CSE481: Artificial Intelligence

Intelligent Mancala Game

Submitted by:

Andrew Raafat Farid	1600331
Passant Mohamed ElBaroudy	1600398
Pavly Mario Lofty	1600385
Salah El-Din Adel	1600705
Youssef Raafat Aziz Labib	1601726

Submitted to:

Dr. Manal Mourad

Eng. Ahmed Fathy

Table of contents:

1- Game Description:

Mancala is a two player game that is played with a board and several stones.

Setup:

Players sit on either side of the board. Each player has six bowls and one mancala. Before play begins, four stones are placed into each bowl, while both mancalas are left empty. The object of the game is for a player to collect the most stones in his/her mancala.

Rules:

Play begins by randomly choosing a player to move first. The first player picks up all of the stones in one of his/her bowls. The player then drops one stone into the bowl to the left or the right of the current bowl and continues to move to each bowl sequentially, dropping a stone into each, until no more stones remain in hand. Once a direction counter-clockwise, the player must continue moving in that direction for the duration of the turn. If a player passes by his/her own mancala, a stone is dropped into it. The opponent's mancala, however, is simply ignored. (A player never drops a stone into the opponent's mancala.) If stones still remain in the player's hand after reaching a mancala, players continue moving around to the opposite side of the board (in a circular fashion).

If the last stone a player drops during a turn lands in his/her own mancala, then the player immediately takes another turn.

If a player drops the last stone into one of his/her empty bowls (on the correct side of the board), then that last stone, plus all the stones in the opponent's corresponding bowl, are collected and placed in the player's mancala “Stealing Mode”

2-MiniMax Function Explanation:

Minimax is a recursive algorithm which is used to choose an optimal move for a player assuming that the other player is also playing optimally.

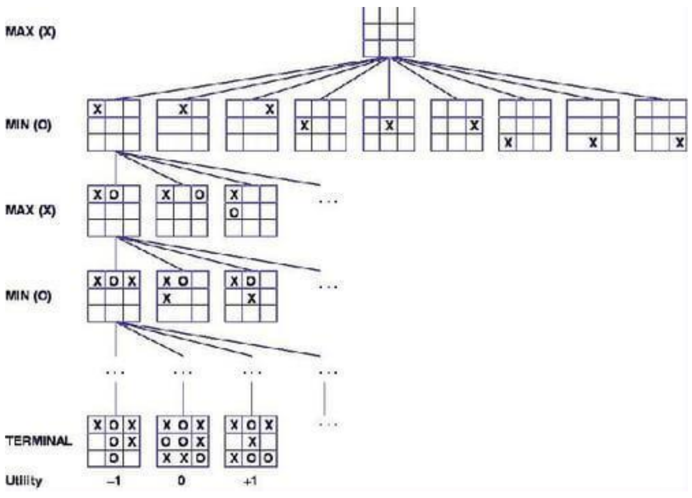
It is similar to how we think when we play a game: “if I make this move, then my opponent can only make only these moves,” and so on.

Minimax is called so because it helps in minimizing the loss when the other player chooses the strategy having the maximum loss

There are two players involved in a game, called MIN and MAX. The player MAX tries to get the highest possible score and MIN tries to get the lowest possible score,

The general process of the Minimax algorithm is as follows:

Step 1: generate the entire game tree starting with the current position of the game all the way up to the terminal states.



The initial state is the first layer that defines that the board is blank it’s MAX’s turn to play.

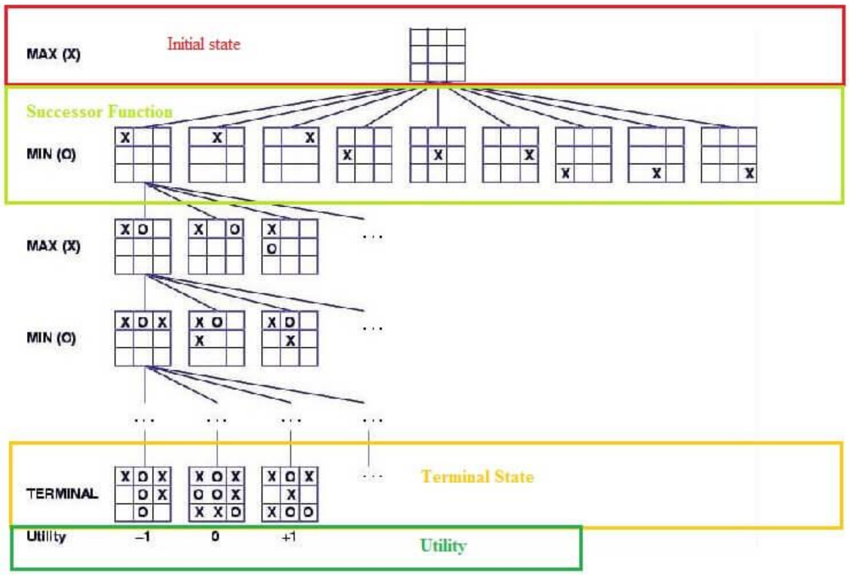
Successor function lists all the possible successor moves. It is defined for all the layers in the tree.

Terminal State is the last layer of the tree that shows the final state, i.e whether the player MAX wins, loses, or ties with the opponent.

Utilities in this case for the terminal states are 1, 0, and -1 as discussed earlier, and they can be used to determine the utilities of the other nodes as well.

Step 2: Apply the utility function to get the utility values for all the terminal states.
Step 3: Determine the utilities of the higher nodes with the help of the utilities of the terminal nodes. For instance, in the diagram below, we have the utilities for the terminal states written in the squares.

Step 4: Calculate the utility values with the help of leaves considering one layer at a time until the root of the tree.
Step 5: Eventually, all the backed-up values reach to the root of the tree, i.e., the topmost point. At that point, MAX has to choose the highest value.



3-Implementation:

We implemented the project with python language through Spyder IDE

We used:

For loops

While loops

Lists

If conditions

Break and Continue

Input function to take the input from the user

Implemented Functions:

1-Minimax Function
2-makeMove Function
3-Board Function
4-PlayerOne Function
5-Playertwo Function
6- Winner Function
7- evalHeuristics Function
8-Iswinner Function
9-mancalaMain “Main Function”
10-Bonus Features:
- store and load function
-easy-medium-hard

MiniMax Function:

miniMax(stonesAmount, depth, alpha, beta, stealing, isMaximizing=True, playAgain=False)

stonesAmout = [4,4,4,4,4,4,0,4,4,4,4,4,4,0] number of stones in each hole.

depth= this an optional number as we increase the depth, the performance of the AI player increases.

alpha=negative infinity.

beta=positive infinity.

Stealing= choose whether we want to be in stealing mode or not.

isMaxmizing= True or False, decided by the user. Choose whether I want to maximize or minimize the current player’s score.

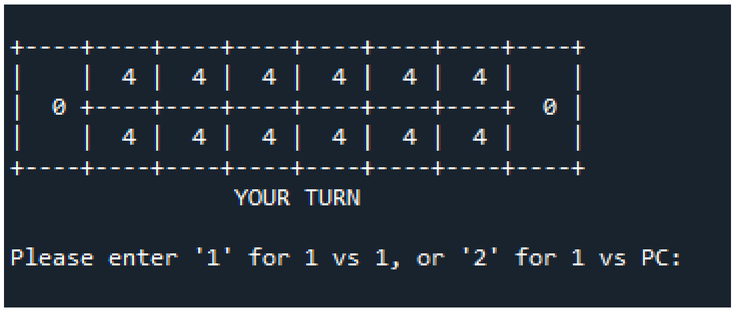
PlayAgain=tells if the AI player will play again or not, and this depends on the rules explained above.

Explanation:

First we check whether we reached depth = 0, or the game has ended; in both cases we evaluate the heuristics and return them.
If the miniMax call is to maximize, we set the maxEval variable to -ve infinity,
and we loop 6 times (Number of moves from A to F), and we call the makeMove with that specific move and we return the new state, if that move is valid, and the player didn't get another turn,
we call the miniMax function again with the new state, but this time minimizing as it's the opponent turn
minEval is initialized with +ve infinity, then we do another loop for the opponents' 6 moves,
call the makeMove function and get the state after making each move, if it's a valid move we send its new state to the miniMax, and we keep going till we reach the depth = 0, that's where the evalHeurisitcs function is called and returns the heuristic, then this value is compared to maxEval or minEval.
After examining all the nodes with all the children, the best move is returned with its score.

Board Function:

Board(stonesAmount,first)
StonesAmount: initial values of stones in the holes. If first is true, the lower side of the game is printed, if it’s false, then the upper side (Player two’s side) is printed. It returns the display of the game.



```
+----+----+----+----+----+----+----+----+
|    | 4  | 4  | 4  | 4  | 4  | 4  |    |
|  0 +----+----+----+----+----+----+  0 |
|    | 4  | 4  | 4  | 4  | 4  | 4  |    |
+----+----+----+----+----+----+----+----+
                YOUR TURN

Please enter '1' for 1 vs 1, or '2' for 1 vs PC:
```

If player entered 1 he will play without using AI , if he entered 2 he will play VS Ai using function Miniax

Player 1 And Player 2 Function:

playerOne(u_in)

playerTwo(u_in)

It’s a function for each player it takes the letter the user entered and transform it to the right bin number to be used in the other function for example if player 1 is playing and he entered ‘a’ so it will return it as bin number 0, It also takes Q or q to quit from the game.

Winner Function:

winner(board, miniMax)

winner(board, miniMax) It takes the board and checks if there's a player that has 6 empty holes the function will return all the stones for the other player on his mancala then will check which player has a higher number of stones then this will be the winner, it also takes a Boolean var 'miniMax', to differentiate whether it was called from the main or minimax, if it's called from minimax we don't print the board if the game is over as in minimax we're just calculating all the possible cases, not actually playing the game.

MakeMove Function:

makeMove(stones, bin_index, isPlayer1,stealing)

this function is the one responsible for the game rules it takes the number of stones, bin index (which hole), is player1 to know which player will play one or two, and stealing to know if there's stealing or no. The first thing we check if the chosen bin is valid or not (if it's an empty hole we tell the player to choose another valid one), then we distribute the stones in the holes and the player's mancala then check if the last stone was in the player's mancala he plays again and check if stealing mode is one (if the player put the last stone in an empty hole he takes all the other stones in the corresponding hole for the opponent). The same rules for player 2.

evalHeuristics:

evalHeuristics(stonesEval,playAgain,isPlayerOne)

this function returns the heuristic is Minimax function when the depth = zero , it returns the difference between first player's mancala and second player's mancala

mancalaMain:

Here's the main function where we call all the other functions and integrate the whole code, we

ask the user all the questions like which level , which mode stealing or not stealing we set the values true or false , add call each function to do it's role in the game.

StoreAndLoad:

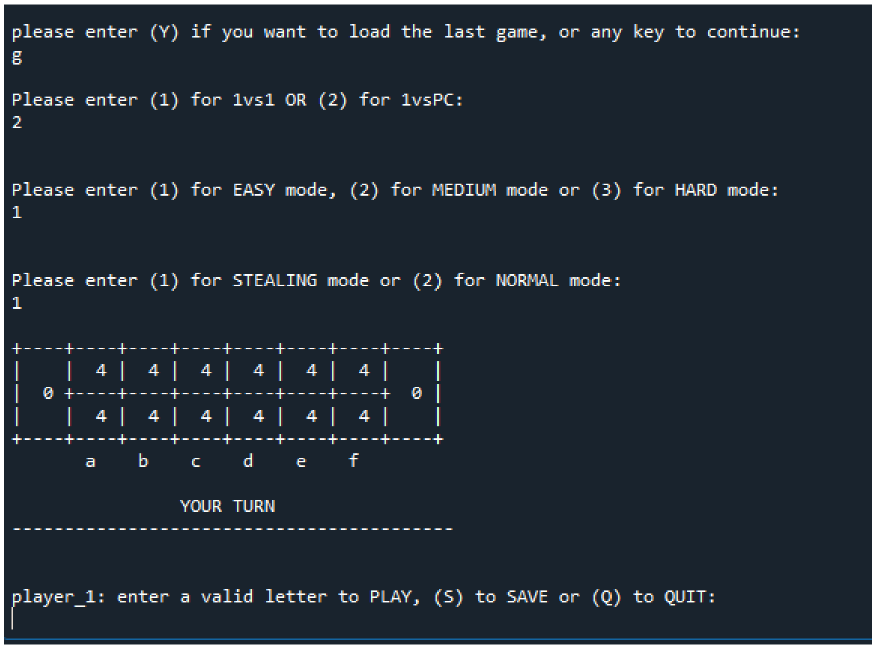
saveLoad Function takes arguments (stonesAmount,gameMode,Stealing,gameDepth,Save,Load)

If we're saving the current game, we open a textfile and write the status of Game Mode, Stealing, Game Depth and then we write the stones amount of each player, each value is written in a new line. If we're loading the game, we open the text file and we loop on each line in it and retrieve the Game Mode, Game Depth, Stealing, and the number of stones in each hole and then we pass them to our main function and the game is loaded.

A user guide with snapshots:

-Start:

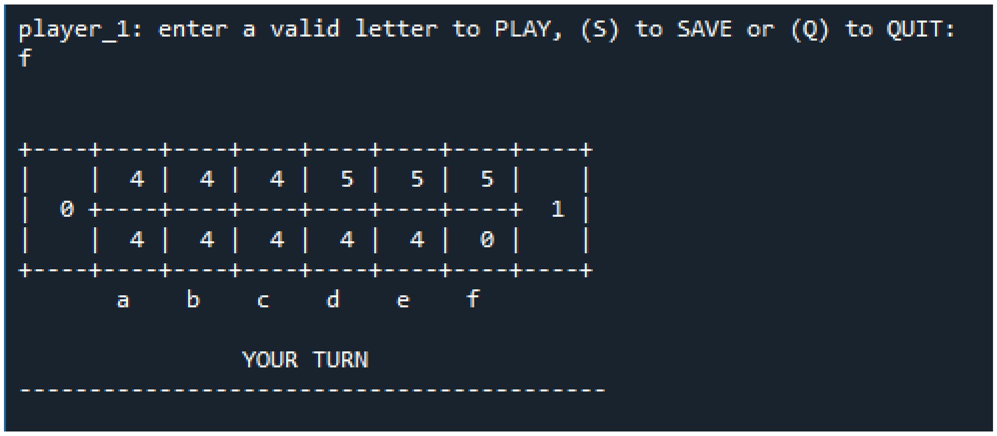
-First we ask some questions to start the game.



```
please enter (Y) if you want to load the last game, or any key to continue:
g

Please enter (1) for 1vs1 OR (2) for 1vsPC:
2


Please enter (1) for EASY mode, (2) for MEDIUM mode or (3) for HARD mode:
1


Please enter (1) for STEALING mode or (2) for NORMAL mode:
1

+----+----+----+----+----+----+----+----+
|    |  4 |  4 |  4 |  4 |  4 |  4 |    |
|  0 +----+----+----+----+----+----+  0 |
|    |  4 |  4 |  4 |  4 |  4 |  4 |    |
+----+----+----+----+----+----+----+----+
       a    b    c    d    e    f

                YOUR TURN
------------------------------------------


player_1: enter a valid letter to PLAY, (S) to SAVE or (Q) to QUIT:
```

-Player’s One Turn:

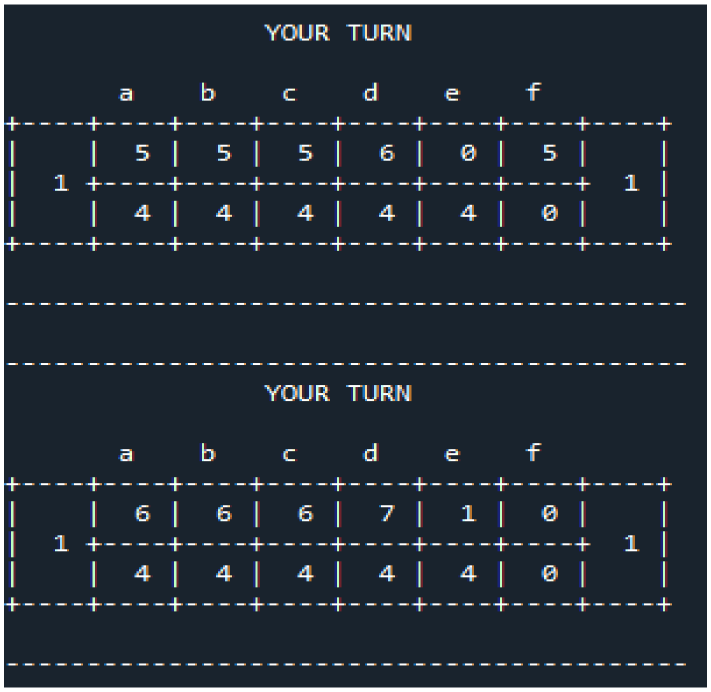
Player One Played ‘F’.



```
player_1: enter a valid letter to PLAY, (S) to SAVE or (Q) to QUIT:
f


+----+----+----+----+----+----+----+----+
|    |  4 |  4 |  4 |  5 |  5 |  5 |    |
|  0 +----+----+----+----+----+----+  1 |
|    |  4 |  4 |  4 |  4 |  4 |  0 |    |
+----+----+----+----+----+----+----+----+
       a    b    c    d    e    f

                YOUR TURN
------------------------------------------
```

-Player’s Two Turn:

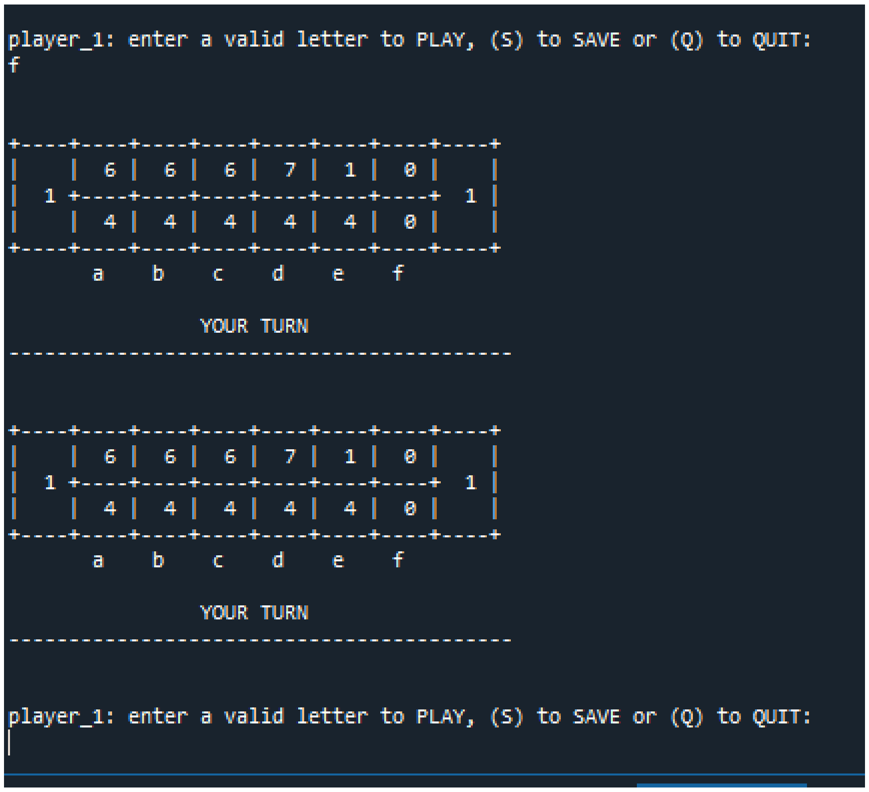
Player 2 played twice, as it played ‘e’ first so the last stone was dropped in his mancala, then it played ‘f’.



```
                YOUR TURN

       a    b    c    d    e    f
+----+----+----+----+----+----+----+----+
|    |  5 |  5 |  5 |  6 |  0 |  5 |    |
|  1 +----+----+----+----+----+----+  1 |
|    |  4 |  4 |  4 |  4 |  4 |  0 |    |
+----+----+----+----+----+----+----+----+

------------------------------------------

------------------------------------------
                YOUR TURN

       a    b    c    d    e    f
+----+----+----+----+----+----+----+----+
|    |  6 |  6 |  6 |  7 |  1 |  0 |    |
|  1 +----+----+----+----+----+----+  1 |
|    |  4 |  4 |  4 |  4 |  4 |  0 |    |
+----+----+----+----+----+----+----+----+

------------------------------------------
```

-Trying to play invalid letters:

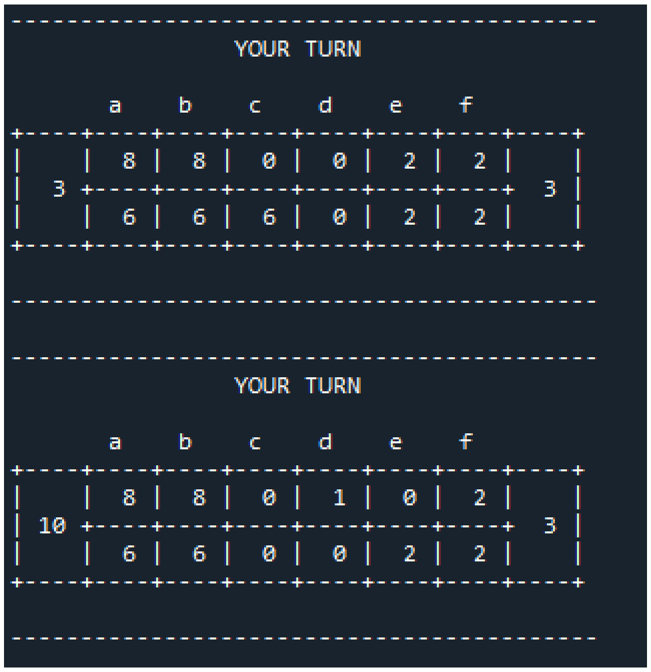
Game won’t continue until a valid move is entered.



```
player_1: enter a valid letter to PLAY, (S) to SAVE or (Q) to QUIT:
f


+----+----+----+----+----+----+----+----+
|    |  6 |  6 |  6 |  7 |  1 |  0 |    |
|  1 +----+----+----+----+----+----+  1 |
|    |  4 |  4 |  4 |  4 |  4 |  0 |    |
+----+----+----+----+----+----+----+----+
       a    b    c    d    e    f

                YOUR TURN
------------------------------------------


+----+----+----+----+----+----+----+----+
|    |  6 |  6 |  6 |  7 |  1 |  0 |    |
|  1 +----+----+----+----+----+----+  1 |
|    |  4 |  4 |  4 |  4 |  4 |  0 |    |
+----+----+----+----+----+----+----+----+
       a    b    c    d    e    f

                YOUR TURN
------------------------------------------


player_1: enter a valid letter to PLAY, (S) to SAVE or (Q) to QUIT:
```

-After some moves, stealing by the PC:

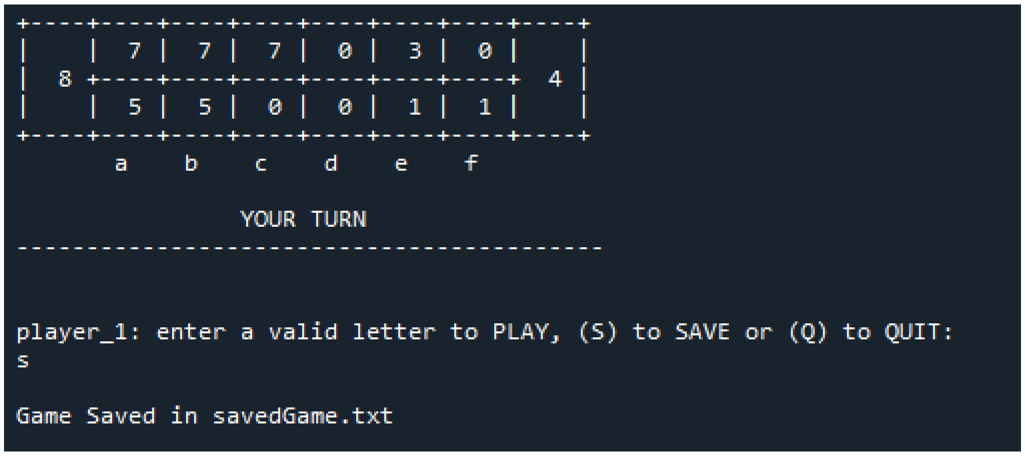
PC made a move ‘E’ so the last stone ended in hole ‘C’ which was empty and had 6 stones in the corresponding holes.



```
----------------------------------------
               YOUR TURN

       a    b    c    d    e    f
+----+----+----+----+----+----+----+----+
|    |  8 |  8 |  0 |  0 |  2 |  2 |    |
|  3 +----+----+----+----+----+----+  3 |
|    |  6 |  6 |  6 |  0 |  2 |  2 |    |
+----+----+----+----+----+----+----+----+

----------------------------------------

----------------------------------------
               YOUR TURN

       a    b    c    d    e    f
+----+----+----+----+----+----+----+----+
|    |  8 |  8 |  0 |  1 |  0 |  2 |    |
| 10 +----+----+----+----+----+----+  3 |
|    |  6 |  6 |  0 |  0 |  2 |  2 |    |
+----+----+----+----+----+----+----+----+

----------------------------------------
```

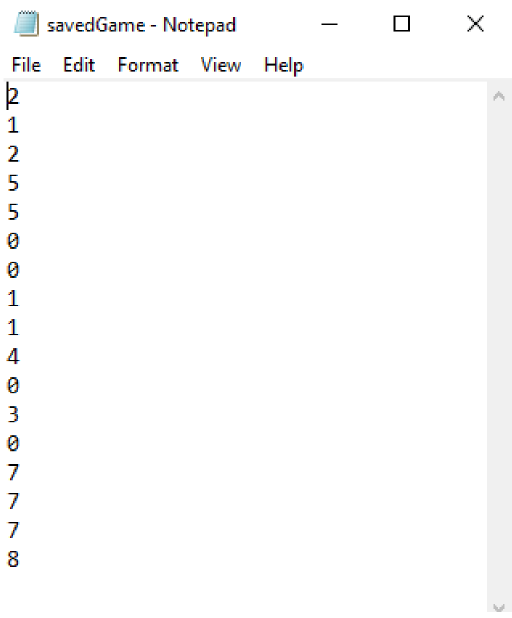
Store & Load Function:

-Saving the game:

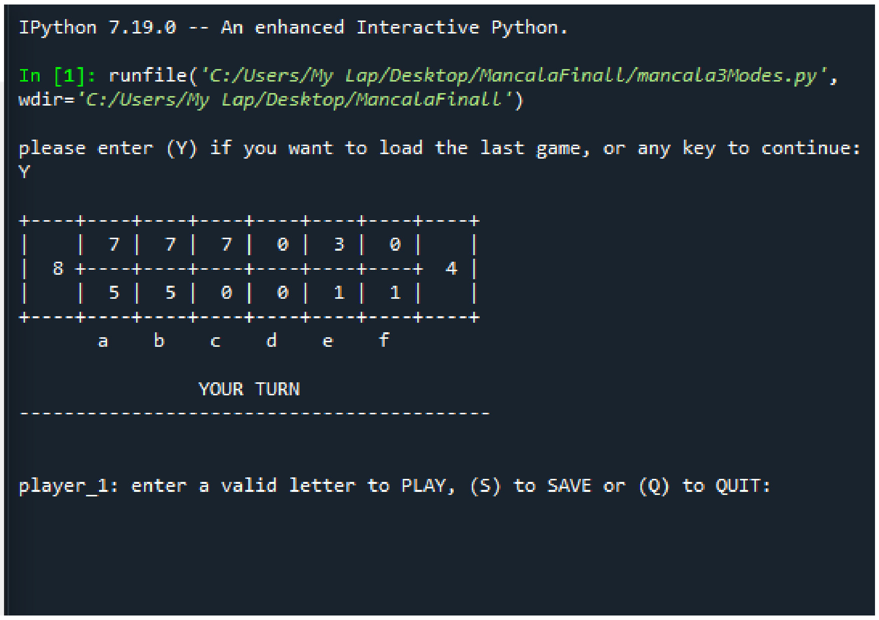


```
+----+----+----+----+----+----+----+----+
|    |  7 |  7 |  7 |  0 |  3 |  0 |    |
|  8 +----+----+----+----+----+----+  4 |
|    |  5 |  5 |  0 |  0 |  1 |  1 |    |
+----+----+----+----+----+----+----+----+
       a    b    c    d    e    f

                YOUR TURN
------------------------------------------


player_1: enter a valid letter to PLAY, (S) to SAVE or (Q) to QUIT:
s

Game Saved in savedGame.txt
```

-The game was stored in savedGame.txt file and then we were able to load it again in the game and continue playing.



```
2
1
2
5
5
0
0
1
1
4
0
3
0
7
7
7
8
```



```
IPython 7.19.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/My Lap/Desktop/MancalaFinall/mancala3Modes.py',
wdir='C:/Users/My Lap/Desktop/MancalaFinall')

please enter (Y) if you want to load the last game, or any key to continue:
Y

+----+----+----+----+----+----+----+----+
|    |  7 |  7 |  7 |  0 |  3 |  0 |    |
|  8 +----+----+----+----+----+----+  4 |
|    |  5 |  5 |  0 |  0 |  1 |  1 |    |
+----+----+----+----+----+----+----+----+
       a    b    c    d    e    f

                YOUR TURN
------------------------------------------


player_1: enter a valid letter to PLAY, (S) to SAVE or (Q) to QUIT:
```

A summary of how the work was split among our team members:

Name	Code	Work
Andrew Rafaat	1600331	-Main Function -MiniMax Function -Store and load Function -Debugging
Pavly mario	1600385	-Board Function -Winner Function -Testing
Passant Mohamed	1600398	-Makemove Function -PlayerOne and PlayerTwo Function -Report -Testing
Salah El-Din Adel	1600705	-Main Function -MiniMax Function -Testing
Youssef Rafaat	1601726	-Makemove Function -evalHeuristics -Debugging

YouTube Link: Mancala Game with Artificial Intelligence. - YouTube

Github Link: https://github.com/PassantElBaroudy/Mancala

References:

https://en.wikipedia.org/wiki/Minimax
https://endlessgames.com/wp-content/uploads/Mancala_Instructions.pdf