DEVELOPMENT OF HEURISTIC FUNCTIONS FOR MINIMAX GAME PLAYING STRATEGY

MR. THANAKRIT THITHAPANT

MR. NUTTHAPON SARIGA

MR. TANYAWAT SANGUANCHUE

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIRMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

IN COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2009

Special Project Title	Development of Heuristic Functions for Minimax Game Playing Strategy	
Students	Mr. Thanakrit Thithapant	ID.49050159
	Mr. Nutthapon Sariga	ID.49050376
	Mr. Tanyawat Sanguanchue	ID.49050383
Degree	Bachelor of Science	
Major Program	Computer Science (International Programme)	
Academic Year	2009	
Special Project Advisor	Associate Professor Dr.Veera Boonjing	

ABSTRACT

This special project proposes a new heuristic function to solve a "depth of search" problem of minimax game playing. The heuristic function is created for the checker game using minimax algorithms to provide all possible moves. Our heuristic function consists of three sub functions which are working together and help minimax algorithm to decide the best position. We made experiments to compare game results of our heuristic function with Ander Bauman's heuristic function in different depth of search. The results show the percentage of competition (won 35%, tie 23.75% and, lost 41.25%).

Keywords : Heuristic function, Minimax, Game Playing, Checker, Draught

Acknowledgement

The authors would like to express their gratitude to all those who gave us the opportunity to complete this project

The authors would like to sincerely express their gratitude to the advisor Assoc.Prof. Dr.Veera Boonjing for his invaluable guidance, useful advice, kind and constructive criticism, consistent inspiration and encouragement throughout this research.

In addition, we would also like to thank to Asst.Prof. Dr.Sarun Intakosum and Asst.Prof.Dr.Jeeraporn Werapun, members of the project committee for their invaluable comments.

We greatly appreciate all professors who have given invaluable knowledge while we were studying in the department of computer science, Faculty of Science King Mongkut's Institute of Technology Ladkrabang.

We would like to express our greatest gratitude to Dr.Songkarn Jarusisisawad and Mr.Chaiyong Ragkhitwetsagul for their invaluable guidance and useful advice.

Also thanks go to Mr. Anders Baumann for his checker program that we used to study and to produce the experiment for our special project.

We would also like to give special thanks to all students in the International Program at Faculty of Science, King Mongkut's Institute of Technology Ladkrabang, who have always provided their encouragement and collaboration during this study.

Finally, the authors would like to express their deepest appreciation to their dearest fathers, mothers, brothers and sisters for love, care and encouragement. They are the most important in our lives forever.

Mr. Thanakrit Thithapant

Mr. Nutthapon Sariga

Mr. Tanyawat Sanguanchue

Table of Contents

Table of Contents (cont.)

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Rationale and Research Motivation

Artificial intelligence (AI, from this point forward AI will be used to refer to Artificial Intelligence) as a science is to make machines do things that would require intelligence if done by a human (Boden, 1977). In Artificial Intelligence, problem solving is a fundamental problem for most AI applications. Many search techniques can be used in problem solving. In two player board game system, the main problem is a "depth of search" and the problem can be solved by a heuristic function.

A heuristic function is a general function that helps minimax algorithm to solve a problem and a result expected to be the best answer or the best position.

In this special project, we use a two player board game to present an idea of Artificial Intelligence application, because the two player board game is very complex and includes many ways of move form. Furthermore, in two player board game, both players need to guess the move of the opposition. We will use the heuristic function to decide the best move approximated from exhaustive search which is the standard for selecting the move ply in many game playing strategy, then we will test for efficiency of heuristic function by comparing game results with another heuristic function.

1.2 Objectives

The objective of this project is to develop heuristic functions for Minimax game playing strategy that used in game playing. We use n ply search to evaluate the value of node, then we compare with another Minimax algorithms.

1.3 Scope of Study

In this project, the authors applied the created heuristic function into a two player board game called "checker" and asked it to select the best move. In experiments, we compare game results with Ander Baumann's checker who used a heuristic function and minimax algorithm with Alpha-Beta pruning technique. Furthermore, we limited our experiments to 3 and 5 ply.

1.4 Organization

The rest of this special project consists of 4 chapters as follows:

Chapter 2 shows background of Artificial Intelligence techniques, and checker rules.

Chapter 3 shows how to use our checker game and our heuristic function's workflow.

Chapter 4 shows the experiments rule and experiment results.

Chapter 5 shows conclusion and recommendation for this special project.

Chapter 2

Background

2.1 Checker

Checker is a board game played between two players, who alternate moves. The player, who cannot move, because he has no pieces, or because all of his pieces are blocked, loses the game. Players can resign or agree to draw.

2.1.1 A checker set: a checker board and pieces.

2.1.1.1 Board

The board is square, with sixty-four smaller squares, arranged in an 8x8 grid. The smaller squares are alternately light and dark colored (green and buff in tournaments), in the famous "checker-board" pattern. The game of checkers is played on the dark (black or green) squares. Each player has a dark square on his far left and a light square on his far right. The double-corner is the distinctive pair of dark squares in the near right corner.

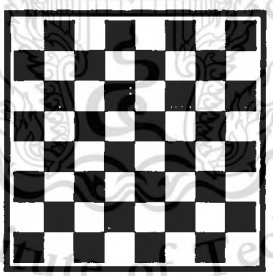


Figure 2.1 A Checker board

2.1.1.2 Pieces

The pieces are Red and White, and are called Black and White in most books. In some modern publications, they are called Red and White. Sets bought in stores may be other colors. Black and Red pieces are still called Black (or Red) and White, so that you can read the books. The pieces are of cylindrical shape, much wider than they are tall (see Figure 2.2). Tournament pieces are smooth, and have no designs (crowns or concentric circles) on them. The pieces are placed on the dark squares of the board.

Figure 2.2 Checker pieces

2.1.2 The starting position

The starting position is with each player having twelve pieces, on the twelve dark squares closest to his edge of the board. Notice that in checker diagrams, the pieces are usually placed on the light colored squares, for readability. On a real board they are on the dark squares.

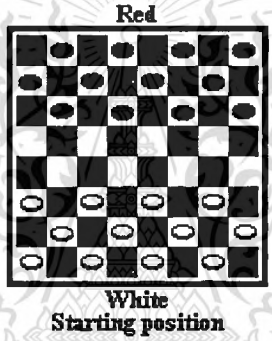


Figure 2.3 Initial setup of checker pieces

2.1.3 Moving

You can make only one move per turn. You must move. If you cannot move, you lose. The pieces (not kings) move one square, diagonally, forward. Kings can move one square diagonally, forward or backward. A piece can only move to a vacant square.

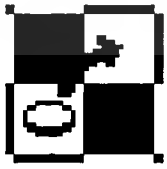


Figure 2.4 Moving

2.1.4 Kings

When a piece reaches the last row (the King Row), it becomes a King. A second checker is placed on top of that one, by the opponent. A piece that has just become a king, cannot continue jumping pieces, until the next move.

2.1.5 Jumping

You can capture an opponent's piece by jumping over it, diagonally, to the adjacent vacant square. A king can jump diagonally, forward or backward. A piece, which is not a king, can only jump diagonally forward. You can make a multiple jump, with one piece only, by jumping from empty square to empty square. You can only jump one piece, with any given jump. But you can jump several pieces, with a move of several jumps. You remove the jumped pieces from the board. You cannot jump your own piece. You cannot jump the same piece twice, in the same move, jumps must be completed. If you can jump, you must. And, a multiple jump must be completed. You cannot stop part way through a multiple jump. If you have a choice of jumps, you can choose among them, regardless of whether some of them are multiple, or not. A piece, whether it is a king or not, can jump a king.

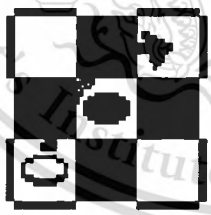


Figure 2.5 Jumping.

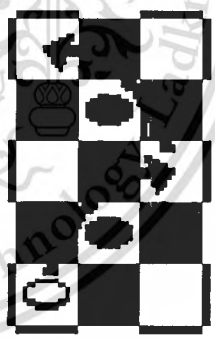


Figure 2.6 Multiple Jumping.

2.1.6 How the game ends

2.1.6.1 A player wins by capturing all of the opposing player's pieces

2.1.6.2 Leaving the opposing player with no legal moves

2.2 Game Playing

Ever since the beginning of AI, there has been a great fascination in pitting the human expert against the computer. Game playing provided a high-visibility platform for this contest. It is important to note, however, that the performance of the human expert and the AI game-playing program reflect qualitatively different processes. More specifically, as mentioned earlier, the performance of the human expert utilizes a vast amount of domain specific knowledge and procedures. Such knowledge allows the human expert to generate a few promising moves for each game situation (irrelevant moves are never considered). In contrast, when selecting the best move, the game playing program exploits brute-force computational speed to explore as many alternative moves and consequences as possible. As the computational speed of modern computers increases, the contest of knowledge vs. speed is tilting more and more in favor of the computers.

2.2.1 The Min-Max Algorithm

The Min-Max algorithm (minimax algorithm) is applied in two player games, such as tic-tac-toe, checker, chess, go, and so on.

All these games have at least one thing in common, they are logic games. This means that they can be described by a set of rules and premises. With them, it is possible to know from a given point in the game, what are the next available moves. So they also share other characteristics, they are "full information games". Each player knows everything about the possible moves of the adversary.

Before explaining the algorithm, a brief introduction to search trees is required. Search trees are a way to represent searches. In Figure 2.7 you can see a representation of a search tree. The squares are known as nodes and they represent points of the decision in the search. The nodes are connected with branches. The search starts at the root node, the one at the top of the figure. At each decision point, nodes for the available search paths are generated, until no more decisions are possible. The nodes that represent the end of the search are known as leaf nodes.

There are two players involved, MAX and MIN. A search tree is generated, depth-first, starting with the current game position up to the end game position. Then, the final game position is evaluated from MAX's point of view, as shown in Figure 2.7. Afterwards, the inner node values of the tree are filled bottom-up with the evaluated values. The nodes that

belong to the MAX player receive the maximum value of its children. The nodes for the MIN player will select the minimum value of its children. The algorithm is described in Figure 2.8

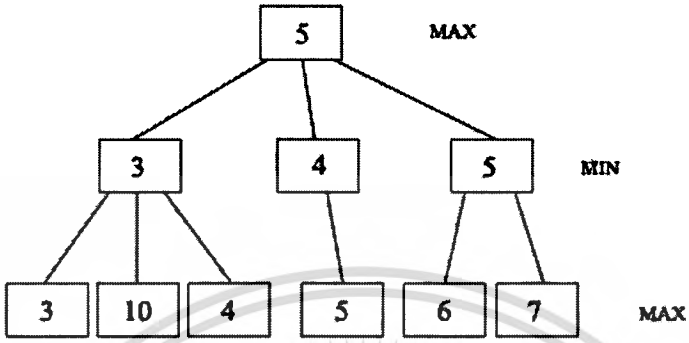


Figure 2.7 Min-Max search tree

So what is happening here? The values represent how good a game move is. So the MAX player will try to select the move with highest value in the end. But the MIN player also has something to say about it and he will try to select the moves that are better to him, thus minimizing MAX's outcome.

```
MinMax (GamePosition game) {
      return MaxMove (game);
}

MaxMove (GamePosition game) {
      if (GameEnded(game)) {
            return EvalGameState(game);
}
else {
      best_move <- {};
      moves <- GenerateMoves(game);
      ForEach moves {
            move <- MinMove(ApplyMove(game));
            if (Value(move) > Value(best_move)) {
            best_move <- move;
            }
      }
      return best_move;
        }
}
MinMove (GamePosition game) {
      best_move <- {};
      moves <- GenerateMoves(game);
      ForEach moves {
            move <- MaxMove(ApplyMove(game));
            if (Value(move) > Value(best_move)) {
                  best_move <- move;
      }
      }
      return best_move;
}
```

Figure 2.8 Basic Min-Max Algorithm

2.2.2 Heuristic function

Heuristic function is used for minimax algorithm which finds solutions among all possible nodes. The heuristic function does not guarantee that the best solution will be found, it usually find a solution close to the best solution.

Chapter 3

Development of Heuristic Function

This special project presents the checker game with the heuristic function and minimax algorithm. The AI of the game will select the best move using heuristic function; we developed. Then we will test our checker game with Ander Baumann's checker who used a heuristic function and minimax algorithm with Alpha-Beta pruning technique.

3.1 Checker Game

Our checker game is developed by using Java. The checker game uses the minimax algorithm to search all possible moves and use the heuristic function to select the best move.

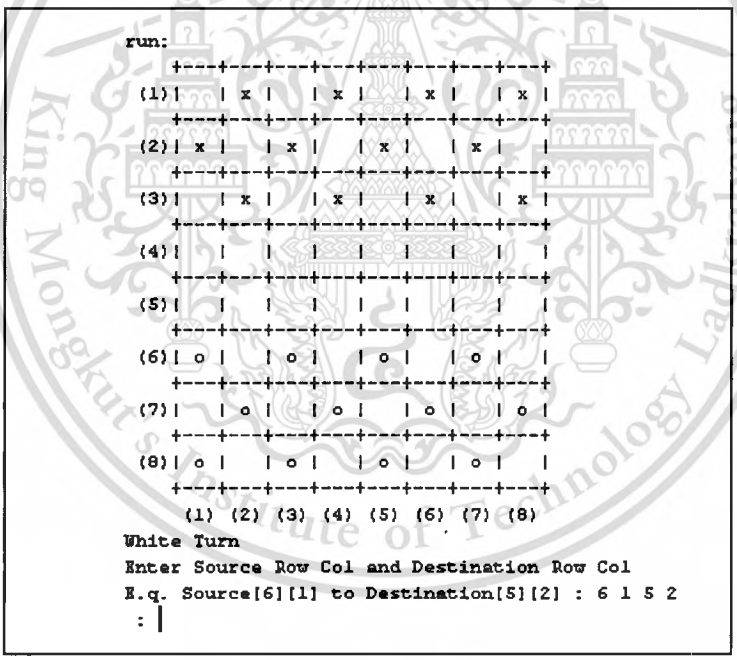


Figure 3.1 Checker game staring position

3.1.1 Moving

While moving, enter row and column of source and destination. E.g. Player wants to move checker piece from row (6) column (1) to row (5) column (2), an input is "6 1 5 2".

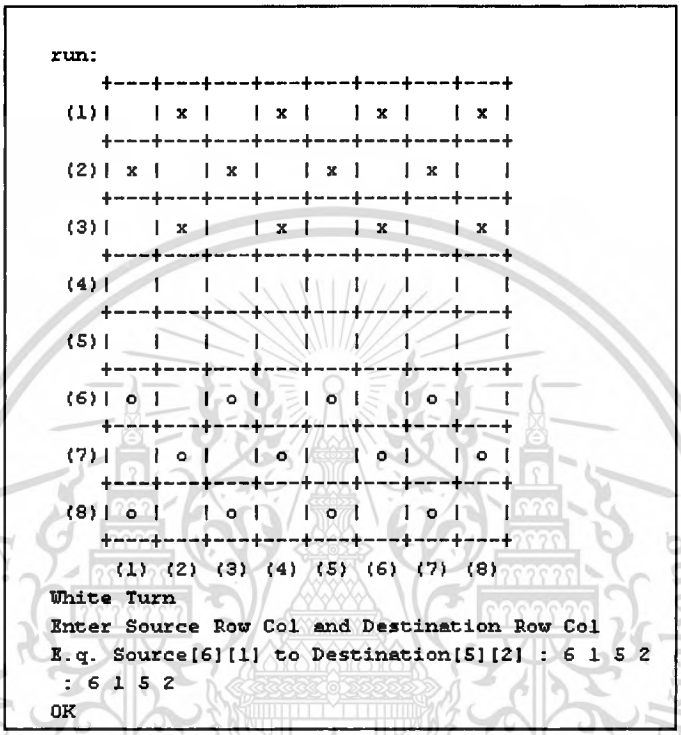


Figure 3.2 Enter source position and destination position.

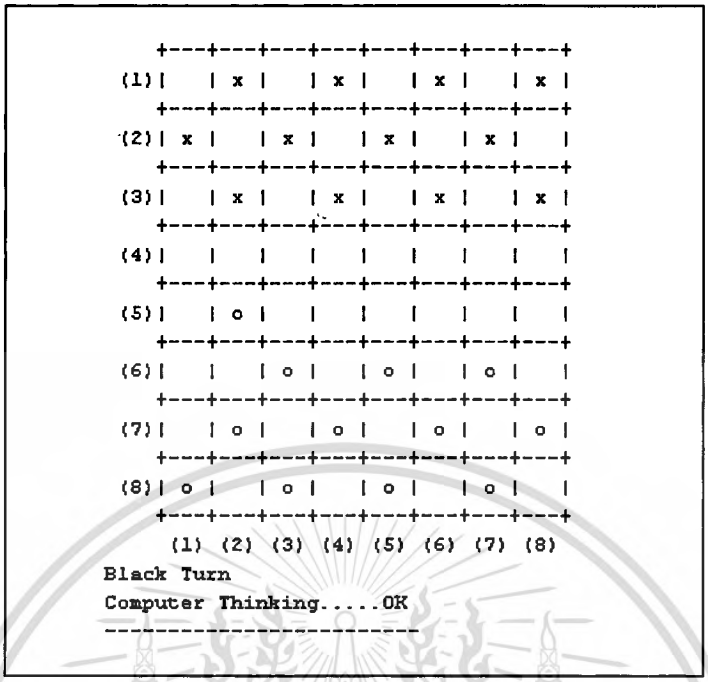


Figure 3.3 Computer Turn (Thinking and Moving)

In figure 3.3, The AI chooses the best move by using our heuristic function and minimax algorithm.

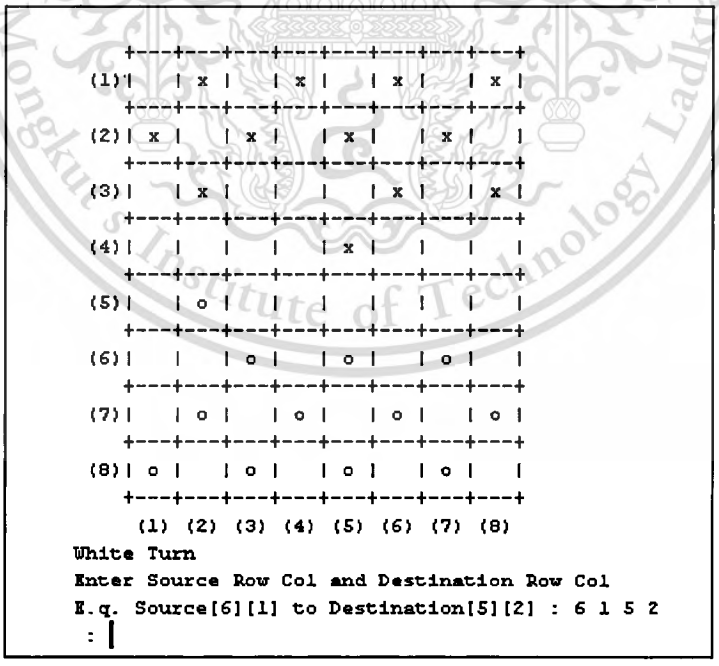


Figure 3.4 Player turn after AI has moved

3.1.2 Jumping

Follow the force jump rule; if there are available jump-positions, player must jump. Players are able to capture the competitor's piece by inputting current position and possible position that able to jump.

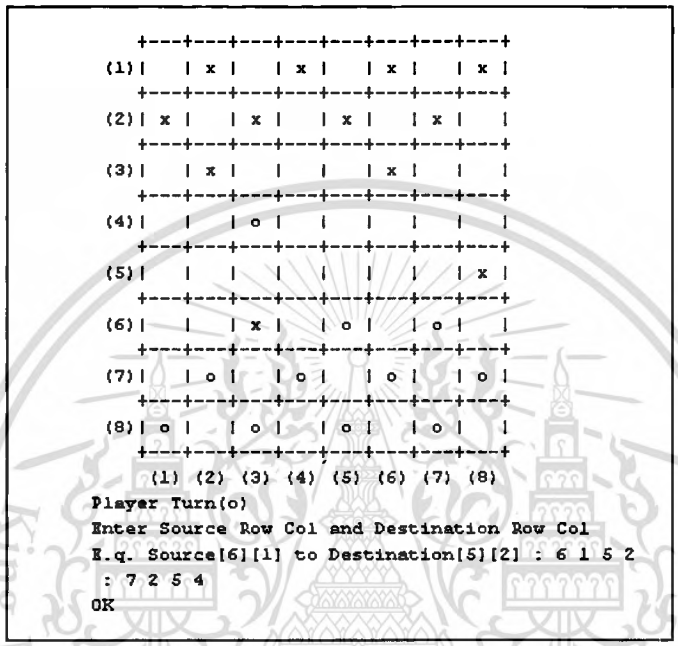


Figure 3.5 Inputting position to jump

In the figure 3.5, player is able to jump from row (7) column (2) to row (5) column (4). So, an input is "7 2 5 4".

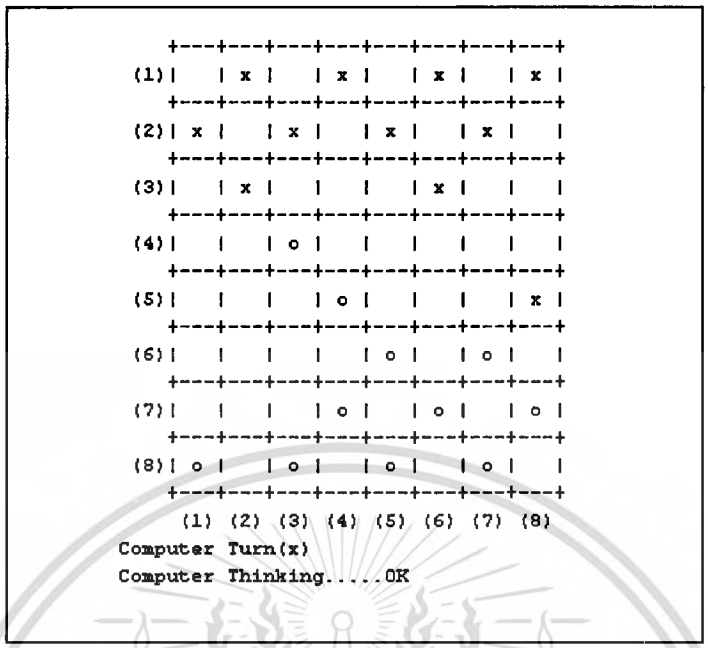


Figure 3.6 The result of jump.

Multiple jumping

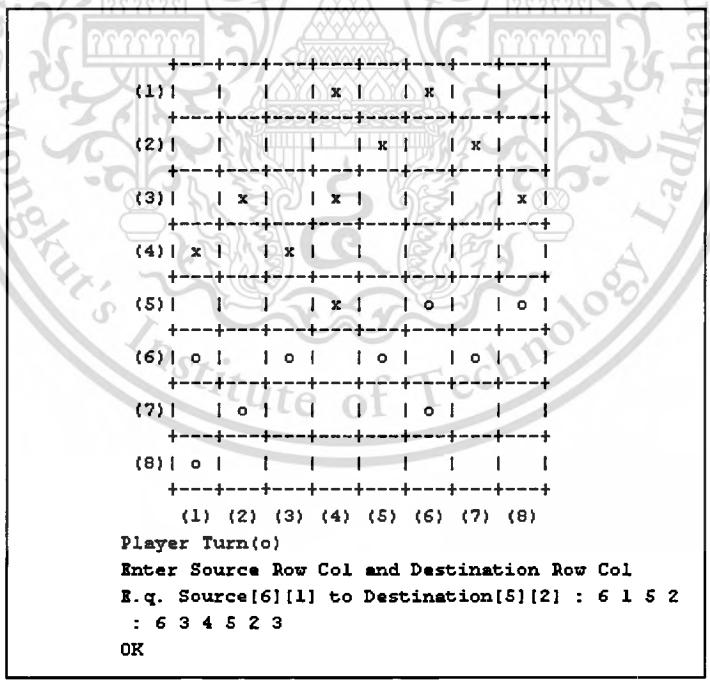


Figure 3.7 Making the multiple jump.

While jumping, if a player able to continue multiple jumps, the player must, by inputting position following this step; "Your Position + Your n position to jump + Your n+1 position to jump + ..." In figure 3.7, player is able to jump from row (6) column (3) to row (4) column (5) and able to multiple jump to row (2) column (3). The input value is "6 3 4 5 2 3".

3.1.3 King

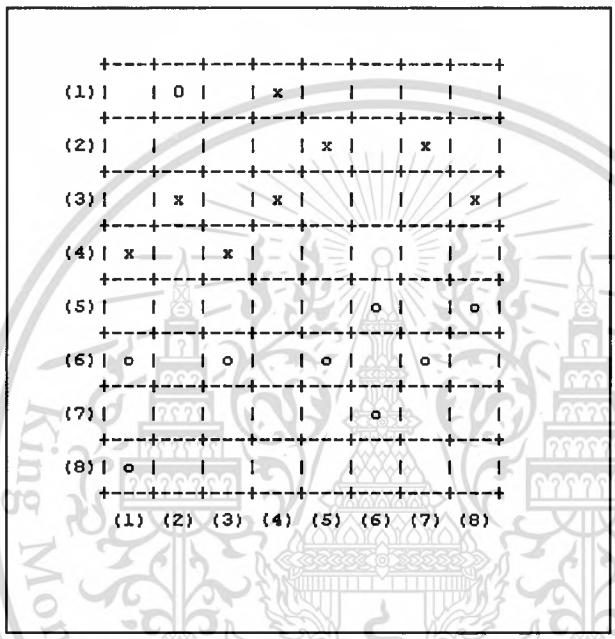


Figure 3.8 Promote the piece to be the king.

If a piece reaches the last row (the King Row), it becomes a King. A piece that has just been a king cannot continue jumping pieces, until next move.

3.1.4 End game

From the checker rules, if a player has no legal moves or checker pieces on board, opposing player has won in that game.

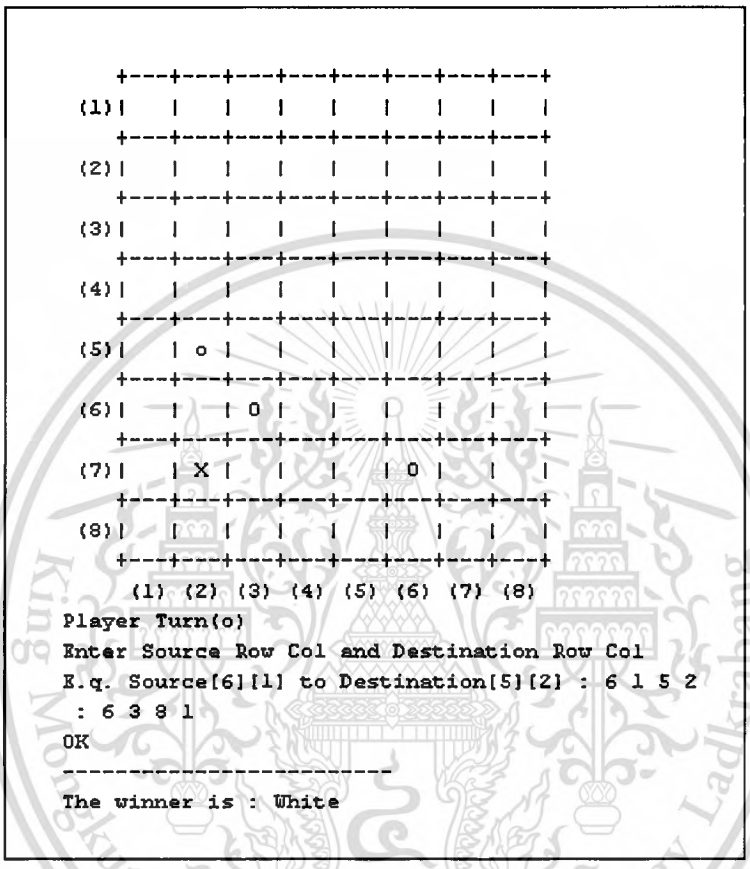


Figure 3.9 End Game.

3.2 Heuristic Function

In our special project we developed a heuristic function, named ScoreFunction. The ScoreFunction consists of three sub functions.

3.2.1 Declare values into all checker pieces.

A value of the Black-Piece equal to -10, the White-Piece equal to 10, the Black-King equal to -15, and the White-King equal to 15.

A(n) = P(n) + B(n) + E(n) where:

A(n) is a total of evaluation for state n.

We use these A(n) to get MIN or MAX value to move the checker.

P(n) = M(n) + O(n) where:

M(n) is a total of My(MIN) checker pieces value on node n.

O(n) is a total of Opponent's(MAX) checker pieces value on node n.

P(n) is a total of evaluation for state n.

```
for(int row=0;row<9;row++)
  for(int col=0;col<9;col++)
    score += board.getValue(board.getChecker(row, col));
```

Figure 3.10 Evaluate checker pieces value on the checker board one by one.

3.2.2 Declare values into the Checker Board

B(n) = M(n) – P(n) where:

M(n) is a value of checker piece on the checker board's position on node n.

P(n) is a value of the checker board's position on node n.

B(n) is a total of evaluation for node n.

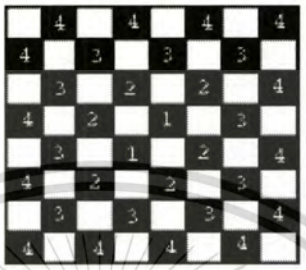


Figure 3.10 Declare values into the Checker Board

3.2.3 Declare value into opponent's checker pieces

If our piece already promoted to be King, we have a function to declare a value to all opponent pieces. So, our AI will move to the opponent pieces and try to jump on them.

E(n) = M(n) - P(n) where:

M(n) is a value of checker-king piece on node n.

P(n) is a value of checker board's positions of 4 square of Opponent's checker pieces and P(n) is equal to 5.

Chapter 4

Experiment Result

4.1 Experiments

In our experiments, we test the ScoreFunction with Anders Baumann's checker game, which uses heuristic function and minimax algorithm with alpha-beta pruning technique. We produced four experiments, record and calculate in percentage. Each experiment we compete our checker (black pieces) and Ander Baumann's checker (white pieces) with different depth of search (see Table4-1) for 20 matches, our checker moves first for 10 matches and Ander Baumann's checker moves first for 10 matches.

Table 4-1: Experimental Rule

Experiments	Our Checker (MAX DEPTH)	Ander Bauman's Checker (MAX DEPTH)
1st experiment	3	3
2nd experiment	3	5
3rd experiment	5	3
4th experiment	5	5

There are four tables of experiment results, in every tables, it shows details of match result containing first move, board scores, white piece scores, black piece scores and match winner. In every match, we also recorded checker's endgame state, and they will be in appendix B as shown in Figure 4.1.

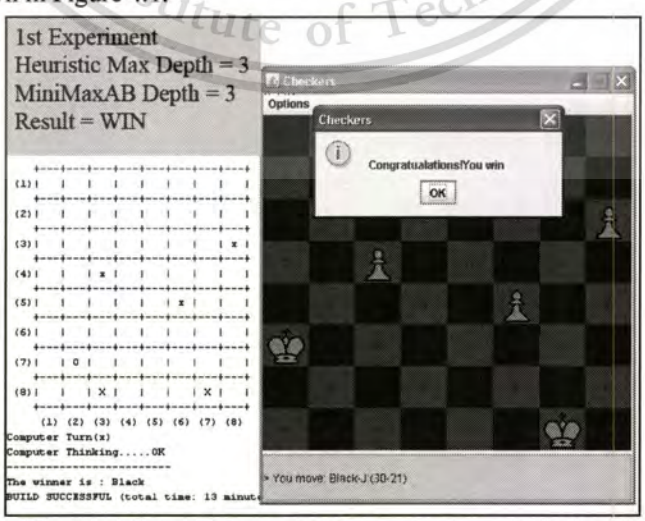


Figure 4.1 An example of checker's endgame state.

4.2 The first experiment

In the first experiment, our checker has won 7 matches it can be calculated in percentage about 35%. If our checker moves first, it has won 6 in 10 matches, it's about 60% and when our checker moves next to Ander's our checker has no win, but it got a tie 4 in 10 matches. The percentages are won 0%, tie 40%, lost 60%. So the percentages are won 30%, tie 20%, lost 50%.

Table 4-2: The first experiment

Match	First Move	Board Score	White Score	Black Score	Match Winner
1	Our checker	-50	0	-50	Our checker
2	Our checker	60	60	0	Ander's
3	Our checker	-25	0	-25	Our checker
4	Our checker	-45	10	-55	Our checker
5	Our checker	75	75	0	Ander's
6	Our checker	-40	0	-40	Our checker
7	Our checker	-60	0	-60	Our checker
8	Our checker	-20	0	-20	Our checker
9	Our checker	70	70	0	Ander's
10	Our checker	35	34	0	Ander's
11	Ander's	15	25	-10	Ander's
12	Ander's	110	110	0	Ander's
13	Ander's	15	45	30	TIE
14	Ander's	10	40	30	TIE
15	Ander's	80	80	0	Ander's
16	Ander's	25	25	0	Ander's
17	Ander's	60	60	0	Ander's
18	Ander's	-40	30	-70	TIE
19	Ander's	50	50	0	Ander's
20	Ander's	45	70	-25	TIE

4.3 The second experiment

In the second experiment, our checker has won 2 in 20 matches it can be calculated in a percentage about 10%. If our checker moves first it has won 1 in 10 matches it's about 10% and, when our checker moves next to Ander's our checker has won 1 in 10 matches again. So the percentages are won 10%, tie 25%, and lost 65%.

Table 4-3: The second experiment

Match	First Move	Board Score	White Score	Black Score	Match Winner
1	Our checker	15	30	-15	Ander's
2	Our checker	90	90	0	Ander's
3	Our checker	40	40	0	Ander's
4	Our checker	0	30	-30	TIE
5	Our checker	-45	0	-45	Our checker
6	Our checker	-10	50	-60	TIE
7	Our checker	30	30	0	Ander's
8	Our checker	60	70	-10	Ander's
9	Our checker	10	50	40	TIE
10	Our checker	65	65	0	Ander's
11	Anders's	40	40	0	Ander's
12	Anders's	5	45	-40	TIE
13	Anders's	45	45	0	Ander's
14	Anders's	65	65	0	Ander's
15	Anders's	55	55	0	Ander's
16	Anders's	15	45	30	TIE
17	Anders's	60	60	0	Ander's
18	Anders's	75	90	15	Ander's
19	Anders's	-55	0	-55	Our checker
20	Anders's	60	0	60	Ander's

4.4 The third experiment

In the third experiment, our checker has won 13 in 20 matches it can be calculated in percentage about 65%. If our checker moves first it had won 7 in 10 matches it's about 70% and when our checker moves next to Ander's our checker won 6 in 10 times it's about 60%. So the percentages are won 65%, tie 20%, lost 15%.

Table 4-4: The third experiment

Match	First Move	Board Score	White Score	Black Score	Match Winner
1	Our checker	-5	55	-60	TIE
2	Our checker	-65	0	-65	Our checker
3	Our checker	-30	0	-30	Our checker
4	Our checker	0	30	-30	TIE
5	Our checker	25	25	0	Ander's
6	Our checker	-50	0	-50	Our checker
7	Our checker	-60	0	-60	Our checker
8	Our checker	-30	0	-30	Our checker
9	Our checker	-45	0	-45	Our checker
10	Our checker	-45	0	-45	Our checker
11	Anders's	-70	0	-70	Our checker
12	Anders's	40	50	-10	Ander's
13	Anders's	-25	0	-25	Our checker
14	Anders's	0	40	-40	TIE
15	Anders's	-55	0	-55	Our checker
16	Anders's	-45	0	-45	Our checker
17	Anders's	-65	10	-75	Our checker
18	Anders's	-30	0	-30	Our checker
19	Anders's	40	40	0	Ander's
20	Anders's	-20	30	-50	TIE

4.5 The fourth experiment

In the fourth experiment, our checker had won 7 in 20 matches it can be calculated in percentage about 35%. If our checker moves first it has won 3 in 10 matches it's about 30% and when our checker moves next to Ander's our checker has won 4 in 10 matches it's about 40%. So the percentages are won 35%, tie 30%, and lost 35%.

Table 4-5: The fourth experiment

Match	First Move	Board Score	White Score	Black Score	Match Winner
1	Our checker	-15	30	-45	TIE
2	Our checker	50	60	-10	Ander's
3	Our checker	15	25	-10	Ander's
4	Our checker	-35	0	-35	Our checker
5	Our checker	-25	30	-55	Our checker
6	Our checker	45	45	0	Ander's
7	Our checker	75	75	0	Ander's
8	Our checker	10	50	-40	TIE
9	Our checker	-15	50	-65	Our checker
10	Our checker	25	40	-15	Ander's
11	Anders's	-40	0	-40	Our checker
12	Anders's	30	55	-25	Ander's
13	Anders's	-40	10	-50	Our checker
14	Anders's	0	30	-30	TIE
15	Anders's	-10	0	-10	Our checker
16	Anders's	-15	10	-25	Our checker
17	Anders's	-10	50	-60	TIE
18	Anders's	50	90	-40	Ander's
19	Anders's	35	65	-30	TIE
20	Anders's	30	45	-15	TIE

Chapter 5

Conclusion and Recommendation

5.1 Conclusion

The authors developed a new heuristic function, named "ScoreFunction", to solve a "depth of search" problem of minimax game playing strategy. We made four different experiments by competing with Ander Bauman's checker program, record and calculated experiments results in percentages. The average percentages of total results are: won 35%, tie 23.75%, and lost 41.25%, it depends on different max depth of search.

5.2 Recommendation

In this special project, we recommend to revision our checker program into Graphic User Interface mode (GUI mode) for more comfortable and easily to play. For further study, there are some interesting problems to be solved.

References

[1] H. Schildt, 1987. **Artificial intelligence using C**. Osborne McGraw-Hill. Berkeley, Calif.

[2] M. Negnevitsky, 2005. **Artificial intelligence : a guide to intelligent systems**. Addison-Wesley, Harlow

[3] M. A. Weiss, 1999. **Data structures & algorithm analysis in Java.** Addison-Wesley, Reading, MA.

[4] S.J. Russell, 2003. **Artificial intelligence : a modern approach.** Pearson Education International, Upper Saddle River

[5] **Anders Baumann's Checker program**. [Online] Available at: http://www.andersbaumann.dk/ (Accessed: 19 March 2010).

[6] **Introducing the Min-Max Algorithm**. [Online] Available at: http://www.progtools.org/games/tutorials/ai_contest/minmax_contest.pdf (Accessed: 13 July 2009).

[7] **The Basic Rules of Checkers**. [Online] Available at: http://www.jimloy.com/checkers/rules2.htm (Accessed: 4 August 2009).

Appendix A

Checker Game Source Code

In our checker we have 5 classes of java codes (Ai.java, BestMove.java, Board.java, CheckerGame.java, and Pos.java)

Our main class is in CheckerGame.java the following are our codes.

```
//////////////////////////////////////
CheckerGame.java
//////////////////////////////////////
import java.util.Scanner;
import java.util.StringTokenizer;
public class CheckerGame
{
   private static final int  MAX_DEPTH = 3;
   public static void main(String[] args) throws NullPointerException
   {
      Scanner input = new Scanner(System.in);
      StringTokenizer st;
      Board checkerGame = new Board();
      checkerGame.initailize();

      System.out.print("Player start the first turn by default\n" +
            "If you want Computer to start first ,enter 2 : ");
      if(input.nextLine().equals("2"))
         checkerGame.setCurrentTurn(Board.BLACK);
      else
         System.out.println("Invalid Enter, So Player start first.");
```

```
while(true)
    {
      // Check if game has ended
      if (checkerGame.isGameEnd())
      {
       System.out.println ("The winner is : " + (checkerGame.findWinner() ==
       checkerGame.WHITE ? "White" : "Black"));
         break;
      }
       System.out.println(checkerGame);
       System.out.println(checkerGame.currentTurn() == Board.WHITE ? "Player Turn(o)" :
       "Computer Turn(x)");
       System.out.print(checkerGame.currentTurn() == Board.WHITE ? "Enter Source Row
      Col and Destination Row Col \n" +
            "E.q. Source[6][1] to Destination[5][2] : 6 1 5 2 \n" +
            " : " : "Computer Thinking.....");
      Pos[] posArr = null;
      Ai AI = new Ai();
      // Get input from human
      if (checkerGame.currentTurn() == Board.BLACK)
      {
        if(checkerGame.chkForceJump(checkerGame).posArr != null)
        {
          BestMove posAi = checkerGame.chkForceJump(checkerGame);
          posArr = posAi.posArr;
        }
```

```
 else
   {
    BestMove posAi = AI.FindBestMove(checkerGame,Board.BLACK,1,MAX_DEPTH);
    posArr = posAi.posArr;
   }
}
else
{
   try
   {
      st = new StringTokenizer(input.nextLine());
      if (st.countTokens() % 2 != 0)
         throw new Exception();
      int moveCount = st.countTokens()/2;
      posArr = new Pos[moveCount];
      for (int i = 0; i < moveCount; i++)
      {
         posArr[i] = new Pos();
         posArr[i].row = Integer.parseInt(st.nextToken());
         posArr[i].col = Integer.parseInt(st.nextToken());
      }
   }
   catch (Exception e)
   {
      System.out.println ("The format of input is invalid.");
      System.out.println ("------------------------------");
      continue;
   }
}
```

```
            if (checkerGame.move(posArr)) // Move success
            {
               System.out.println ("OK");
               System.out.println ("-------------------------");
            }
            else
            {
               System.out.println ("Invalid Move, Enter Again");
               System.out.println ("-------------------------");
            }
         }
      }
   }
////////// End CheckerGame.java //////////
```

```
////////////////////////////////////
BestMove.java
////////////////////////////////////
public class BestMove
{
  public Pos[] posArr;
  public int score;
}
////////////// End BestMove.java ///////////////


////////////////////////////////////
Board.java
////////////////////////////////////
import java.util.ArrayList;

public class Board
{

   public static final int BLANK = 0;
   public static final int WHITE = 1;
   public static final int BLACK = 2;
   public static final int WHITE_HORSE = 3;
   public static final int BLACK_HORSE = 4;

   public static final int WHITE_VALUE = 10;
   public static final int BLACK_VALUE = -10;
   public static final int WHITE_HORSE_VALUE = 15;
   public static final int BLACK_HORSE_VALUE = -15;

   private int [][] board = new int[9][9];
   private int currentTurn = WHITE; // WHITE start the first turn by default
```

```
public int getValue(int value)
{
   if(value == WHITE)
      return WHITE_VALUE;
   if(value == WHITE_HORSE)
      return WHITE_HORSE_VALUE;
   if(value == BLACK)
      return BLACK_VALUE;
   if(value == BLACK_HORSE)
      return BLACK_HORSE_VALUE;
   return BLANK;
}

public Board()
{
}
public void setCurrentTurn(int side){
   currentTurn = side;
}
public Object clone()
{
   Board cloneBoard = new Board();
   cloneBoard.currentTurn = currentTurn;
   for (int row = 0; row < 9; row ++)
      for (int col = 0; col < 9; col++)
         cloneBoard.board[row][col] = board[row][col];
   return cloneBoard;
}

public BestMove chkForceJump(Board board){
   BestMove jumpMove = new BestMove();
```

```
    Ai AI = new Ai();
    ArrayList<Pos[]> jumpList = new ArrayList<Pos[]>();
    for(int row=0;row<9;row++)
      for(int col=0;col<9;col++)
      {
        jumpList.addAll(AI.findAllValidJumps(board, row, col));
      }
    for(int i=0; i<jumpList.size();i++)
      jumpMove.posArr = jumpList.get(i);
    return jumpMove;
  }

  public int findWinner() {
    int countWhite = 0, countBlack = 0;
    for(int row=1;row<=8;row++){
      for(int col=1;col<=8;col++){
        if(board[row][col] == Board.BLACK || board[row][col] == Board.BLACK_HORSE)
          countBlack++;
        else if(board[row][col] == Board.WHITE || board[row][col] ==
Board.WHITE_HORSE)
          countWhite++;
      }
    }
    if(countBlack == 0)
      return Board.WHITE;
    if(countWhite == 0)
      return Board.BLACK;
    Ai AI = new Ai();
    if(AI.findAllValidMoves(this).size() == 0)
      return AI.opponent(currentTurn());
    return 0;
  }
```

```
public boolean isGameEnd() {
    if(findWinner() == Board.WHITE || findWinner() == Board.BLACK)
        return true;
    return false;
}

public int currentTurn()
{
    return currentTurn;
}

private String playerCharactor(int charactor)
{
    switch (charactor)
    {
        case BLANK:
            return " ";
        case WHITE:
            return "o";
        case BLACK:
            return "x";
        case WHITE_HORSE:
            return "0";
        case BLACK_HORSE:
            return "X";
    }
    return "";
}

@Override
public String toString()
```

```
{
   String s = "    +---+---+---+---+---+---+---+---+\n ";
   for(int row=1;row<=8;row++){
      s += "(" + row + ")";
      for(int col=1;col<=8;col++)
            s += "| " + playerCharactor(board[row][col]) + " ";
      s += "| ";
      s += "\n    +---+---+---+---+---+---+---+---+\n ";
   }
   s += "    (1) (2) (3) (4) (5) (6) (7) (8)";
   return s;
}

public int getChecker(int r, int c){
   return board[r][c];
}

public int playerSide (int type)
{
   switch (type)
   {
      case WHITE:
      case WHITE_HORSE:
         return WHITE;
      case BLACK:
      case BLACK_HORSE:
         return BLACK;
      case BLANK:
         return BLANK;
   }
   return BLANK;
}
```

```
private void switchTurn()
{
   currentTurn =  currentTurn == WHITE ? BLACK : WHITE;
}

// return true if possible to move
// return false if the move is invalid
// Change turn to opposite side after moving is completed
public boolean move(Pos [] posArr)
{
   return move (posArr,true);
}

public boolean move(Pos [] posArr, boolean swithTurnAfterMove)
{
   if (posArr == null || posArr.length < 2)
      return false;

   // Check if move outside the board or not
   for (int i = 0; i < posArr.length; i++)
      if (posArr[i].row < 1 || posArr[i].col < 1 || posArr[i].row > 8 || posArr[i].col > 8)
         return false;
   // Check if try moving the correct turn or not
   if (currentTurn != playerSide(board[posArr[0].row][posArr[0].col]))
      return false;
   // Check if move to a BLANK position or not
   for (int i = 1; i < posArr.length; i++)
      if (board[posArr[i].row][posArr[i].col] != BLANK)
         return false;

   // Try walk
```

```
    if (posArr.length == 2 && walk(posArr[0].row,posArr[0].col,posArr[1].row,posArr[1].col))
    {
        changeToHorse(posArr[posArr.length-1].row,posArr[posArr.length-1].col);
        if (swithTurnAfterMove)
            switchTurn();
        return true;
    }
    if (jump(posArr)) // can do multiple jumps
    {
        changeToHorse(posArr[posArr.length-1].row,posArr[posArr.length-1].col);
        if (swithTurnAfterMove)
            switchTurn();
        return true;
    }
    return false;
}

// change to horse if applicable
private void changeToHorse(int r, int c)
{
    // must enter row 1
    if (currentTurn == WHITE && r == 1)
        board[r][c] = WHITE_HORSE;
    else if (currentTurn == BLACK && r == 8)
        board[r][c] = BLACK_HORSE;
}

private boolean walk(int sr, int sc, int dr, int dc)
{
    // Check if possible to walk or not
    // Assume that WHITE start from bottom of array (higher row move to lower row)
    if (Math.abs(sc-dc) != 1)
```

```
        return false;
    int rowDiff = dr - sr;
    if (board[sr][sc] == WHITE && rowDiff != -1)
        return false;
    if (board[sr][sc] == BLACK && rowDiff != 1)
        return false;
    if (board[sr][sc] == WHITE_HORSE && Math.abs(rowDiff) != 1)
        return false;
    if (board[sr][sc] == BLACK_HORSE && Math.abs(rowDiff) != 1)
        return false;

    // Rule checking complete. Now make an actual walk
    board[dr][dc] = board[sr][sc];
    board[sr][sc] = BLANK;
    return true;
}

private boolean jump(Pos [] posArr)
{
    // Keep initial state of board
    int [][] initialBoard = new int[9][9];
                for (int row = 0; row < 9; row++)
                        for (int col = 0; col < 9; col++)
                                initialBoard[row][col] = board[row][col];
                for (int i = 0; i < posArr.length-1; i++)
      if (!jump (posArr[i].row,posArr[i].col,posArr[i+1].row,posArr[i+1].col))
      {
        board = initialBoard; // set to previous state
        return false;
      }
    return true;
}
```

```
private boolean jump(int sr, int sc, int dr, int dc)
{
   // Check if possible to jump or not
   // Assume that WHITE start from bottom of array (higher row move to lower row)
   if (Math.abs(sc-dc) != 2)
      return false;
   int rowDiff = dr - sr;
   if (board[sr][sc] == WHITE && rowDiff != -2)
      return false;
   if (board[sr][sc] == BLACK && rowDiff != 2)
      return false;
   if (board[sr][sc] == WHITE_HORSE && Math.abs(rowDiff) != 2)
      return false;
   if (board[sr][sc] == BLACK_HORSE && Math.abs(rowDiff) != 2)
      return false;

   //  Check if there is an opponent player on the eaten position or not
   int eatRow = sr + (dr-sr)/2;
   int eatCol = sc + (dc-sc)/2;
   if (board[eatRow][eatCol] == BLANK)
      return false;
   if (currentTurn == playerSide(board[eatRow][eatCol]))
      return false;

   // Perform jumping
   board[eatRow][eatCol] = BLANK;
   board[dr][dc] = board[sr][sc];
   board[sr][sc] = BLANK;
   return true;
}
```

```
public void initailize()
{
   clearBoard();
   initializeBlack();
   initializeWhite();
}

private void clearBoard()
{
   for(int row=1;row<=4;row++)
      for(int col=2;col<=8;col++)
         board[row][col] = BLANK;
}

private void initializeBlack()
{
   for(int row=1;row<=4;row+=2){
      for(int col=2;col<=8;col+=2){
         board[row][col] = BLACK;
      }
   }
   for(int col=1;col<=7;col+=2){
      board[2][col] = BLACK;
   }
}

private void initializeWhite() {
   for(int row=6;row<=8;row+=2){
      for(int col=1;col<=7;col+=2){
         board[row][col] = WHITE;
      }
```

```
        }
        for(int col=2;col<=8;col+=2){
            board[7][col] = WHITE;
        }
    }
}
////// End Board.java /////////
```

```
////////////////////////////////////
Ai.java
////////////////////////////////////
import java.util.Random;
import java.util.ArrayList;

public class Ai {
   int finalScore = 0;
   int listSize = 0;
   int validMovesCount = 0;
   Board board = new Board();
   Random random = new Random();
private Pos[] routeOffset = new Pos[]
   {
      new Pos(-1,-1),
      new Pos(1,1),
      new Pos(-1,1),
      new Pos(1,-1)
   };

   ArrayList<Pos[]> forceRouteList = new ArrayList<Pos[]>();

   public Pos[] findAllRoute(int row, int col)
   {
      Pos[] forceArr = new Pos[4];
      forceArr[0] = new Pos();
      forceArr[1] = new Pos();
      forceArr[2] = new Pos();
      forceArr[3] = new Pos();
      for(int ioff=0; ioff<routeOffset.length; ioff++)
      {
         forceArr[ioff].row = row + routeOffset[ioff].row;
```

```
        forceArr[ioff].col = row + routeOffset[ioff].col;
    }
    return forceArr;
}

public boolean countCheckerAndKing(final Board board)
{
    int white = 0;
    int black = 0;
    boolean king = false;
    for(int row=0;row<9;row++)
        for(int col=0;col<9;col++)
        {
            if (board.playerSide(board.getChecker(row, col)) == Board.WHITE)
                white++;
            if (board.playerSide(board.getChecker(row, col)) == Board.BLACK)
                black++;
            if (board.getChecker(row, col) == Board.BLACK_HORSE)
                king = true;
        }
    if(black > white && king)
        return true;
    else
        return false;
}

public void tryToEatFunc(final Board board) {
    for(int row=1;row<=8;row++)
        for(int col=1;col<=8;col++)
        {
            if(row>1 && row<8 && col>1 && col<8)
```

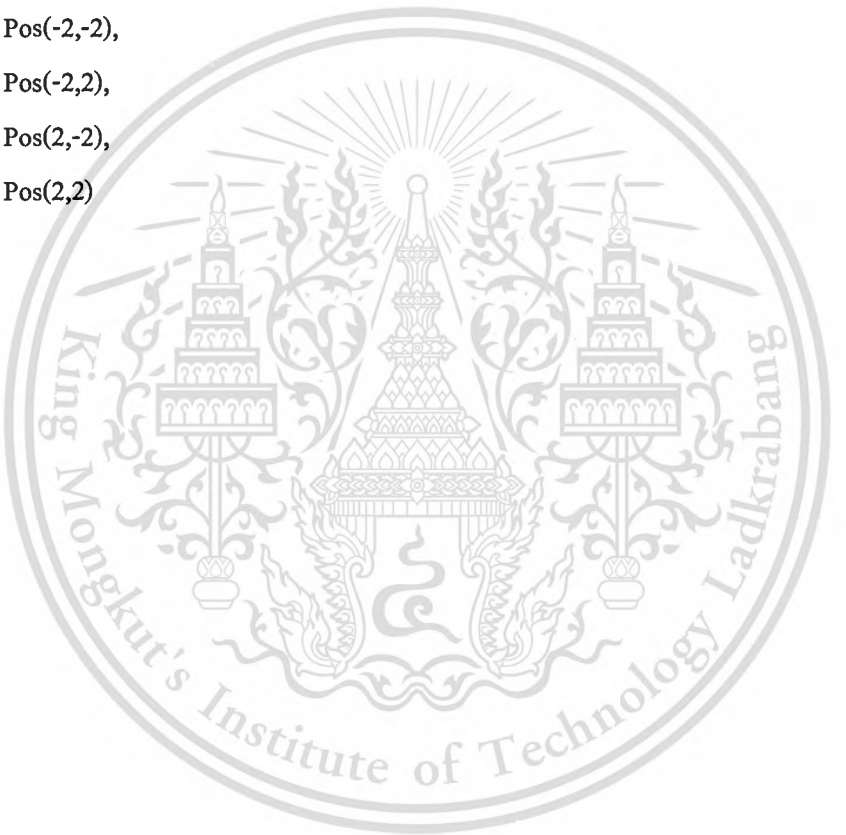
```
            if(board.getChecker(row, col) == Board.WHITE || board.getChecker(row, col) ==
Board.WHITE_HORSE)
            {
              Pos[] list = findAllRoute(row,col);
              forceRouteList.add(list);
            }
        }
    }

    public int scoreFunc(final Board board){
        int score = 0;
        for(int row=1;row<=8;row++){
            for(int col=1;col<=8;col++){
                score += board.getValue(board.getChecker(row, col));
                if(board.getChecker(row, col) == Board.BLACK)
                {
                  if(row == 1 || row == 8 || col == 1 || col == 8)
                    score += board.getValue(board.getChecker(row, col)) - 4;
                  if(row == 2 || row == 7)
                    if(col > 1 || col < 8)
                      score += board.getValue(board.getChecker(row, col)) - 3;
                  if(col == 2 || col == 7)
                    if(row > 1 || row < 8)
                      score += board.getValue(board.getChecker(row, col)) - 3;
                  if(row == 3 || row == 6)
                    if(col > 2 || col < 7)
                      score += board.getValue(board.getChecker(row, col)) - 2;
                  if(col == 3 || col == 6)
                    if(row > 2 || row < 7)
                      score += board.getValue(board.getChecker(row, col)) - 2;
                }
                if(board.getChecker(row, col) == Board.BLACK && row == 5 )
```

```
            score += board.getValue(board.getChecker(row, col)) -2;
          if(board.getChecker(row, col) == Board.BLACK && row == 6 )
            score += board.getValue(board.getChecker(row, col)) -3;
          if(board.getChecker(row, col) == Board.BLACK && row == 7)
            score += board.getValue(board.getChecker(row, col)) - 4;
          if(board.getChecker(row, col) == Board.BLACK && row == 8)
            score += board.getValue(board.getChecker(row, col)) - 5;

          if(board.getChecker(row, col) == Board.BLACK_HORSE &&
countCheckerAndKing(board))
              if(!forceRouteList.isEmpty())
              {
                for(int i=0;i<forceRouteList.size();i++)
                {
                  Pos[] forceArr = new Pos[forceRouteList.size()];
                  forceArr = forceRouteList.get(i);
                  for(int j=0;j<forceArr.length;j++)
                  {
                    int forceRow = forceArr[j].row;
                    int forceCol = forceArr[j].col;

                    if(row == forceRow && col == forceCol)
                    score += board.getValue(board.getChecker(row, col)) -14;
                  }
                }
              }
        }
      }
      return score;
    }

    public int opponent(int color){
```

```
    if(color == Board.BLACK)
        return Board.WHITE;
    return Board.BLACK;
  }

  private static int MAX_SCORE = 1000;
  private static int MIN_SCORE = -1000;

  public BestMove FindBestMove (final Board board, int color, int currentDepth, int maxDepth)
throws NullPointerException
  {
    // If game end
    if (board.isGameEnd())
    {
      BestMove bestMove = new BestMove();
      if (board.findWinner() == board.BLACK)
        bestMove.score = -1000;
      else if (board.findWinner() == board.WHITE)
        bestMove.score = 1000;
      return bestMove;
    }
    else if (currentDepth == maxDepth)
    {
      BestMove bestMove = new BestMove();
      if(countCheckerAndKing(board))
        tryToEatFunc(board);
      bestMove.score = scoreFunc(board);
      finalScore = bestMove.score;
      return bestMove;
    }

    ArrayList<Pos[]> allNextMovesList = findAllValidMoves(board);
```

```
    // Black try to minimize
    // White try to maximize
    int bestScore = color == Board.BLACK ? MAX_SCORE : MIN_SCORE;
    BestMove bestMove = null;
    for (int i = 0 ; i < allNextMovesList.size(); i++)
    {
      Board cloneBoard = (Board)board.clone();
      if (!cloneBoard.move(allNextMovesList.get(i)))
      {
        //System.out.println ("Bug is found !!");
        continue;
      }
      BestMove move = new BestMove();
      move = FindBestMove (cloneBoard, opponent(color), currentDepth+1, maxDepth);

      if ((color == Board.BLACK && move.score <= bestScore) ||
        (color == Board.WHITE && move.score >= bestScore))
      {
        if (move.score == bestScore)
          if (random.nextBoolean()){
            continue;
          }
        bestScore = move.score;
        bestMove = move;
        bestMove.posArr = allNextMovesList.get(i);
      }
    }
    return bestMove;
  }


  private Pos[] walkOffset = new Pos[]
```

```
{
    new Pos(-1,-1),
    new Pos(-1,1),
    new Pos(1,-1),
    new Pos(1,1)
};
private Pos[] jumpOffset = new Pos[]
{
    new Pos(-2,-2),
    new Pos(-2,2),
    new Pos(2,-2),
    new Pos(2,2)
};
```



```
public ArrayList<Pos[]> findAllValidMoves(Board board)
  {
    ArrayList<Pos[]> moveList = new ArrayList<Pos[]>();
    for (int row = 1; row <= 8; row++)
      for (int col = 1; col <= 8; col++)
      {
        if (board.playerSide(board.getChecker(row, col)) == board.currentTurn())
        {
          // Find all possible walks of the chess at row,col
          moveList.addAll(findAllValidWalks(board,row,col));
          // Find all possible jumps of the chess at row,col
          moveList.addAll(findAllValidJumps(board,row,col));
        }
      }
    return moveList;
  }

  // Find all valid walks of the player at row == sr , col == sc
  public ArrayList<Pos[]> findAllValidWalks(Board board, int sr, int sc)
  {
    ArrayList<Pos[]> walkList = new ArrayList<Pos[]>();
    Pos srcPos = new Pos(sr,sc);
    for (int ioff = 0; ioff < walkOffset.length; ioff++)
    {
      Pos dstPos = new Pos(sr+walkOffset[ioff].row,sc+walkOffset[ioff].col);
      Board cloneBoard = (Board)board.clone();
      Pos[] walkArr = new Pos[]{srcPos,dstPos};
      if (cloneBoard.move(walkArr)) // Check if possible to walk
        walkList.add(walkArr);
    }
    return walkList;
  }
```

```
// Find all valid jumps of the player at row == sr , col == sc
// This include double jumps, triple jumps, and so on
public ArrayList<Pos[]> findAllValidJumps(Board board, int sr, int sc)
{
   ArrayList<Pos[]> jumpList = new ArrayList<Pos[]>();
   Pos srcPos = new Pos(sr,sc);
   for (int ioff = 0; ioff < jumpOffset.length; ioff++)
   {
      Pos dstPos = new Pos(sr+jumpOffset[ioff].row,sc+jumpOffset[ioff].col);
      Board cloneBoard = (Board)board.clone();
      Pos[] jumpArr = new Pos[] {srcPos,dstPos};
      if (cloneBoard.move(jumpArr,false)) // Check if possible to jump
      {
         // Add single jump to list first
         jumpList.add(jumpArr);
         // Find multiple jumps if possible
         ArrayList<Pos[]> multiJumpList =
findAllValidJumps(cloneBoard,dstPos.row,dstPos.col);
         for (int i = 0; i < multiJumpList.size(); i++)
         {
            Pos[] tmpJumpArr = new Pos[multiJumpList.get(i).length+1];
            tmpJumpArr[0] = srcPos;
            tmpJumpArr[1] = dstPos;
            for (int j = 1; j < multiJumpList.get(i).length; j++)
               tmpJumpArr[j+1] = multiJumpList.get(i)[j];
            jumpList.add(tmpJumpArr);
         }
      }
   }
   return jumpList;
}
```

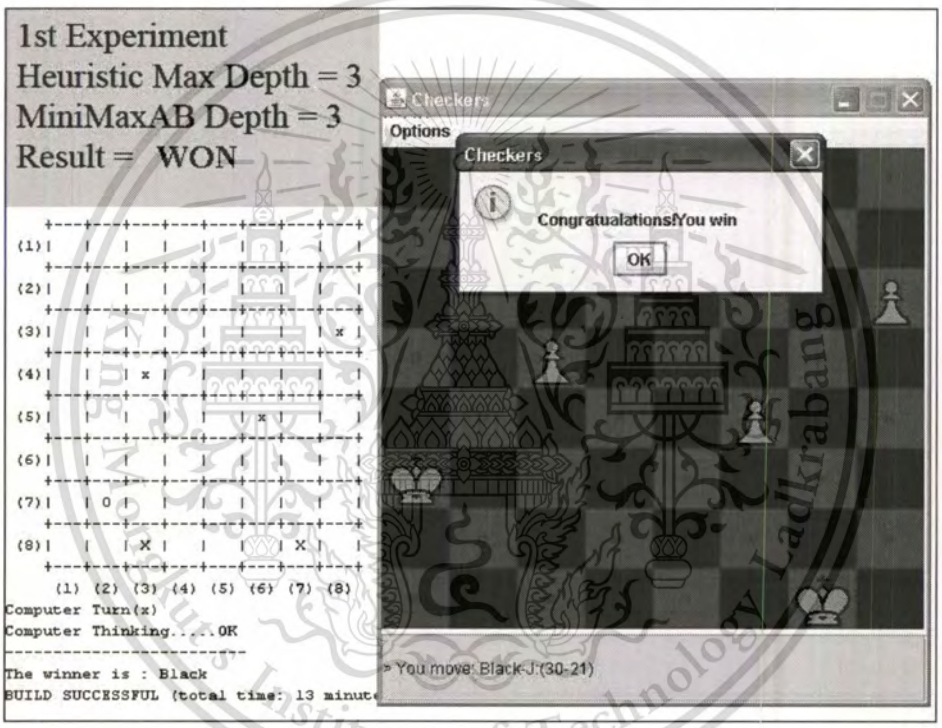
```
}
//////// End Ai.java //////////

////////////////////////////////////
Pos.java
////////////////////////////////////
public class Pos
{
   public Pos()
   {
   }

   public Pos (int row, int col)
   {
      this.row = row;
      this.col = col;
   }
   public int row;
   public int col;
}
////// End Pos.java //////////
```

Appendix B

Checker Endgame Transcript

The first experiment

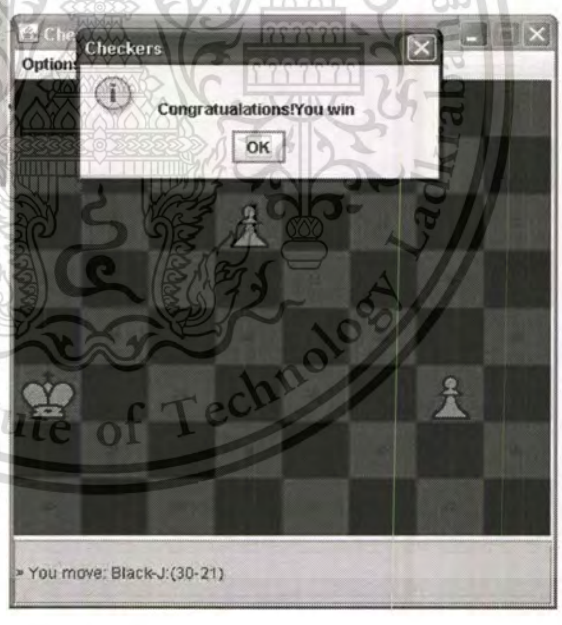
Our checker move first



2rd Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = LOST


```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   | 0 |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   | o |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   | X |   |   |   | o |
    +---+---+---+---+---+---+---+---+
(8)| o |   |   |   | 0 |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Row
E.g. Source[6][1] to Destination[5][2] :
 : 8 5 6 3
OK
--------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 12 minutes 38 seconds)
```

3rd Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = WON




```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   | x |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   | x |   |
    +---+---+---+---+---+---+---+---+
(7)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   | X |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
--------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 20 minutes 10 seconds)
```

4th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = WON

```
    +---+---+---+---+---+---+---+---+
(1)|   | x |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)| o |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   | X |
    +---+---+---+---+---+---+---+---+
(6)| x |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | O |   | X |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 77 minutes 44 seconds)
```

5th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = LOST

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   | O |   |   |   | X |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   | O |   | O |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   | o |   | o |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | o |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Row
E.g. Source[6][1] to Destination[5][2] :
 : 4 5 2 7
OK
-------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 25 minutes 22 seconds)
```

6th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = WON

```
    +---+---+---+---+---+---+---+---+
 (1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (4)| x |   |   |   |   | X |   |   |
    +---+---+---+---+---+---+---+---+
 (5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (7)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (8)|   |   | X |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 67 minutes 5 seconds)
```

7th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = WON

```
    +---+---+---+---+---+---+---+---+
 (1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)|   |   | x |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   |   |   |   |   |   |   | x |
    +---+---+---+---+---+---+---+---+
 (4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (6)|   |   | x |   |   |   | x |   |
    +---+---+---+---+---+---+---+---+
 (7)|   | 0 |   |   |   |   |   | X |
    +---+---+---+---+---+---+---+---+
 (8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 9 minutes 39 seconds)
```

8th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = WON


```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   | X |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | O |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   | X |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
--------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 30 minutes 3 seconds)
```

9th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = LOST


```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   | x |   |
    +---+---+---+---+---+---+---+---+
(3)|   | o |   | O |   |   |   | o |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   | o |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   | o |   | o |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Row
E.g. Source[6][1] to Destination[5][2] :
 : 3 8 1 6
OK
--------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 5 minutes 40 seconds)
```

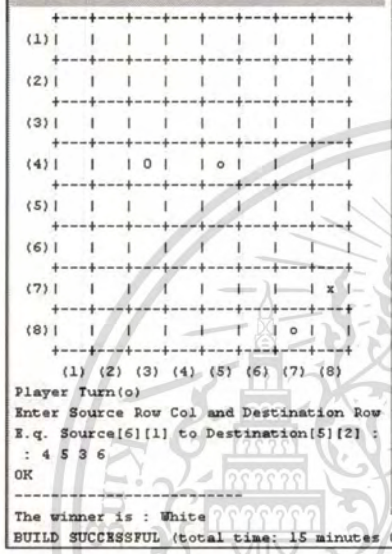
10th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = LOST

```
    +---+---+---+---+---+---+---+---+
(1)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)| o |   |   |   | x |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   | o |   |   |
    +---+---+---+---+---+---+---+---+
(8)| o |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Row
E.g. Source[6][1] to Destination[5][2] :
 : 7 6 5 4
OK
--------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 5 minutes 59 seconds)
```

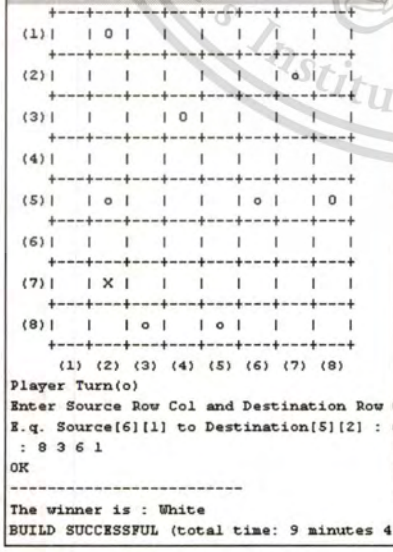
The first experiment

Ander Baumann's checker move first.

1st Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = LOST



2rd Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = LOST

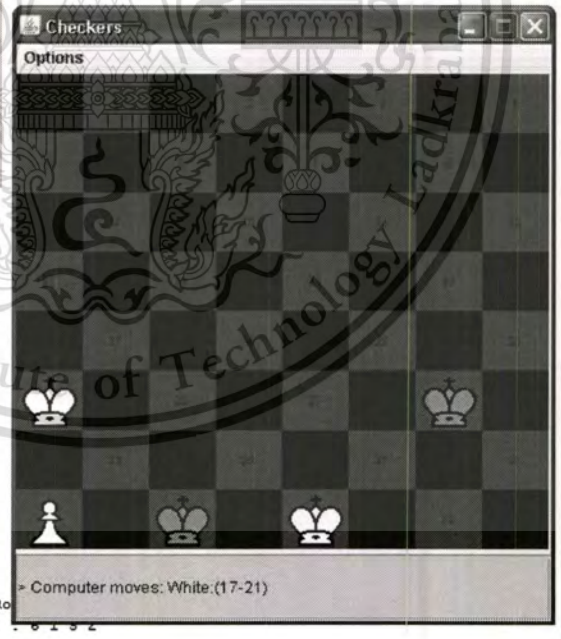


3rd Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = TIE
(-30 + 45 = 15)

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   | O |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | O |   |   |   | X |   | X |
    +---+---+---+---+---+---+---+---+
(8)|   |   | O |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Ro
E.g. Source[6][1] to Destination[5][2]
 :
```

4th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = TIE
(-30 + 40 = 10)

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   | O |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   | X |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)| o |   | X |   | O |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Ro
E.g. Source[6][1] to Destination[5][2]
 :
```



5th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = LOST

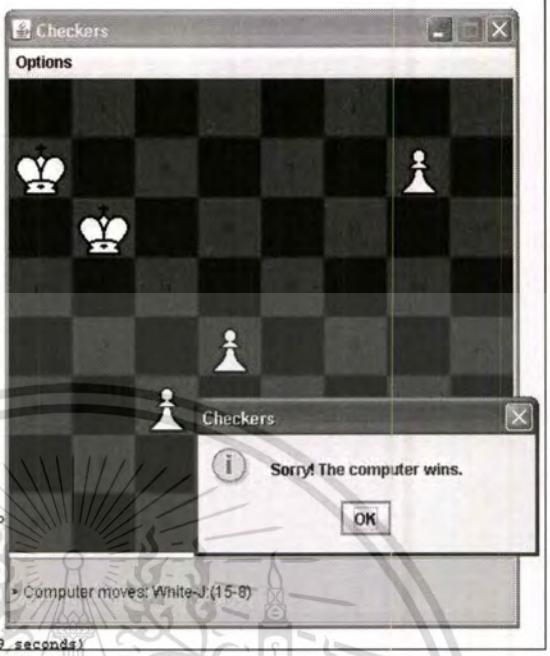
```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   | 0 |   | x |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   | o |   |   |   | o |
    +---+---+---+---+---+---+---+---+
(4)|   |   | o |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   | o |   | o |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   | o |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination R
E.g. Source[6][1] to Destination[5][2]
 : 3 4 1 6
OK
--------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 15 minutes 24 seconds)
```

6th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = LOST

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | X |   |   |   | o |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   | 0 |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination R
E.g. Source[6][1] to Destination[5][2]
 : 8 3 6 1
OK
--------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 15 minutes 21 seconds)
```

7th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = LOST

```
    +---+---+---+---+---+---+---+---+
 (1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)| 0 |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   | 0 |   |   |   | X |   |   |
    +---+---+---+---+---+---+---+---+
 (4)|   |   |   |   | o |   |   |   |
    +---+---+---+---+---+---+---+---+
 (5)|   |   |   | o |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (6)|   |   | o |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (7)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Row Co
E.g. Source[6][1] to Destination[5][2] : 6
 : 4 5 2 7
OK
---------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 10 minutes 19 seconds)
```



8th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = TIE
(-70 + 30 = -40)

```
    +---+---+---+---+---+---+---+---+
 (1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   |   |   |   |   |   |   | x |
    +---+---+---+---+---+---+---+---+
 (4)|   |   |   |   |   |   | X |   |
    +---+---+---+---+---+---+---+---+
 (5)|   |   |   | X |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (6)|   |   |   |   | X |   | X |   |
    +---+---+---+---+---+---+---+---+
 (7)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (8)| 0 |   | 0 |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination R
E.g. Source[6][1] to Destination[5][2]
 : BUILD STOPPED (total time: 43 minutes 59 seconds)
```

9th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = LOST

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)| o |   |   |   |   |   | x |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   | o |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   | 0 |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Ro
E.g. Source[6][1] to Destination[5][2]
 : 7 6 5 8
OK
-------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 9 minutes 6 seconds)
```

10th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 3
Result = TIE
(-25 + 70 = 45)

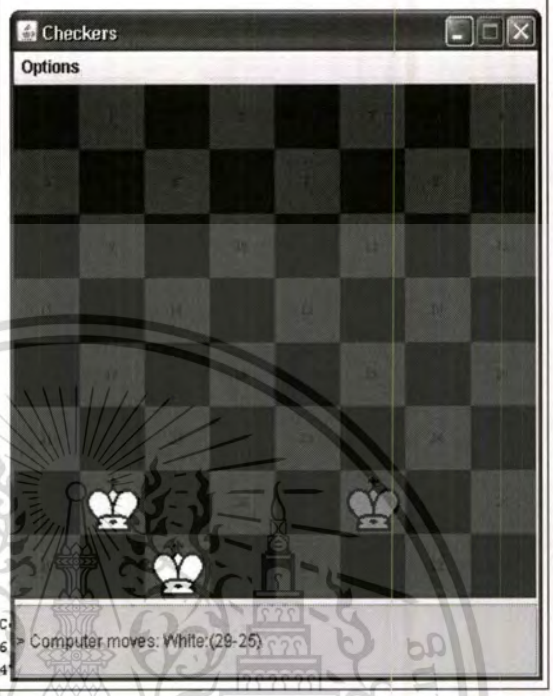
```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)| 0 |   |   |   |   |   | o |   |
    +---+---+---+---+---+---+---+---+
(3)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   | o |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   | o |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)| x |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   |   |   | X |
    +---+---+---+---+---+---+---+---+
(8)|   |   | o |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Row Col
E.g. Source[6][1] to Destination[5][2] : 6 1 5 2
 :
```

The second experiment

Our checker move first

1st Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 5
Result = TIE
(-15 + 30 = 15)

```
   +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   | X |   |   |
   +---+---+---+---+---+---+---+---+
(8)| O |   | O |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Row C
E.g. Source[6][1] to Destination[5][2] : 6
: BUILD STOPPED (total time: 14 minutes 4
```



2rd Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 5
Result = LOST

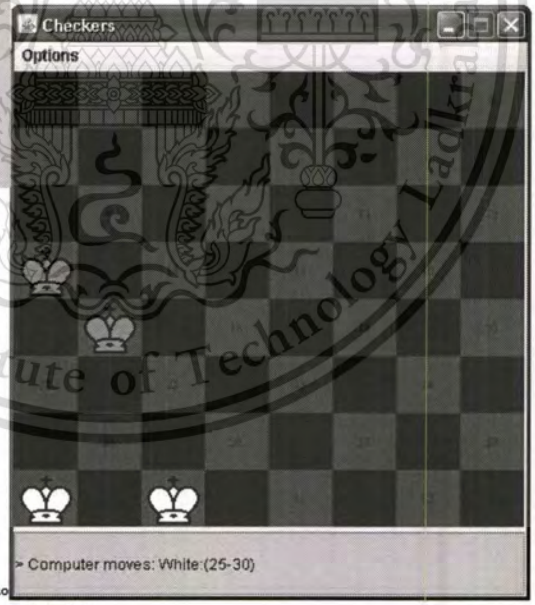
```
   +---+---+---+---+---+---+---+---+
(1)|   | O |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
(2)|   |   |   |   | O |   |   |   |
   +---+---+---+---+---+---+---+---+
(3)|   |   |   | o |   |   |   | o |
   +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
(5)|   | x |   |   |   |   |   | o |
   +---+---+---+---+---+---+---+---+
(6)|   |   | o |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
(7)|   | o |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
(8)| o |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination R
E.g. Source[6][1] to Destination[5][2]
 : 6 3 4 1
OK
--------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 9 minutes 1 second)
```

3rd Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 5
Result = LOST

```
    +---+---+---+---+---+---+---+---+
 (1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)| 0 |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (5)|   |   |   |   |   | o |   |   |
    +---+---+---+---+---+---+---+---+
 (6)|   |   | 0 |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (7)|   | X |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination R
E.g. Source[6][1] to Destination[5][2]
 : 6 3 8 1
OK
--------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 13 minutes 54 seconds)
```

4th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 5
Result = TIE
(-30 + 30 = 0)

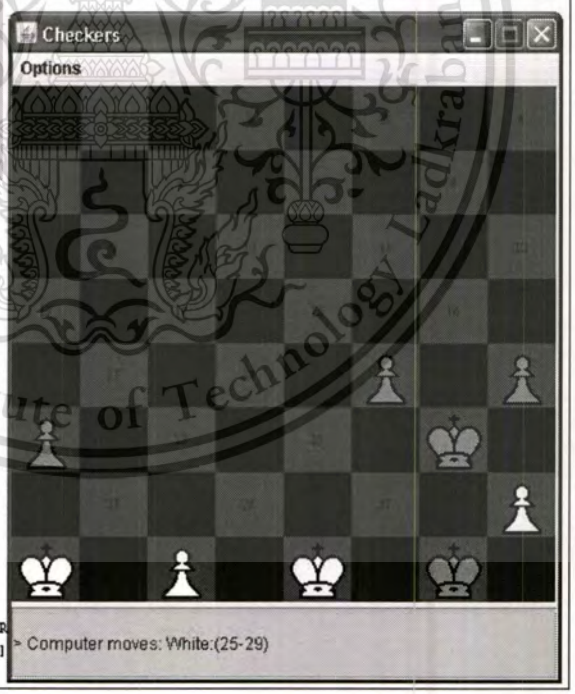
```
    +---+---+---+---+---+---+---+---+
 (1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (4)| X |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (5)|   | X |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (7)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (8)| 0 |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Ro
E.g. Source[6][1] to Destination[5][2] : 6 1 5 2
 : BUILD STOPPED (total time: 13 minutes 15 seconds)
```



5th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 5
Result = WON


```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   | x |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)| x |   |   |   | x |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)| x |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 16 minutes 33 seconds)
```

6th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 5
Result = TIE
(-60 + 50 = -10)




```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   | x |   | x |
    +---+---+---+---+---+---+---+---+
(6)| x |   |   |   |   |   | X |   |
    +---+---+---+---+---+---+---+---+
(7)|   | 0 |   |   |   |   |   | o |
    +---+---+---+---+---+---+---+---+
(8)|   |   | o |   | 0 |   | X |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination R
E.g. Source[6][1] to Destination[5][2]
 :
```

7th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 5
Result = LOST

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   | O |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   | X |   |   |
    +---+---+---+---+---+---+---+---+
(8)| O |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination R
E.g. Source[6][1] to Destination[5][2]
 : 6 5 8 7
OK
--------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 13 minutes 44 seconds)
```

8th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 5
Result = LOST

```
--------------------------
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   | x |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   | o |   | O |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   | O |   |   |   | x |
    +---+---+---+---+---+---+---+---+
(8)|   |   | o |   | o |   | o |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination R
E.g. Source[6][1] to Destination[5][2]
 : 6 3 4 1
OK
--------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 6 minutes 54 seconds)
```

9th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 5
Result = TIE
(-40 + 50 = 10)

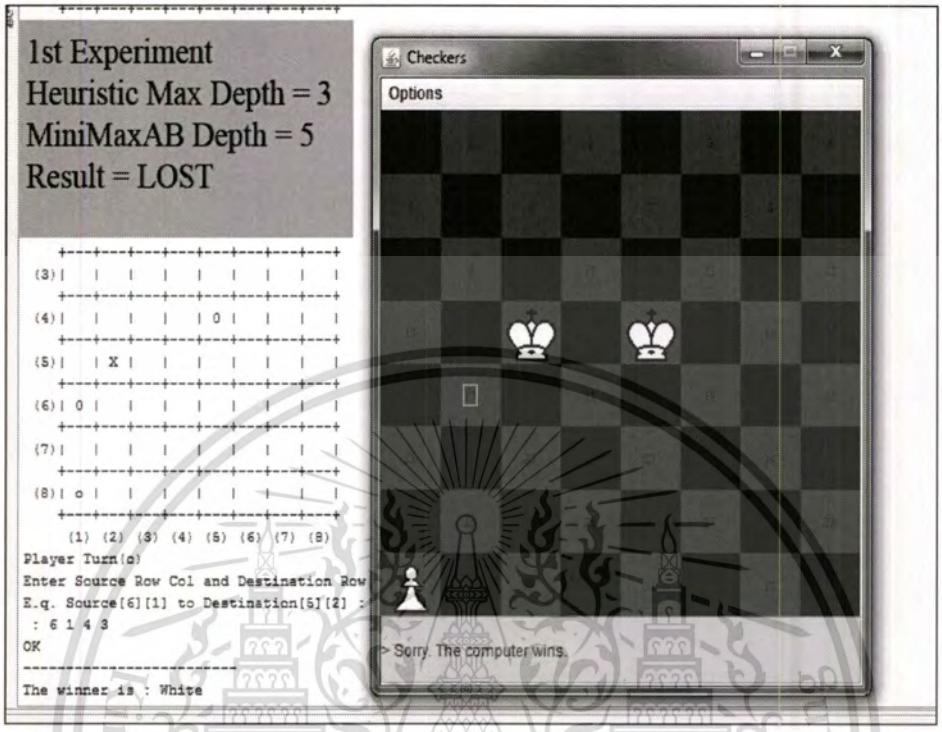

```
    +---+---+---+---+---+---+---+---+
(1)|   | x |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)| o |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   | o |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   | X |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   |   |   | X |
    +---+---+---+---+---+---+---+---+
(8)| O |   |   |   | O |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Row Col
E.g. Source[6][1] to Destination[5][2] : 6 1 5 2
 : BUILD STOPPED (total time: 13 minutes 6 seconds)
```

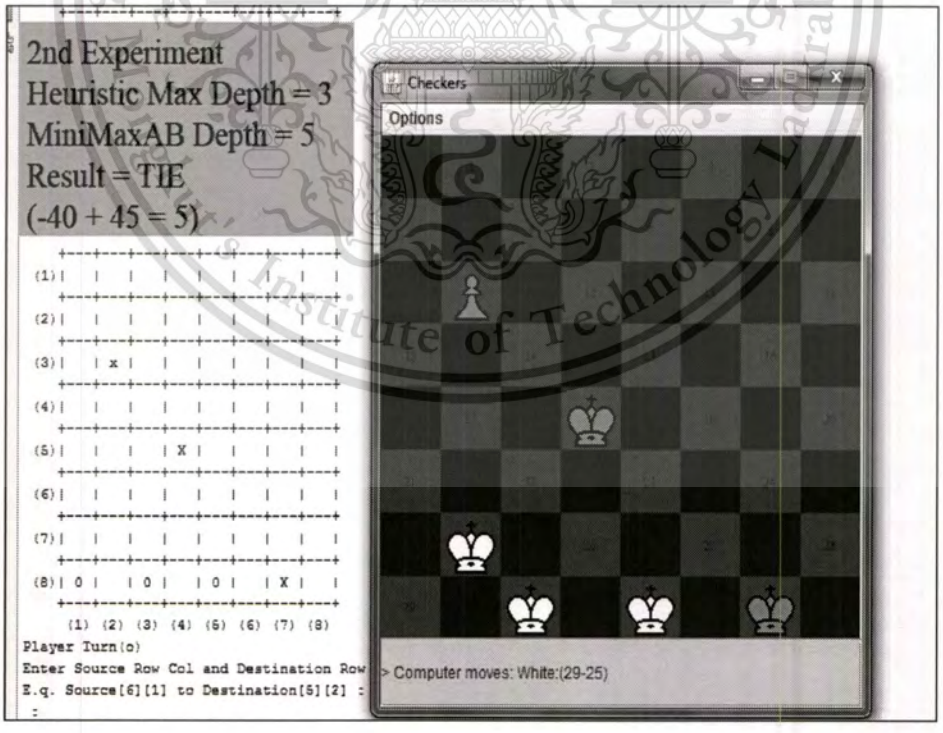
10th Experiment
Heuristic Max Depth = 3
MiniMaxAB Depth = 5
Result = LOST

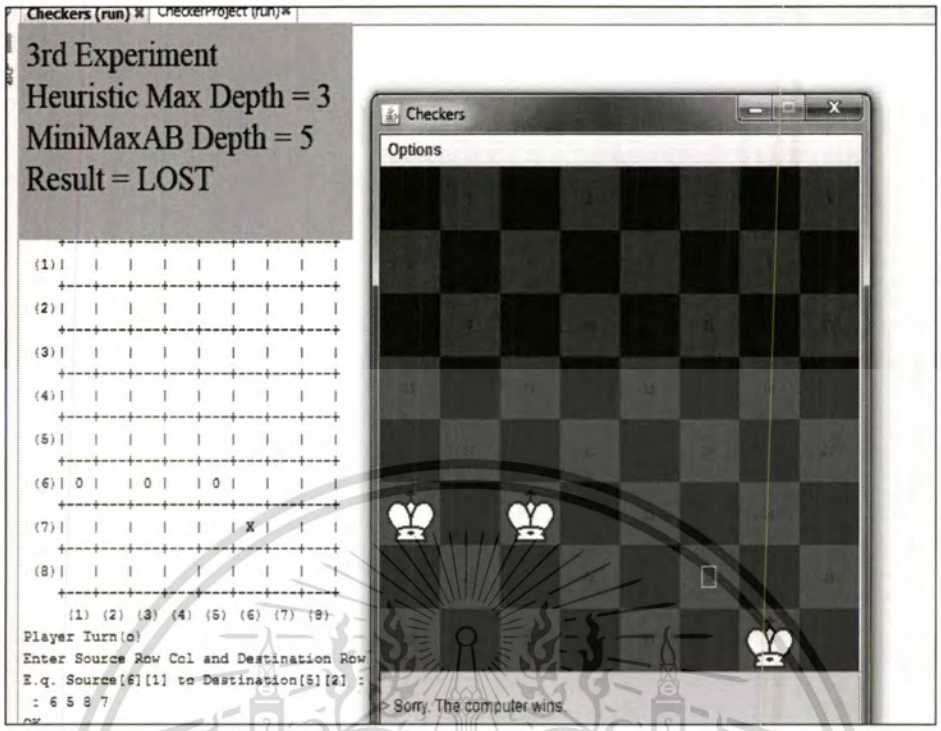

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   | O |
    +---+---+---+---+---+---+---+---+
(2)| o |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   | O |   |   |
    +---+---+---+---+---+---+---+---+
(4)| O |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | x |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   | o |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Po
E.g. Source[6][1] to Destination[5][2]
 : 8 3 6 1
OK
-------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 16 minutes 33 seconds)
```

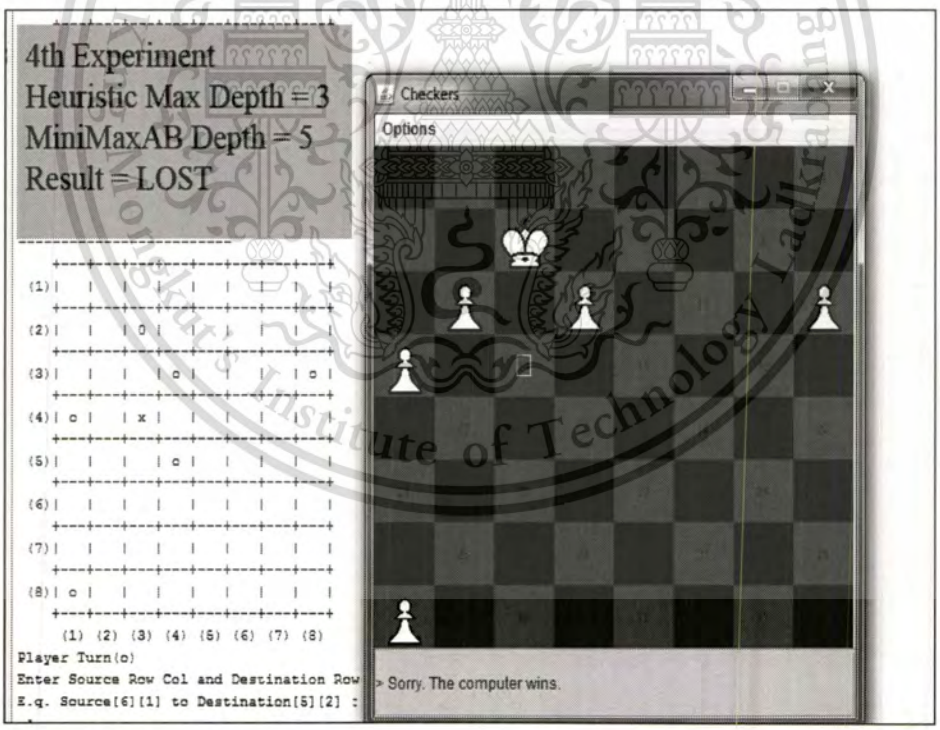
The second experiment

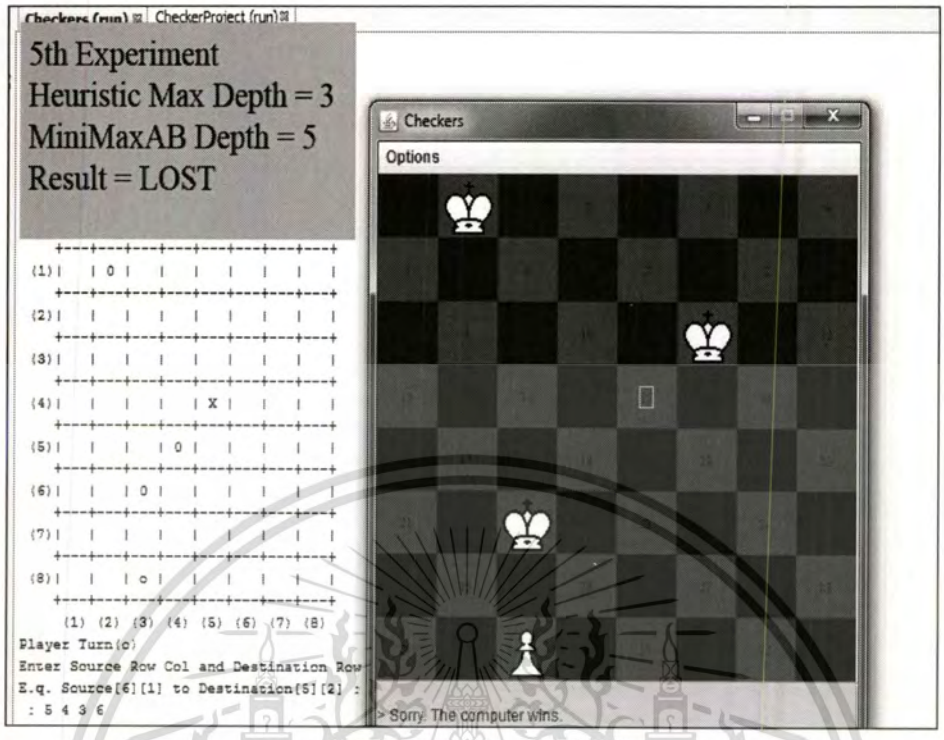
Ander Baumann's checker move first.

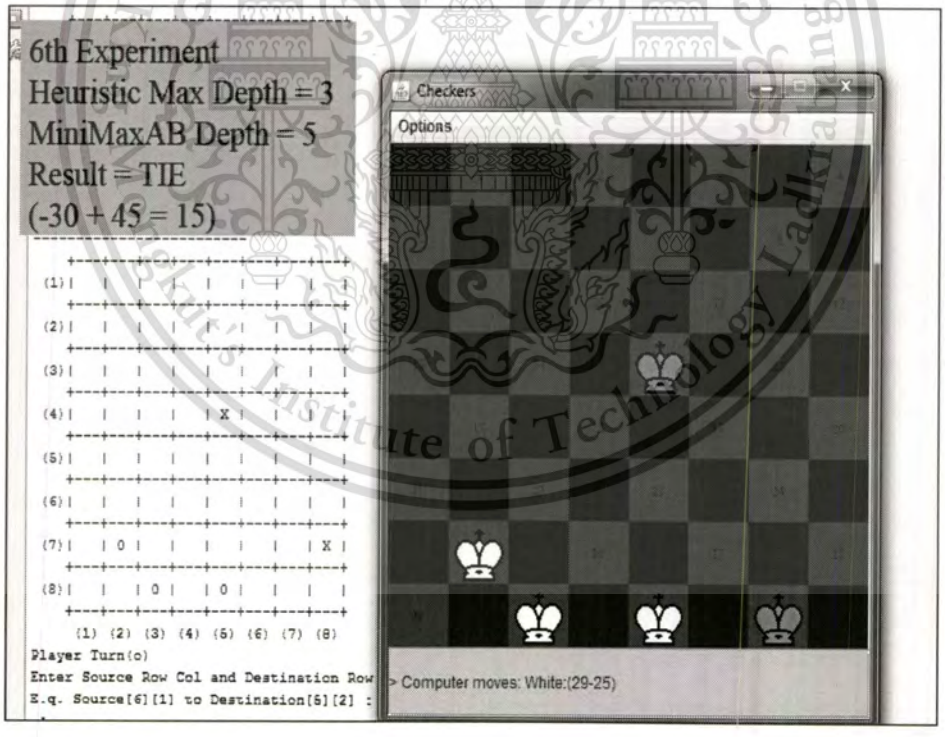


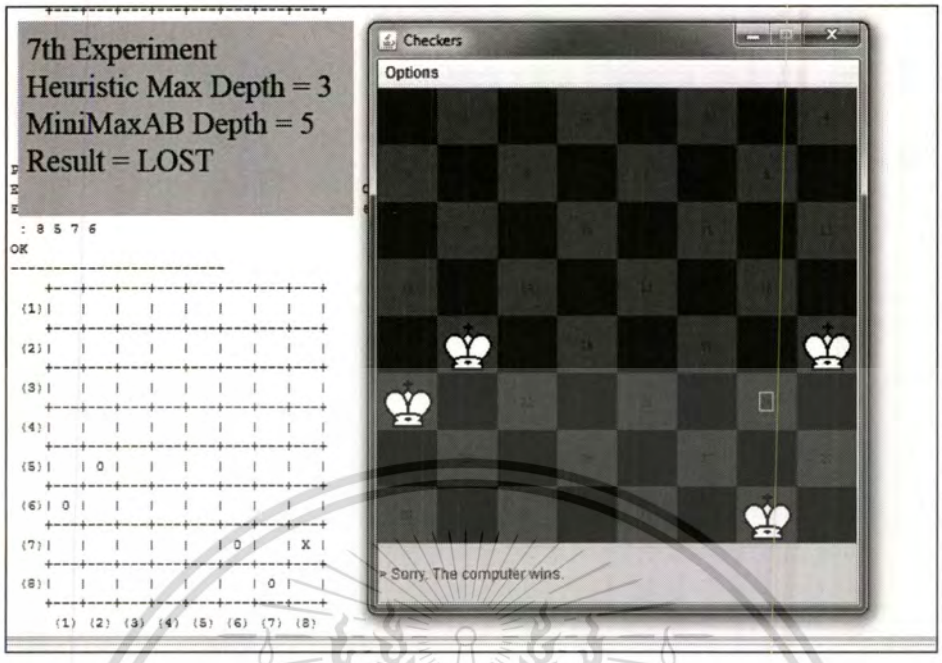


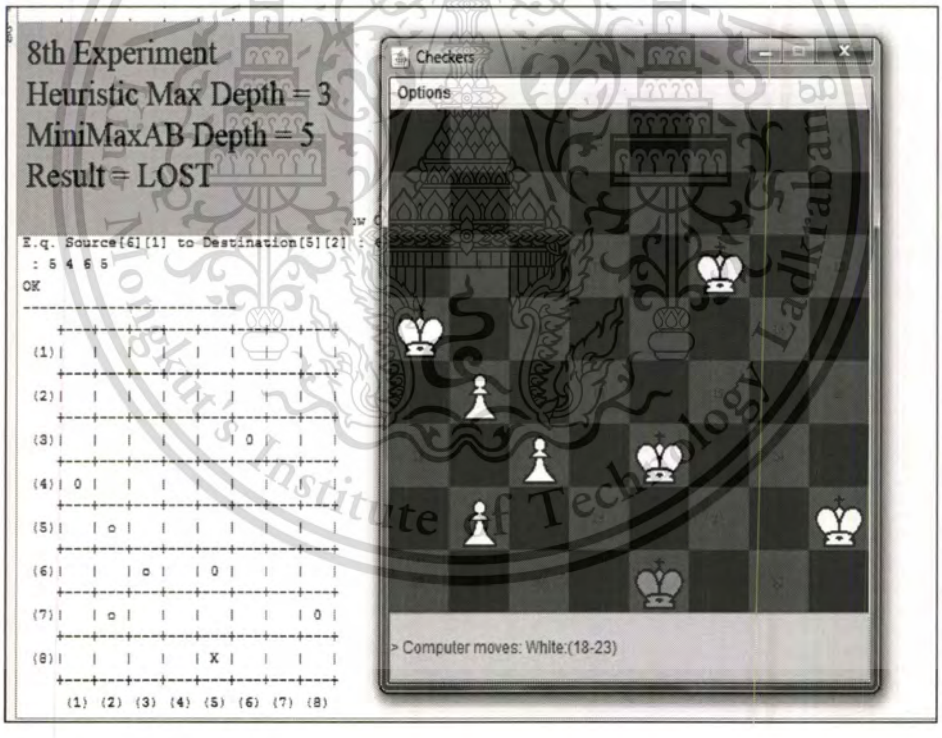


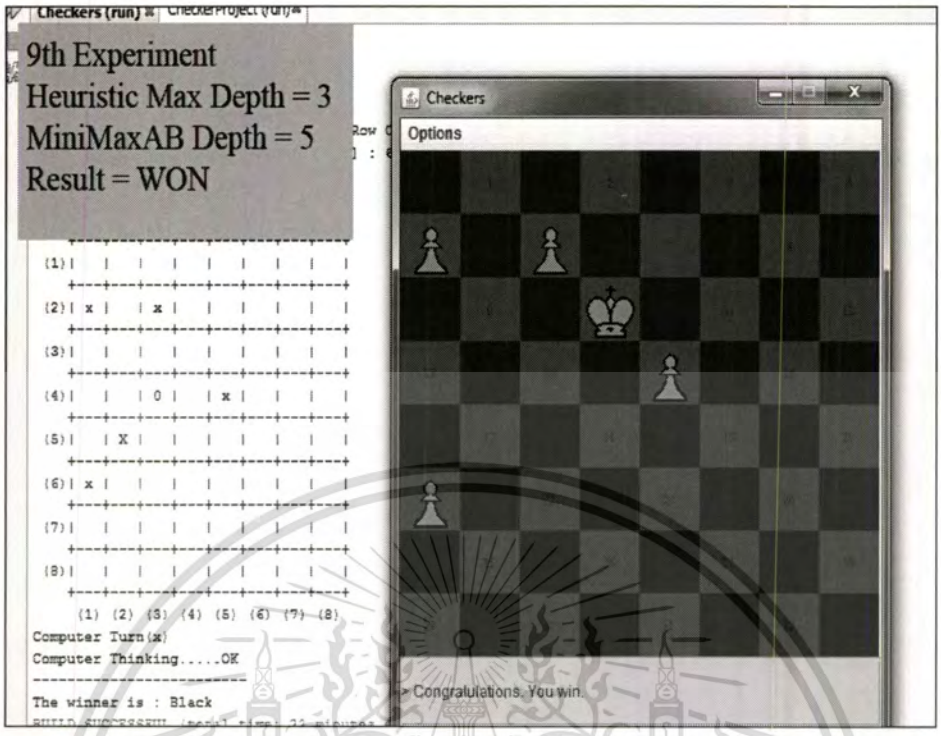


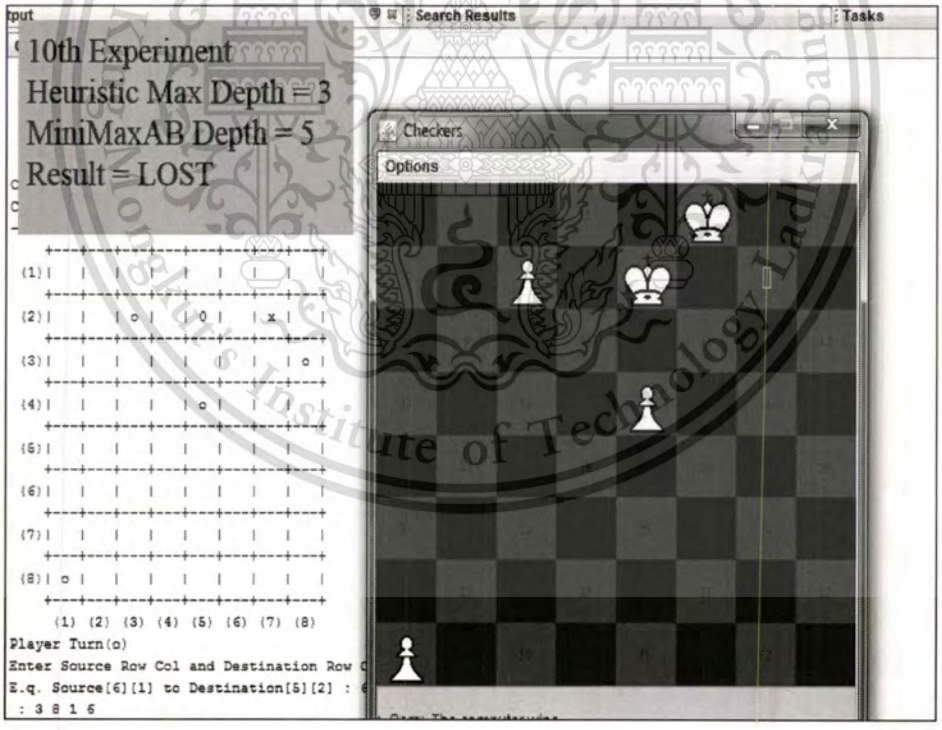








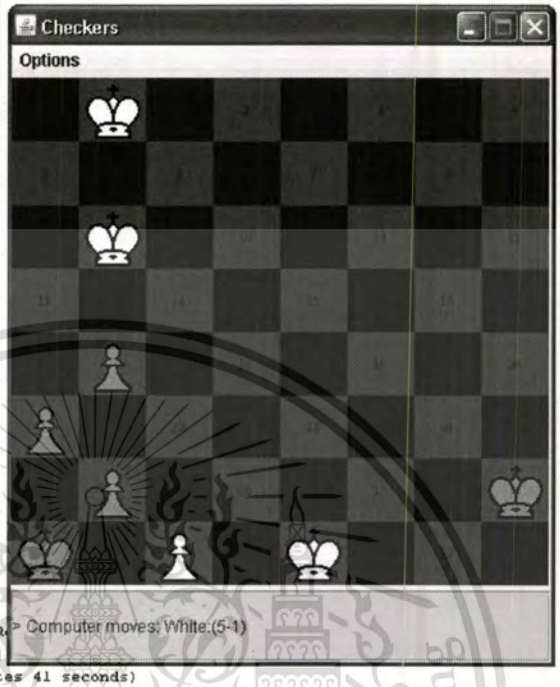




The third experiment

Our checker move first.

1st Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = TIE
(-60 + 55 = -5)




```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)| 0 |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   | x |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)| x |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | x |   |   |   |   |   | X |
    +---+---+---+---+---+---+---+---+
(8)| X |   | o |   | 0 |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination R
E.g. Source[6][1] to Destination[5][2]
 : BUILD STOPPED (total time: 25 minutes 41 seconds)
```

2rd Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON

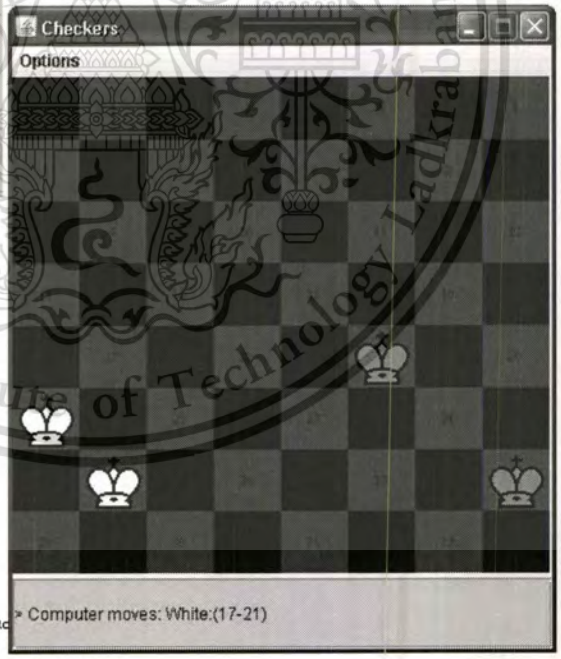

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   | X |   | x |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   | X |   | x |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   | X |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
--------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 25 minutes 30 seconds)
```

3rd Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON

```
    +---+---+---+---+---+---+---+---+
 (1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   |   |   |   |   |   |   | X |
    +---+---+---+---+---+---+---+---+
 (4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (6)| x |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (7)|   | o |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 22 minutes 22 seconds)
```

4th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = TIE
(-30 + 30 = 0)

```
    +---+---+---+---+---+---+---+---+
 (1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (5)|   | 0 |   |   |   | X |   |   |
    +---+---+---+---+---+---+---+---+
 (6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (7)|   | 0 |   |   |   |   |   | X |
    +---+---+---+---+---+---+---+---+
 (8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Ro
E.g. Source[6][1] to Destination[5][2]
 : BUILD STOPPED (total time: 22 minutes 6 seconds)
```



5th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = LOST

```
    +---+---+---+---+---+---+---+---+
(1)|   | O |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   | X |   | x |   | x |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   | o |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination R
E.g. Source[6][1] to Destination[5][2]
 : 1 2 3 4 1 6 3 8
OK
-------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 11 minutes 15 seconds)
```

6th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON

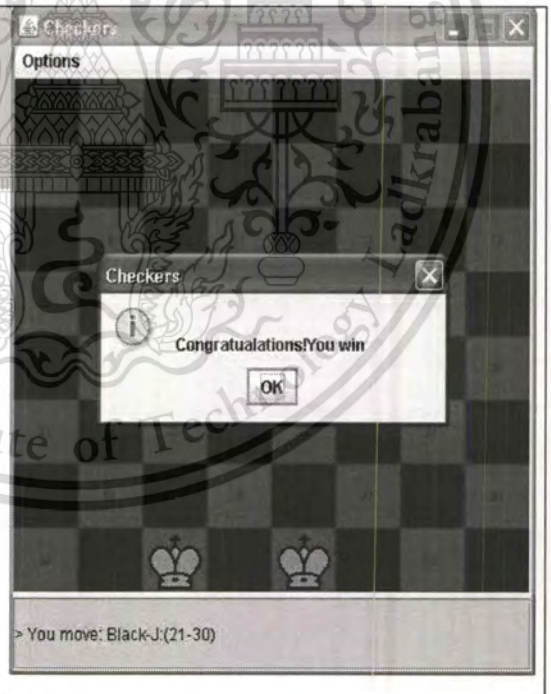
```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)| x |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   | x |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   | X |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | O |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   |   |   |   |   | X |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 13 minutes 56 seconds)
```

7th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON

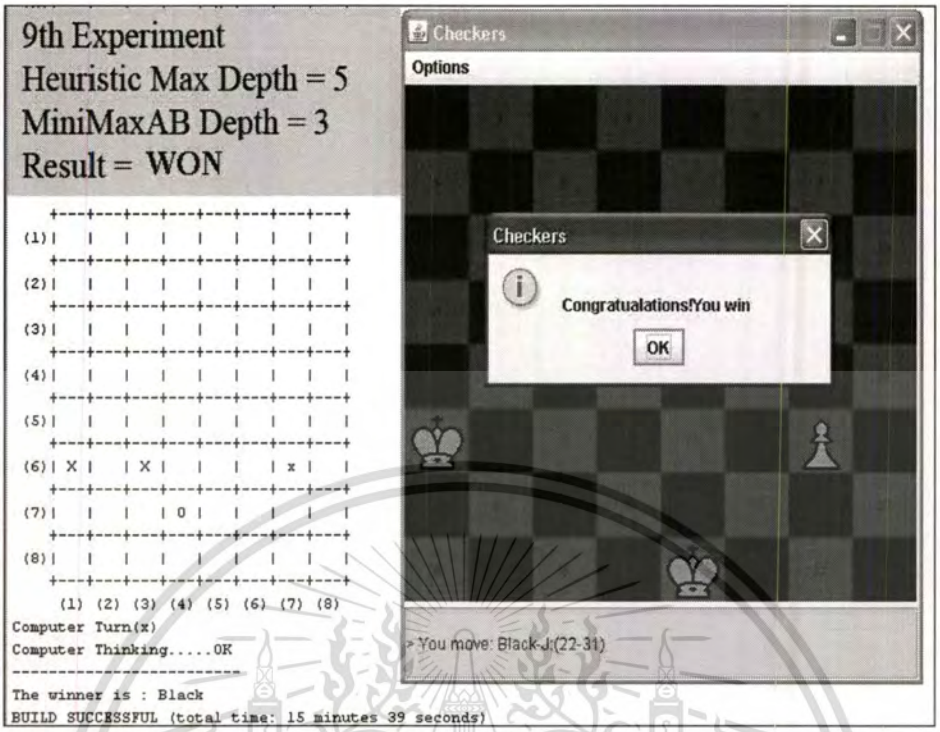
```
    +---+---+---+---+---+---+---+---+
 (1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   |   |   |   |   |   |   | x |
    +---+---+---+---+---+---+---+---+
 (4)|   |   | x |   | X |   | x |   |
    +---+---+---+---+---+---+---+---+
 (5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (7)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (8)|   |   | X |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
      (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 12 minutes 48 seconds)
```

8th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON

```
    +---+---+---+---+---+---+---+---+
 (1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (6)| x |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (7)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (8)|   |   |   |   | X |   |   |   |
    +---+---+---+---+---+---+---+---+
      (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 29 minutes 43 seconds)
```



9th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON




```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)| X |   | X |   |   |   | x |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   | O |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 15 minutes 39 seconds)
```

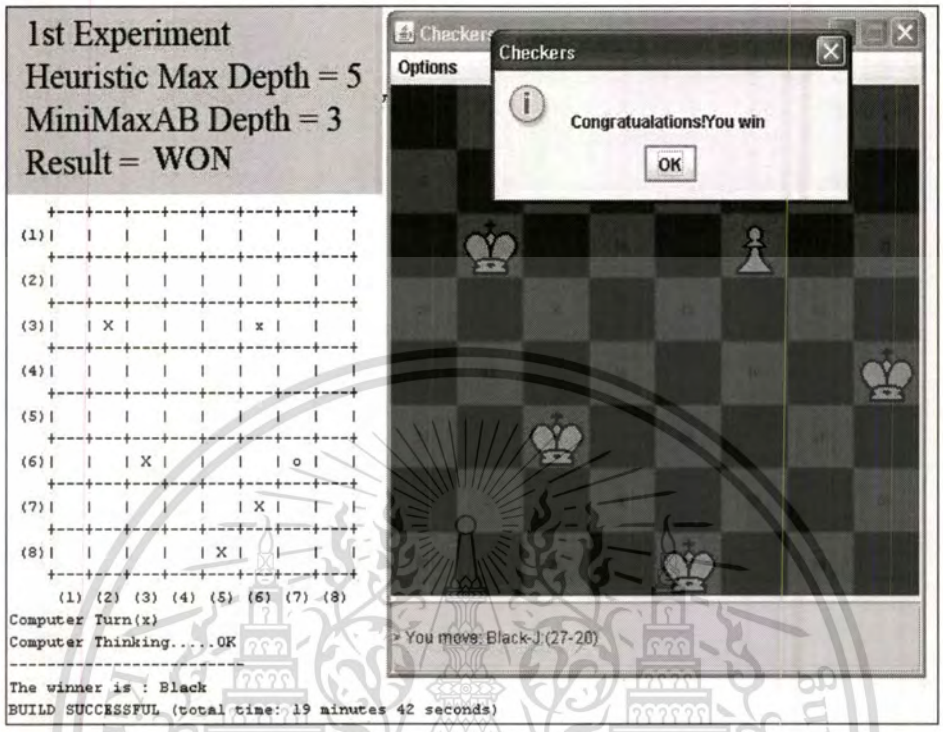
10th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON


```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   | X |
    +---+---+---+---+---+---+---+---+
(6)|   |   | X |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | O |   |   |   | x |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 14 minutes 23 seconds)
```

The third experiment

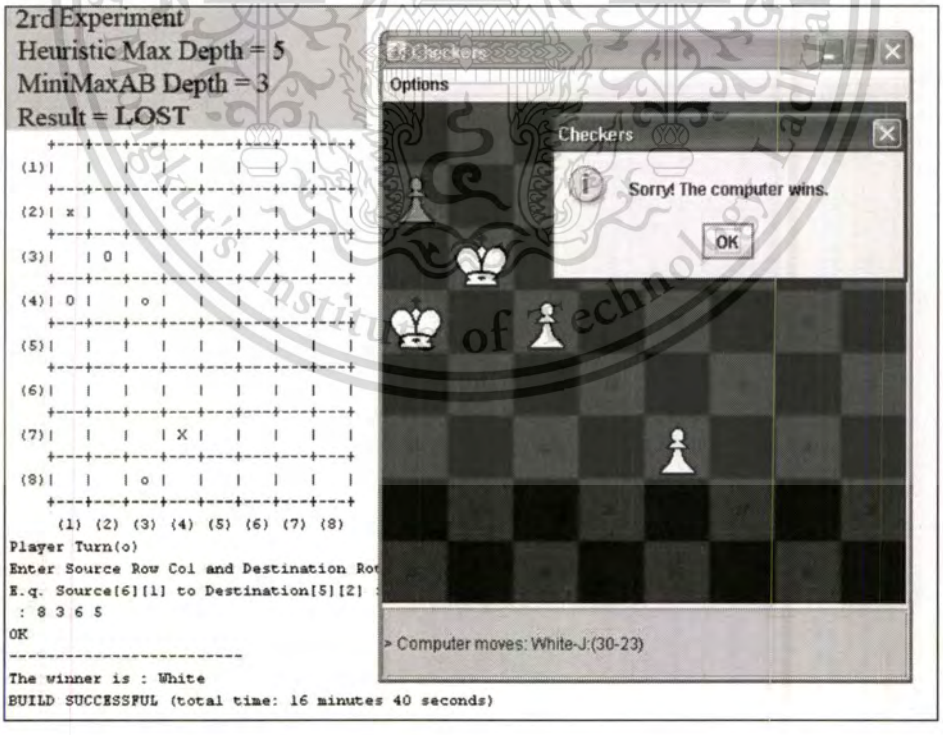
Ander Baumann's checker move first.

1st Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON




```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   | X |   |   |   | x |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   | X |   |   |   | o |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   | X |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   |   |   | X |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 19 minutes 42 seconds)
```

2rd Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = LOST



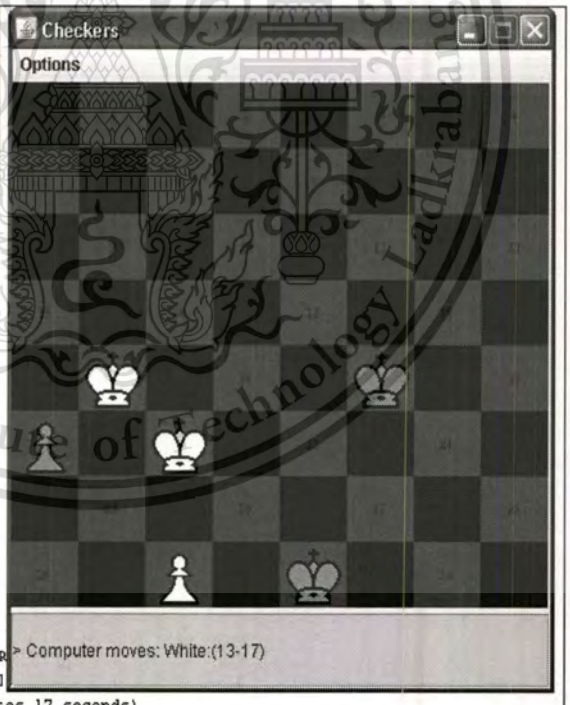

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)| x |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)| 0 |   | o |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   | X |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   | o |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Ro
E.g. Source[6][1] to Destination[5][2]
 : 8 3 6 5
OK
-------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 16 minutes 40 seconds)
```

3rd Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   | x |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   | X |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 17 minutes 5 seconds)
```

4th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = TIE
(-40 + 40 = 0)

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   | 0 |   |   |   | X |   |   |
    +---+---+---+---+---+---+---+---+
(6)| x |   | 0 |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   | o |   | X |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination R
E.g. Source[6][1] to Destination[5][2]
BUILD SUCCESSFUL (total time: 17 minutes 17 seconds)
```



5th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON

```
    +---+---+---+---+---+---+---+---+
 (1)|   | x |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   |   |   |   |   |   |   | x |
    +---+---+---+---+---+---+---+---+
 (4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (5)|   |   |   | x |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (6)| x |   | o |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (7)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (8)| X |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 8 minutes 23 seconds)
```

6th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON

```
    +---+---+---+---+---+---+---+---+
 (1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (4)|   |   |   |   |   |   | X |   |
    +---+---+---+---+---+---+---+---+
 (5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (6)|   |   | X |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
 (7)|   | O |   |   |   |   |   | X |
    +---+---+---+---+---+---+---+---+
 (8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 37 minutes 11 seconds)
```

7th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   | x |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   | x |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   | x |   |   |   | o |
    +---+---+---+---+---+---+---+---+
(4)|   |   | x |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   | x |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   | X |   |   |   | x |
    +---+---+---+---+---+---+---+---+
(8)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 7 minutes 30 seconds)
```

8th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = WON

```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)| X |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   | 0 |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   |   |   | X |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Computer Turn(x)
Computer Thinking.....OK
-------------------------
The winner is : Black
BUILD SUCCESSFUL (total time: 12 minutes 15 seconds)
```

9th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = LOST

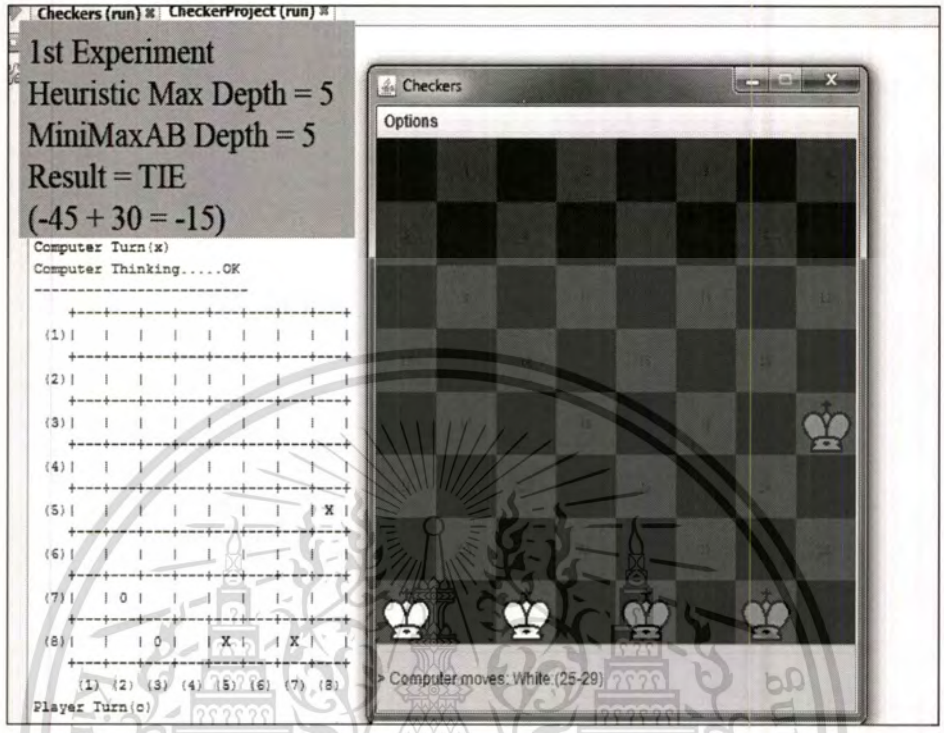
```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   | X |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(8)|   |   | O |   | O |   | o |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Ro
E.g. Source[6][1] to Destination[5][2]
 : 8 3 6 5
OK
-------------------------
The winner is : White
BUILD SUCCESSFUL (total time: 18 minutes 42 seconds)
```

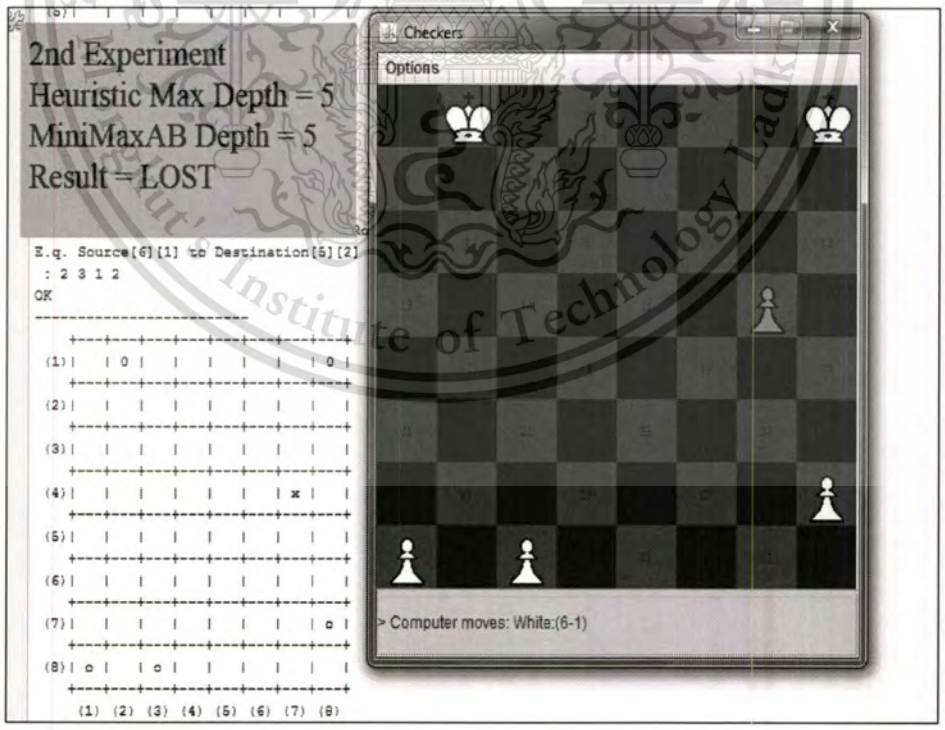
10th Experiment
Heuristic Max Depth = 5
MiniMaxAB Depth = 3
Result = TIE
(-50 + 30 = -20)

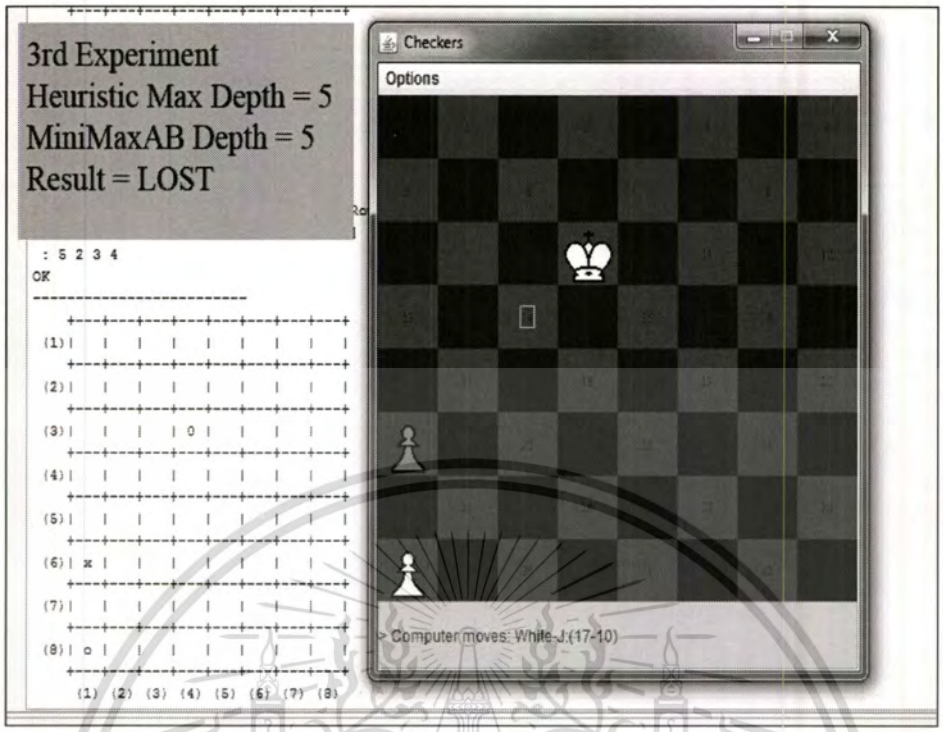
```
    +---+---+---+---+---+---+---+---+
(1)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(2)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(3)|   |   |   |   |   |   |   | x |
    +---+---+---+---+---+---+---+---+
(4)|   |   |   |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
(5)|   |   |   |   |   | x |   |   |
    +---+---+---+---+---+---+---+---+
(6)|   |   |   |   | X |   |   |   |
    +---+---+---+---+---+---+---+---+
(7)|   |   |   |   |   |   |   | X |
    +---+---+---+---+---+---+---+---+
(8)| O |   | O |   |   |   |   |   |
    +---+---+---+---+---+---+---+---+
     (1) (2) (3) (4) (5) (6) (7) (8)
Player Turn(o)
Enter Source Row Col and Destination Row
E.g. Source[6][1] to Destination[5][2] : 6 1 5 2
 : BUILD STOPPED (total time: 30 minutes 6 seconds)
```

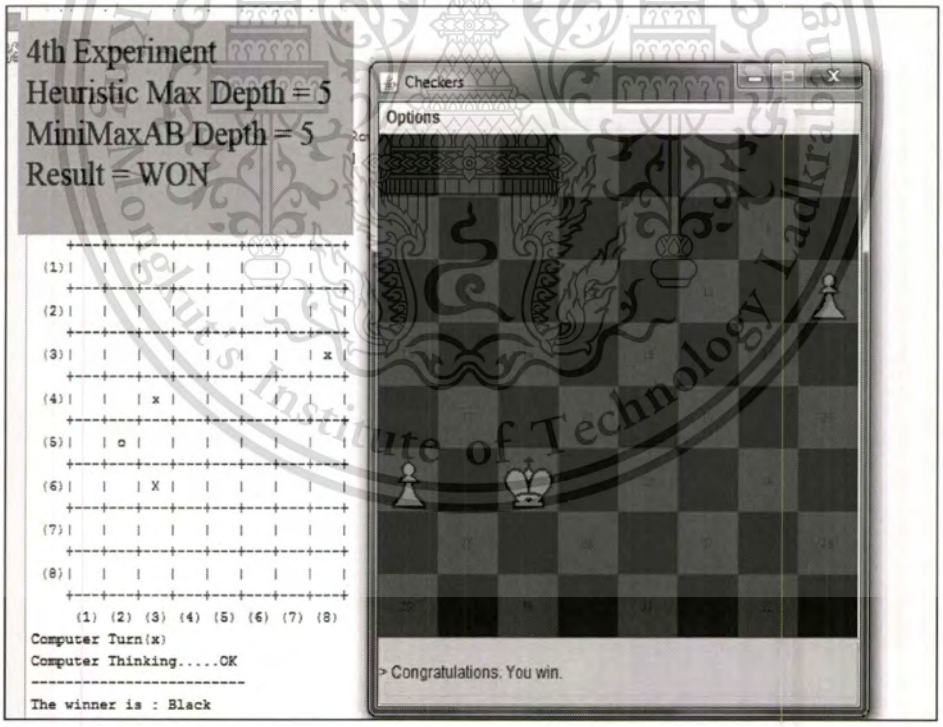
The fourth experiment

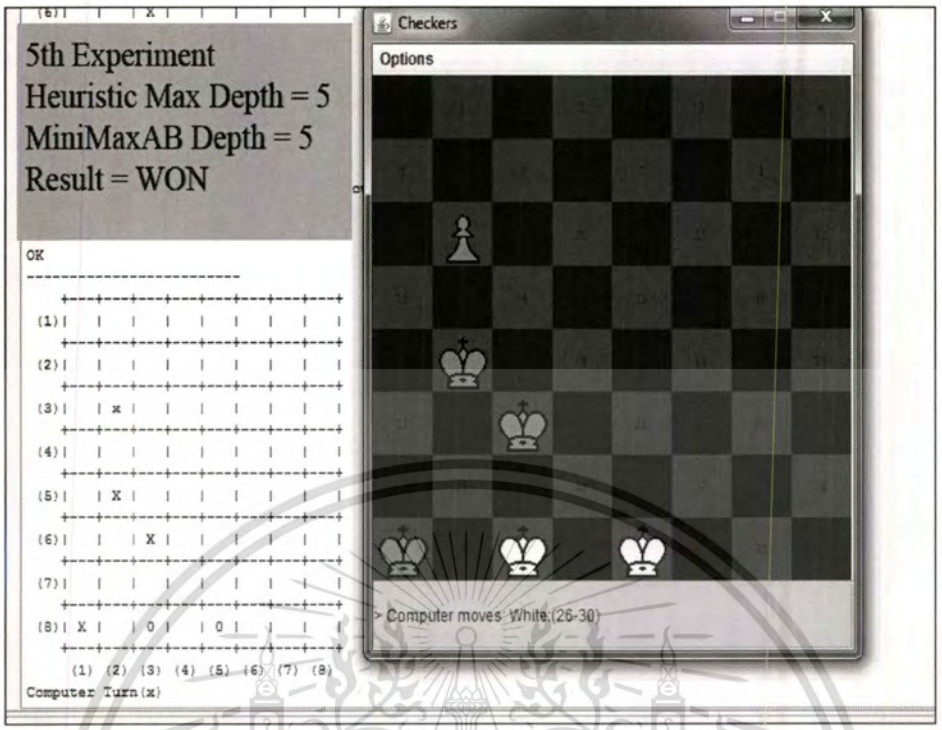
Our checker move first.

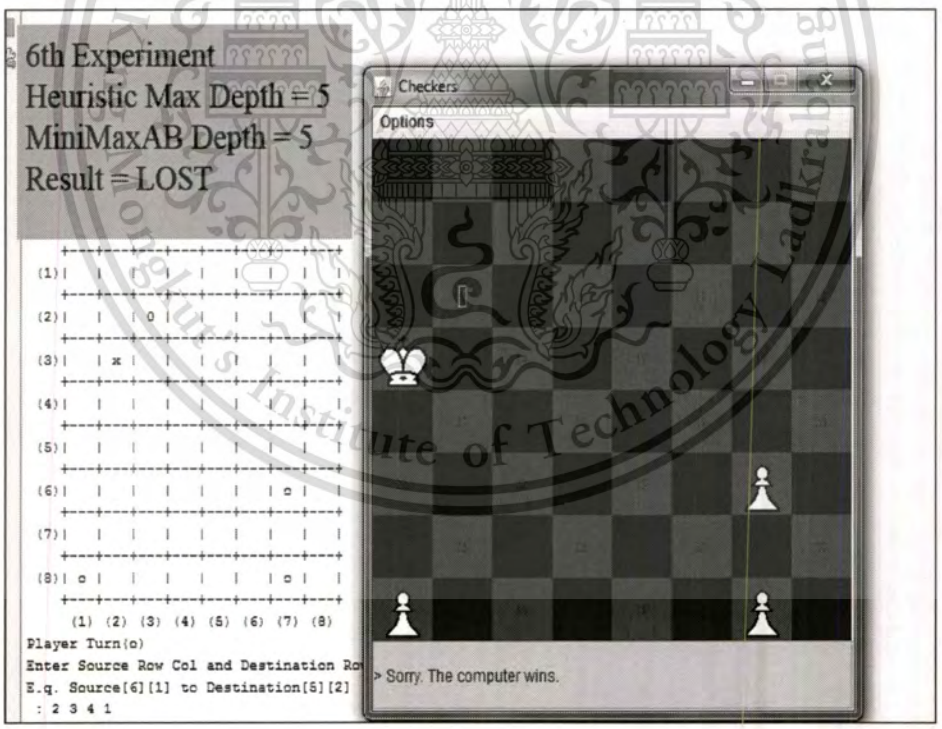


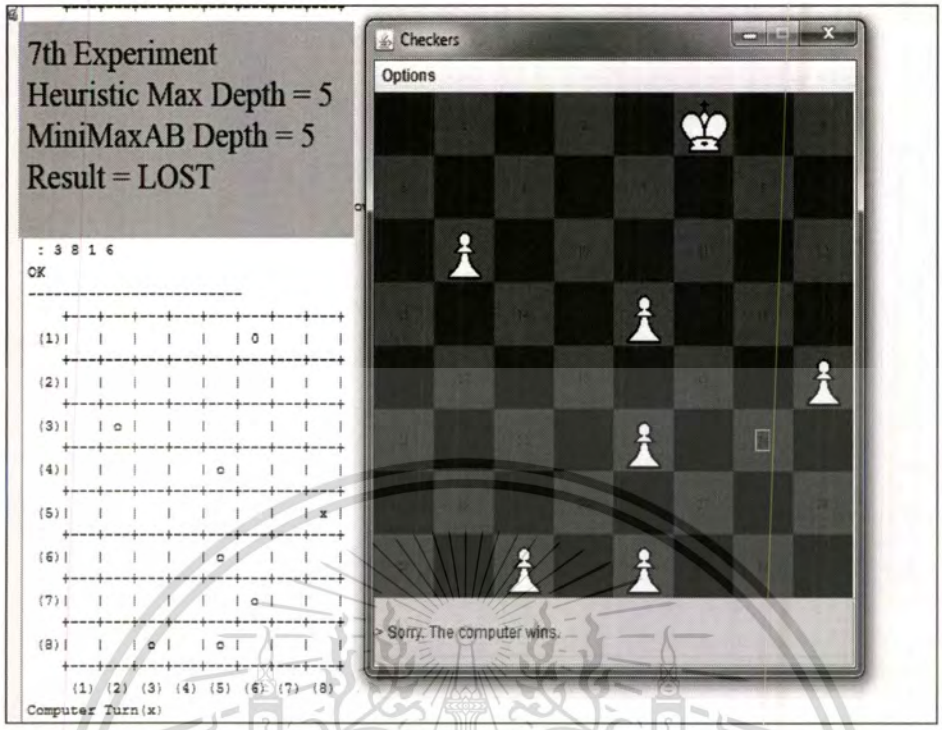


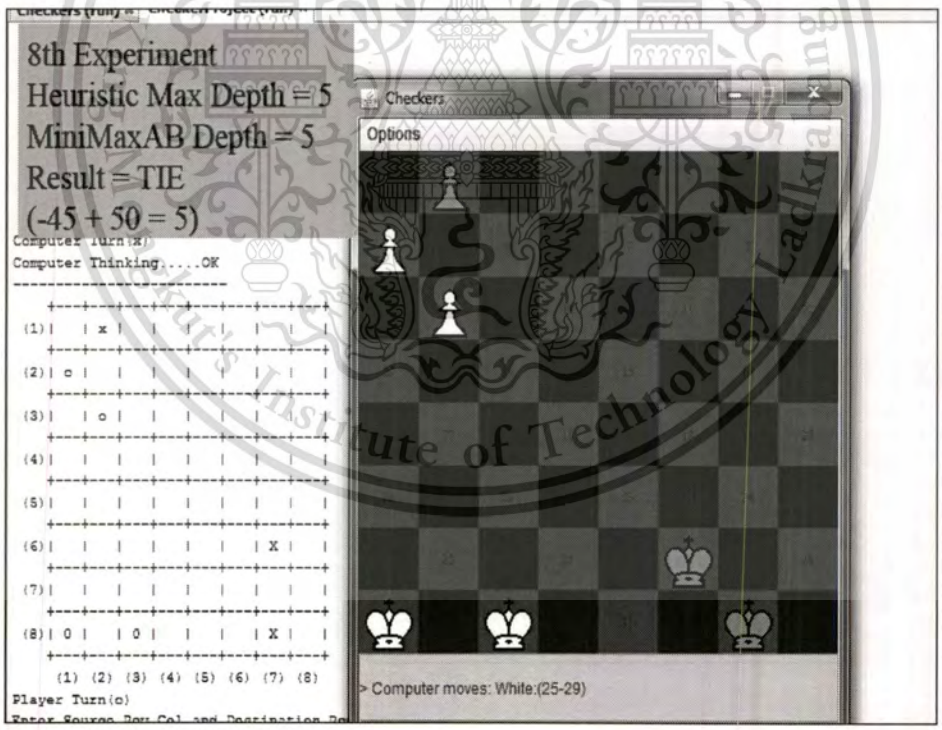


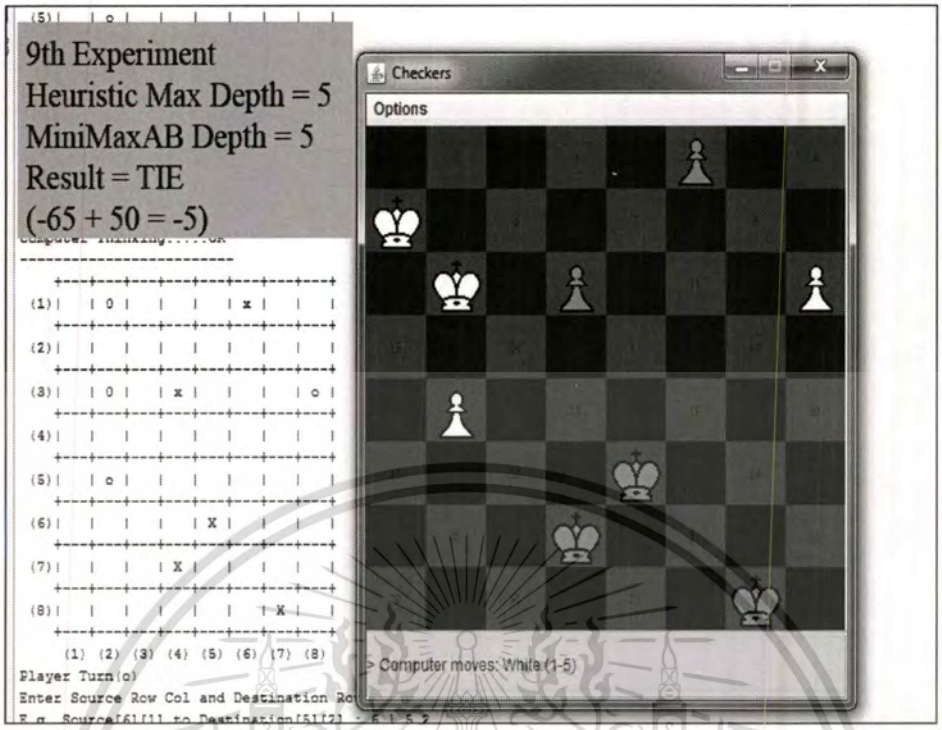


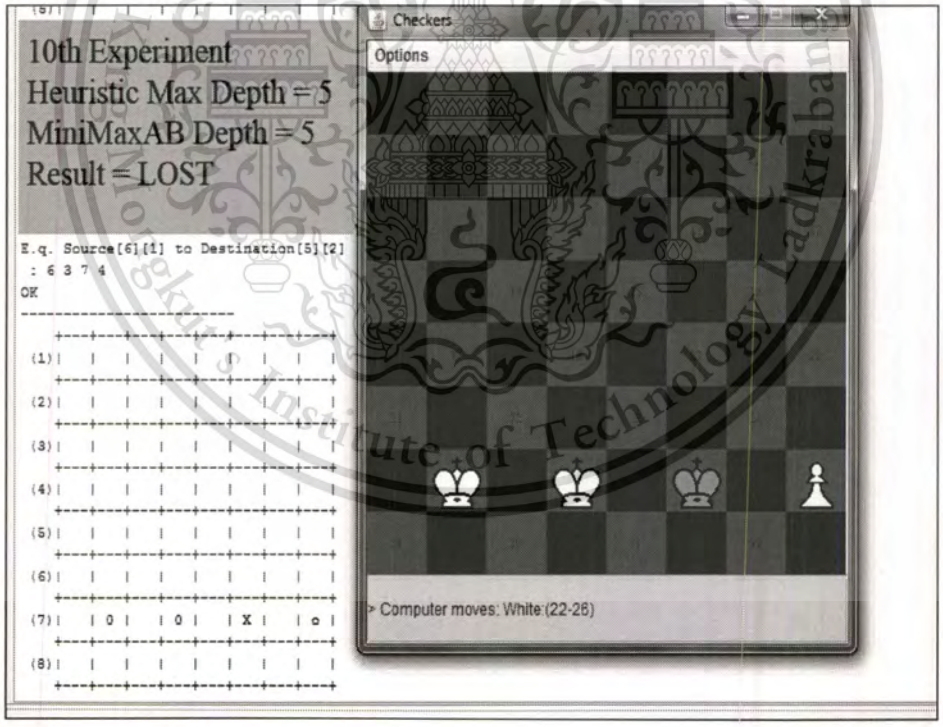












The fourth experiment

Ander Baumann's checker move first.