X X X X X
- - X X X X P X
- - X X X X X X
```

AD D X Y C
5 2 2 2 1

995

```
- - - X X X X X
- - - X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X X X
- - - X X X P X
- - - X X X X X
- - - X X X X X
```

AD D X Y C
5 2 2 3 0

996

```
- - X X X X X X
- - X X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X X X
- - X X X X P X
- - X X X X X X
- - X X X X X X
```

AD D X Y C
5 2 2 3 1

997

```
- - - X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X X X
- - - X X X P X
- - - X X X X X
- - - X X X X X
- - - - X X X X
```

AD D X Y C
5 2 2 4 0

998

```
X X X X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X X X
X X X X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
5 2 2 4 1

999

```
- - - X X X X X
- K - X X X X X
- - - X X X X X
X X X X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
5 2 2 5 0

1000

```
- - - X X X X X
- K - X X X X X
- - - X X X X X
X X X X X X P X
X X X X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
5 2 2 5 1

1001

```
- K - X X X X X
- - - X X X X X
X X X X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
5 2 2 6 0

1002

```
- K - X X X X X
- - - X X X X X
X X X X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
5 2 2 6 1

1003

```
- - - X X X X X
- - - X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X X X
- X - X X X X X
- - - X X X P X
- - - X X X X X
```

AD D X Y C
5 3 2 2 0

1004

```
- - X X X X X X
- - X X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X X X
X - X X X X X X
X X X X X X P X
- - X X X X X X
```

AD D X Y C
5 3 2 2 1

1005

```
- - - X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X X X
- X - X X X X X
- - - X X X P X
- - - X X X X X
- - - X X X X X
```

AD D X Y C
5 3 2 3 0

1006

```
- - X X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X X X
X - X X X X X X
X X X X X X P X
- - X X X X X X
- - X X X X X X
```

AD D X Y C
5 3 2 3 1

1007

```
- - - X X X X X
- K - X X X X X
- - - X X X X X
- X - X X X X X
- - - X X X P X
- - - X X X X X
- - - - X X X X
```

AD D X Y C
5 3 2 4 0

1008

```
- - - X X X X X
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
5 3 2 4 1

1009

```
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
5 3 2 5 0

1010

```
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
5 3 2 5 1

1011

```
- - - X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X X X
- X X X X X X X
X X X X X X X X
X X X X X X P X
- - - X X X X X
```

AD D X Y C
5 4 2 2 0

1012

```
- - X X X X X X
- - - X X X X X
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X X X
- - X X X X P X
- - X X X X X X
```

AD D X Y C
5 4 2 2 1

1013

```
- - - X X X X X
- K - X X X X X
- - - X X X X X
- X X X X X X X
- - - X X X X X
- - - X X X P X
- - - X X X X X
- - - X X X X X
```

AD D X Y C
5 4 2 3 0

1014

```
- - - X X X X X
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X P X
- - X X X X X X
- - X X X X X X
```

AD D X Y C
5 4 2 3 1

1015

```
- K - X X X X X
- - - X X X X X
- X X X X X X X
- - - X X X X X
- - - X X X P X
- - - X X X X X
- - - X X X X X
- - - - X X X X
```

AD D X Y C
5 4 2 4 0

1016

```
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X P X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
5 4 2 4 1

1017

```
- - - X X X X X
- K - X X X X X
- - - X X X X X
- X X X X X X X
- X X X X X X X
- - - X X X P X
- - - X X X X X
```

AD D X Y C
5 5 2 2 0

1018

```
- - - X X X X X
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X P X
- - X X X X X X
```

AD D X Y C
5 5 2 2 1

1019

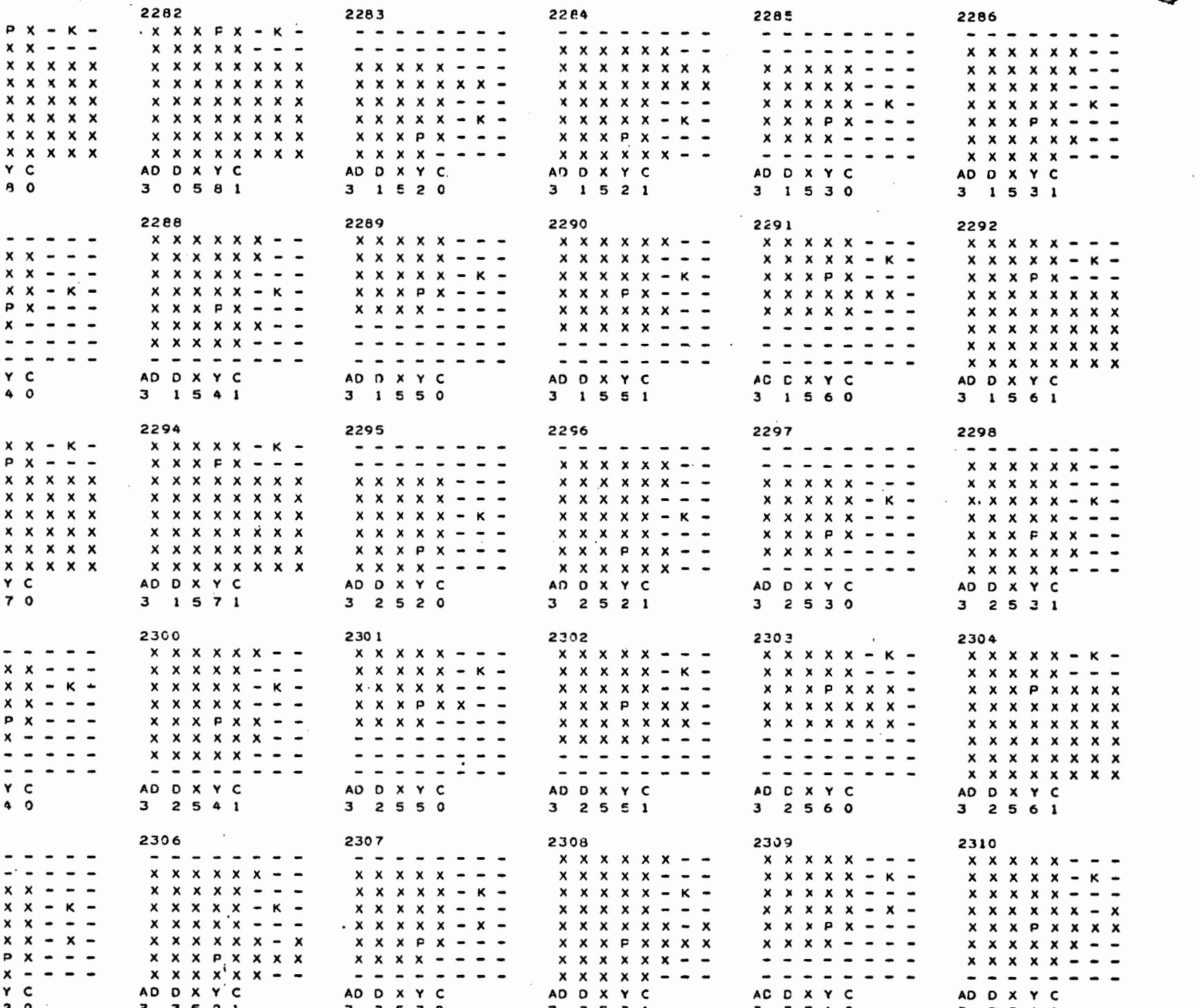
```
- K - X X X X X
- - - X X X X X
- X X X X X X X
- X X X X X X X
- X X X X X X X
- - - X X X P X
- - - X X X X X
- - - X X X X X
```

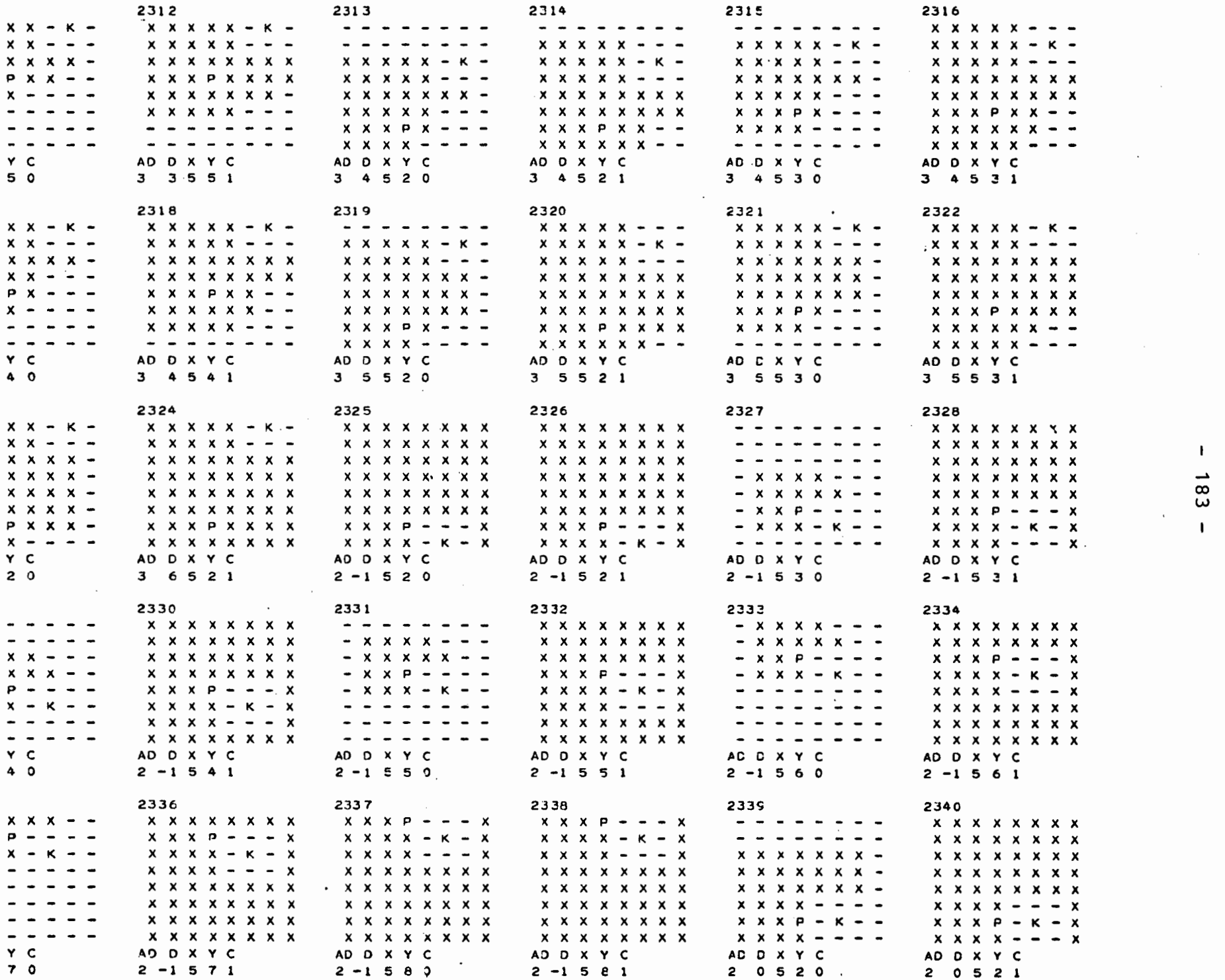
AD D X Y C
5 5 2 3 0

1020

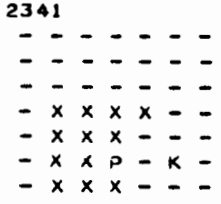
```
- K - X X X X X
- - - X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X P X
- - X X X X X X
- - X X X X X X
```

AD D X Y C
5 5 2 3 1


2281
X X X P X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
3 0 5 8 0
2282
X X X P X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X P X - - -
X X X X X X X X
AD D X Y C
3 0 5 8 1
2283
- - - - - - - - -
- - - - - - - - -
X X X X X - - -
X X X X X X X X
X X X X X X X -
X X X X X - - -
X X X X X - K -
X X X P X - - -
AD D X Y C
3 1 5 2 0
2284
- - - - - - - - -
X X X X X X - -
X X X X X X X X
X X X X X X X X
X X X X X - - -
X X X X X - K -
X X X P X - - -
X X X X - - - -
AD D X Y C
3 1 5 2 1
2285
- - - - - - - - -
- - - - - - - - -
X X X X X - - -
X X X X X - - -
X X X X X - K -
X X X P X - - -
X X X X - - - -
- - - - - - - - -
AD D X Y C
3 1 5 3 0
2286
- - - - - - - - -
X X X X X X - -
X X X X X X - -
X X X X X - - -
X X X X X - K -
X X X P X - - -
X X X X X X - -
X X X X X - - -
AD D X Y C
3 1 5 3 1
2287
- - - - - - - - -
X X X X X - - -
X X X X X - - -
X X X X X - K -
X X X P X - - -
X X X X - - - -
- - - - - - - - -
- - - - - - - - -
AD D X Y C
3 1 5 4 0
2288
X X X X X X - -
X X X X X X - -
X X X X X - - -
X X X X X - K -
X X X P X - - -
X X X X X X - -
X X X X X - - -
- - - - - - - - -
AD D X Y C
3 1 5 4 1
2289
X X X X X - - -
X X X X X - - -
X X X X X - K -
X X X P X - - -
X X X X X X - -
X X X X X - - -
- - - - - - - - -
- - - - - - - - -
AD D X Y C
3 1 5 5 0
2290
X X X X X X - -
X X X X X - - -
X X X X X - K -
X X X P X - - -
X X X X X X - -
X X X X X - - -
- - - - - - - - -
- - - - - - - - -
AD D X Y C
3 1 5 5 1
2291
X X X X X - - -
X X X X X - K -
X X X P X - - -
X X X X X X X -
X X X X X - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
AC D X Y C
3 1 5 6 0
2292
X X X X X - - -
X X X X X - K -
X X X P X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
3 1 5 6 1
2293
X X X X X - K -
X X X P X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
3 1 5 7 0
2294
X X X X X - K -
X X X P X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
3 1 5 7 1
2295
- - - - - - - - -
- - - - - - - - -
X X X X X - - -
X X X X X - - -
X X X X X - K -
X X X P X - - -
X X X P X X - -
X X X X X X - -
AD D X Y C
3 2 5 2 0
2296
- - - - - - - - -
- - - - - - - - -
X X X X X X - -
X X X X X X - -
X X X X X - K -
X X X P X - - -
X X X P X X - -
X X X X X X - -
AD D X Y C
3 2 5 2 1
2297
- - - - - - - - -
- - - - - - - - -
X X X X X - - -
X X X X X - K -
X X X P X - - -
X X X X - - - -
- - - - - - - - -
- - - - - - - - -
AD D X Y C
3 2 5 3 0
2298
- - - - - - - - -
X X X X X X - -
X X X X X - - -
X X X X X - K -
X X X X X - - -
X X X P X X - -
X X X X X X - -
X X X X X - - -
AD D X Y C
3 2 5 3 1
2299
- - - - - - - - -
X X X X X - - -
X X X X X - K -
X X X X X - - -
X X X P X - - -
X X X X - - - -
- - - - - - - - -
- - - - - - - - -
AD D X Y C
3 2 5 4 0
2300
X X X X X X - -
X X X X X - - -
X X X X X - K -
X X X X X - - -
X X X P X X - -
X X X X X X - -
- - - - - - - - -
- - - - - - - - -
AD D X Y C
3 2 5 4 1
2301
X X X X X - - -
X X X X X - K -
X X X X X - - -
X X X P X X - -
X X X X - - - -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
AD D X Y C
3 2 5 5 0
2302
X X X X X - - -
X X X X X - K -
X X X X X - - -
X X X P X X X -
X X X X X X X -
X X X X X - - -
- - - - - - - - -
- - - - - - - - -
AD D X Y C
3 2 5 5 1
2303
X X X X X - K -
X X X X X - - -
X X X P X X X -
X X X X X X X -
X X X X X X X -
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
AD D X Y C
3 2 5 6 0
2304
X X X X X - K -
X X X X X - - -
X X X P X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
AD D X Y C
3 2 5 6 1
2305
- - - - - - - - -
- - - - - - - - -
X X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X - X -
X X X P X - - -
X X X X - - - -
AD D X Y C
3 3 5 2 0
2306
- - - - - - - - -
X X X X X X - -
X X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X - X
X X X P X X X X
X X X X X X - -
AD D X Y C
3 3 5 2 1
2307
- - - - - - - - -
X X X X X - - -
X X X X X - K -
X X X X X - X -
X X X P X - - -
X X X X - - - -
- - - - - - - - -
- - - - - - - - -
AD D X Y C
3 3 5 3 0
2308
X X X X X X - -
X X X X X - - -
X X X X X - K -
X X X X X X - X
X X X P X X X X
X X X X X X - -
X X X X X - - -
- - - - - - - - -
AD D X Y C
3 3 5 3 1
2309
X X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X - X -
X X X P X - - -
X X X X - - - -
- - - - - - - - -
- - - - - - - - -
AC D X Y C
3 3 5 4 0
2310
X X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X - X
X X X P X X X X
X X X X X X - -
X X X X X - - -
- - - - - - - - -
AD D X Y C
3 3 5 4 1

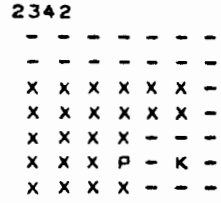

2311
AD D X Y C
3 3 5 5 0
2312
AD D X Y C
3 3 5 5 1
2313
AD D X Y C
3 4 5 2 0
2314
AD D X Y C
3 4 5 2 1
2315
AD D X Y C
3 4 5 3 0
2316
AD D X Y C
3 4 5 3 1
2317
AD D X Y C
3 4 5 4 0
2318
AD D X Y C
3 4 5 4 1
2319
AD D X Y C
3 5 5 2 0
2320
AD D X Y C
3 5 5 2 1
2321
AD D X Y C
3 5 5 3 0
2322
AD D X Y C
3 5 5 3 1
2323
AD D X Y C
3 6 5 2 0
2324
AD D X Y C
3 6 5 2 1
2325
AD D X Y C
2 -1 5 2 0
2326
AD D X Y C
2 -1 5 2 1
2327
AD D X Y C
2 -1 5 3 0
2328
AD D X Y C
2 -1 5 3 1
2329
AD D X Y C
2 -1 5 4 0
2330
AD D X Y C
2 -1 5 4 1
2331
AD D X Y C
2 -1 5 5 0
2332
AD D X Y C
2 -1 5 5 1
2333
AD D X Y C
2 -1 5 6 0
2334
AD D X Y C
2 -1 5 6 1
2335
AD D X Y C
2 -1 5 7 0
2336
AD D X Y C
2 -1 5 7 1
2337
AD D X Y C
2 -1 5 8 0
2338
AD D X Y C
2 -1 5 8 1
2339
AD D X Y C
2 0 5 2 0
2340
AD D X Y C
2 0 5 2 1

2341



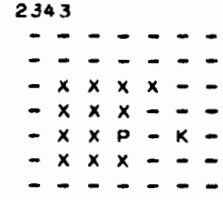
AD D X Y C
2 0 5 3 0

2342



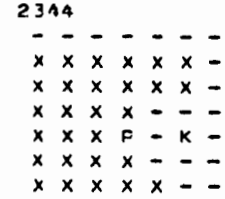
AD D X Y C
2 0 5 3 1

2343



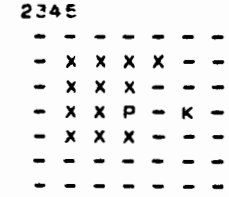
AD D X Y C
2 0 5 4 0

2344



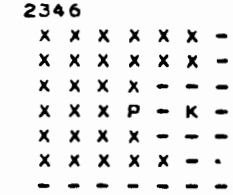
AD D X Y C
2 0 5 4 1

2345



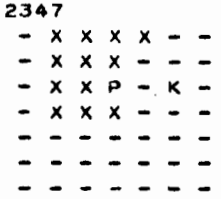
AD D X Y C
2 0 5 5 0

2346



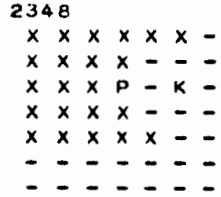
AD D X Y C
2 0 5 5 1

2347



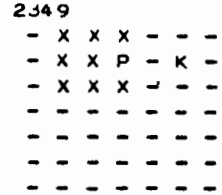
AD D X Y C
2 0 5 6 0

2348



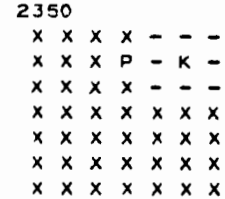
AD D X Y C
2 0 5 6 1

2349



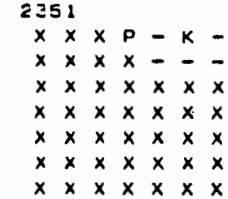
AD D X Y C
2 0 5 7 0

2350



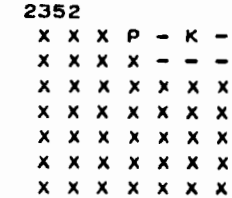
AD D X Y C
2 0 5 7 1

2351



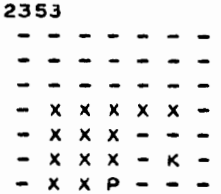
AD D X Y C
2 0 5 8 0

2352



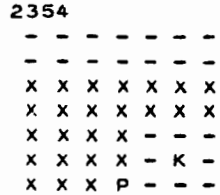
AD D X Y C
2 0 5 8 1

2353



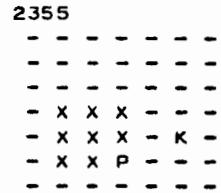
AD D X Y C
2 1 5 2 0

2354



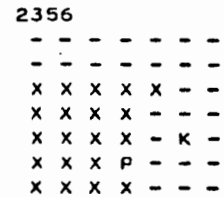
AD D X Y C
2 1 5 2 1

2355



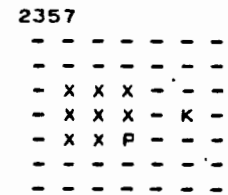
AD D X Y C
2 1 5 3 0

2356



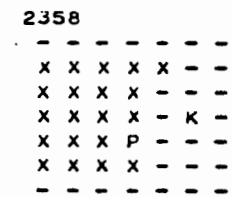
AD D X Y C
2 1 5 3 1

2357



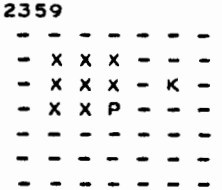
AD D X Y C
2 1 5 4 0

2358



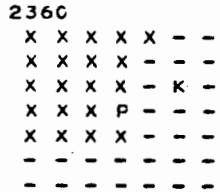
AD D X Y C
2 1 5 4 1

2359



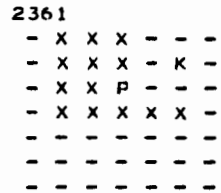
AD D X Y C
2 1 5 5 0

2360



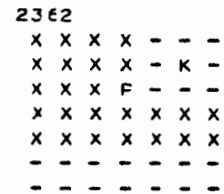
AD D X Y C
2 1 5 5 1

2361



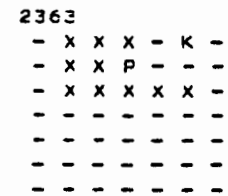
AD D X Y C
2 1 5 6 0

2362



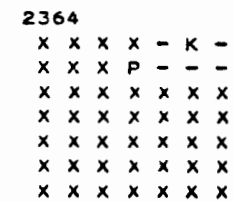
AD D X Y C
2 1 5 6 1

2363



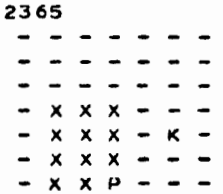
AD D X Y C
2 1 5 7 0

2364



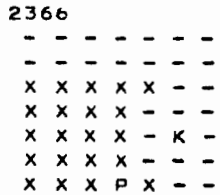
AD D X Y C
2 1 5 7 1

2365



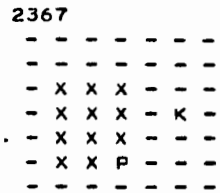
AD D X Y C
2 2 5 2 0

2366



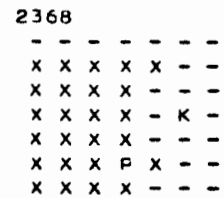
AD D X Y C
2 2 5 2 1

2367



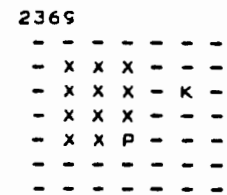
AD D X Y C
2 2 5 3 0

2368



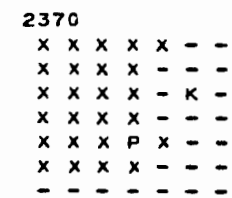
AD D X Y C
2 2 5 3 1

2369



AD D X Y C
2 2 5 4 0

2370



AD D X Y C
2 2 5 4 1

3871

```
X X X X X X - K
X X X X X X - -
P X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 2 8 6 0

3872

```
X X X X X X - K
X X X X X X - -
P X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 2 8 6 1

3873

```
X X X X X X X X
X X X X X X X X
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
P X X X X X X X
X X X X X X X X
```

AD D X Y C
7 3 8 2 0

3874

```
X X X X X X X X
X X X X X X X X
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
P X X X X X X X
X X X X X X X X
```

AD D X Y C
7 3 8 2 1

3875

```
X X X X X X X X
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 3 8 3 0

3876

```
X X X X X X X X
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 3 8 3 1

3877

```
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 3 8 4 0

3878

```
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 3 8 4 1

3879

```
X X X X X X - K
X X X X X X - -
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 3 8 5 0

3880

```
X X X X X X - K
X X X X X X - -
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 3 8 5 1

3881

```
X X X X X X X X
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X X X X
```

AD D X Y C
7 4 8 2 0

3882

```
X X X X X X X X
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
X X X X X X X X
P X X X X X X X
X X X X X X X X
```

AD D X Y C
7 4 8 2 1

3883

```
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 4 8 3 0

3884

```
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 4 8 3 1

3885

```
X X X X X X - K
X X X X X X - -
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 4 8 4 0

3886

```
X X X X X X - K
X X X X X X - -
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 4 8 4 1

3887

```
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 5 8 2 0

3888

```
X X X X X X - -
X X X X X X - K
X X X X X X - -
X X X X X X X X
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 5 8 2 1

3889

```
X X X X X X - K
X X X X X X - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 5 8 3 0

3890

```
X X X X X X - K
X X X X X X - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 5 8 3 1

3891

```
X X X X X X - K
X X X X X X - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 6 8 2 0

3892

```
X X X X X X - K
X X X X X X - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
7 6 8 2 1

3893

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X - - -
X X X X X - K -
X X X X X X - -
```

AD D X Y C
6 -1 8 2 0

3894

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X - - -
X X X X X - K -
X X X X X X - -
```

AD D X Y C
6 -1 8 2 1

3895

```
X X X X X - - -
X X X X X - - -
X X X X X X - -
X X X X X X - -
X X X X X X X -
P X X X X - - -
X X X X X - K -
- - - - - - - -
```

AD D X Y C
6 -1 8 3 0

3896

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X - - -
X X X X X - K -
X X X X X - - -
```

AD D X Y C
6 -1 8 3 1

3897

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X X X
```

AD D X Y C
6 -1 8 4 0

3898

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X X X
```

AD D X Y C
6 -1 8 4 1

3899

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 -1 8 5 0

3900

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
P X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 -1 8 5 1

3901

```
X X X X X X X X
X X X X X X X X
P X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 -1 8 6 0

3902

```
X X X X X X X X
X X X X X X X X
P X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 -1 8 6 1

3903

```
X X X X X X X X
P X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 -1 8 7 0

3904

```
X X X X X X X X
P X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 -1 8 7 1

3905

```
P X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 -1 8 8 0

3906

```
P X X X X - - -
X X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 -1 8 8 1

3907

```
X X X X X - - -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X - - -
P X X X X - K -
- - - - - - - -
```

AD D X Y C
6 0 8 2 0

3908

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X - - -
P X X X X - K -
X X X X X - - -
```

AD D X Y C
6 0 8 2 1

3909

```
X X X X X - - -
X X X X X - - -
X X X X X - - -
X X X X X X X X
X X X X X - - -
P X X X X - K -
- - - - - - - -
- - - - - - - -
```

AD D X Y C
6 0 8 3 0

3910

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X - - -
P X X X X - K -
X X X X X - - -
X X X X X X X X
```

AD D X Y C
6 0 8 3 1

3911

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X - - -
P X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 0 8 4 0

3912

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X - - -
P X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 0 8 4 1

3913

```
X X X X X X X X
X X X X X X X X
X X X X X - - -
P X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 0 8 5 0

3914

```
X X X X X X X X
X X X X X X X X
X X X X X - - -
P X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 0 8 5 1

3915

```
X X X X X X X X
X X X X X - - -
P X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 0 8 6 0

3916

```
X X X X X X X X
X X X X X - - -
P X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 0 8 6 1

3917

```
X X X X X - - -
P X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 0 8 7 0

3918

```
X X X X X - - -
P X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 0 8 7 1

3919

```
P X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 0 8 8 0

3920

```
P X X X X - K -
X X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 0 8 8 1

3921

```
X X X X X - - -
X X X X X - - -
X X X X X - - -
X X X X X X X -
X X X X X - - -
X X X X X - K -
P - - - - - - -
- - - - - - - -
```

AD D X Y C
6 1 8 2 0

3922

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X - - -
X X X X X - K -
P X X X X - - -
X X X X X X X X
```

AD D X Y C
6 1 8 2 1

3923

```
X X X X X - - -
X X X X X - - -
X X X X X - - -
X X X X X - - -
X X X X X - K -
P - - - - - - -
- - - - - - - -
- - - - - - - -
```

AD D X Y C
6 1 8 3 0

3924

```
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X - - -
X X X X X - K -
P X X X X - - -
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 1 8 3 1

3925

```
X X X X X X X X
X X X X X X X X
X X X X X - - -
X X X X X - K -
P X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 1 8 4 0

3926

```
X X X X X X X X
X X X X X X X X
X X X X X - - -
X X X X X - K -
P X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 1 8 4 1

3927

```
X X X X X X X X
X X X X X - - -
X X X X X - K -
P X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 1 8 5 0

3928

```
X X X X X X X X
X X X X X - - -
X X X X X - K -
P X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 1 8 5 1

3929

```
X X X X X - - -
X X X X X - K -
P X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 1 8 6 0

3930

```
X X X X X - - -
X X X X X - K -
P X X X X - - -
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
X X X X X X X X
```

AD D X Y C
6 1 8 6 1