Faculty of Engineering - Ain Shams University
Computer and Systems Engineering Department

Mancala Game

Submitted by:

Abdallah Khaled Ahmed Mahmoud Elshamy	1600766
Omar Abdel Wahab Abdel Monem Ali	1600890
Omar Fathy Hamed Mahmoud	1600899
Toka Abdul Hamied Abou-El Gheit	1600437
Abdallah Nasser Abdallah	1600813

Submitted to:

Dr. Manal M. Zaki

Faculty of Engineering

Ain Shams University

Table of Contents

1.0 Description and implementation details

1.1 game description

The board of mancala has 12 small pockets 6 for each side/player, and 2 larger pockets on each of the ends called mancala, Each of the 12 halls is filled with 4 stones. The goal of the game is to capture the most stones in your mancala.

Play: pick all the stones in any one of your pockets on your side of the board, then drop a single stone into the next pocket in an anticlockwise direction, until the stones run out.

If you run on your mancala drop a stone in it but if you run on the opposite mancala skip it.

If the last stone you drop is in your mancala, take another turn.

Steal mode: If the last stone you drop is in an empty pocket on your side, you capture that stone and any stone in the opposite pocket and place it all in your mancala.

Finish: If you cannot play because all six pockets are empty, the game ends and all the stones on the other side of the board are captured in the other player's mancala.

The winner: after counting the stones in each mancala the player with the most pieces wins.

1.2 implementation details

Evaluate function:

This function is the function responsible for drawing the tree and finding the next optimal move for the AI to play

```
evaluate(state_tuple, is_max=1, level=0, alpha=float('-inf'),
beta=float('inf'))
```

This function takes as input the state_tuple in this tuple exists the following values:

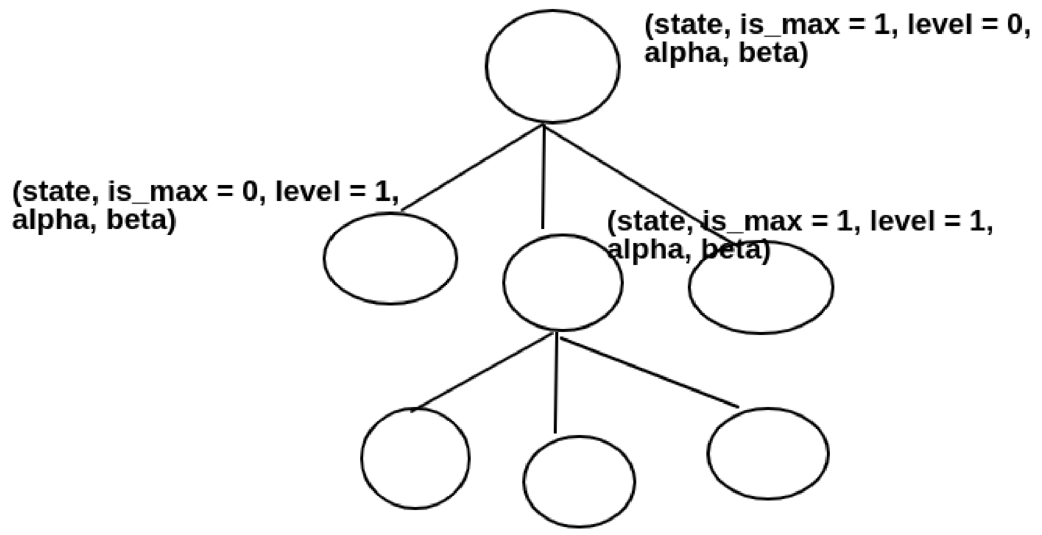
state, is_final, another_turn

The state value is an array representing the mancala board.

Is_final value is a boolean that represents if this move is a final move in the game.

And another_turn represents if there is another turn.

The function depends on these values to operate


(state, is_max = 1, level = 0, alpha, beta)
(state, is_max = 0, level = 1, alpha, beta)
(state, is_max = 1, level = 1, alpha, beta)

In our tree there exists neighbouring nodes that can be different one is a maximizer node and one is minimizer node the value **is_max** is the value representing whether the node is maximizer or minimizer and this value depends on the **another_turn** variable if the state has another turn then in the recursion when calling the evaluate function the is_max is toggled.

```
def evaluate(state_tuple, is_max=1, level=0, alpha=float('-inf'),
beta=float('inf')):

   global max_levels

   global steal

   state, is_final, another_turn = state_tuple

   if is_max:

       possible_states = next_moves(state, "ai")
   else:

       possible_states = next_moves(state, "user")



   optimal_next_state = tuple()

   # if we hit a state with no next moves, return the score of this state

   # if we reached the max depth return the value of the utilty function
(current score)

   if is_final or max_levels == level:

       return state[13] - state[6], state_tuple


   # if the user gets another turn flip is_max, so when it is flipped

   # it has the correct value

   if another_turn:

       next_max = is_max

   else:

       next_max = not is_max


   for next_state in possible_states:

       score, state_tuple = evaluate(

           next_state, next_max, level + 1, alpha, beta)


       if is_max:

           if score > alpha:
               alpha = score

               optimal_next_state = next_state

       else:

           if score < beta:

               beta = score

               optimal_next_state = next_state


       if alpha >= beta:

           break


   score = alpha if is_max else beta

   return score, optimal_next_state
```

We will walk through what the function does:

1- it uses the global **max_level** representing the difficulty and **steal** whether the game is with steal mode activated or not

2- it takes the values discussed above from the state tuple

3- it calls next_moves function to take all the possible next moves if maximizer take moves of ai if minimizer takes moves of user

4- we create an empty tuple for the optimal next_state to play

5- next we check if we reached the max_level or the final move and return the score along with the state tuple

6- next we check if there is another turn if there is we call evaluate function with the same is_max value if no we call it with the toggled value

7- we loop over the next possible moves and call evaluate and we get the score and the optimal move from the function call

8- then we check if maximizer and the score bigger than alpha update alpha by the new score and do the inverse if minimizer and update beta with the new score

9- if alpha > beta we must break to not do more traversing and optimization.

10- we update the score to alpha if maximizer else we update it to beta

11- we return the score and the next optimal_move

Make a move:

It is simply a function that allows its user to empty a certain pit and create a new state of the game, this function allows stealing and no stealing.

1. The function checks if the entered pit number has stones or no and if it doesn't have stone it returns nothing.
2. If the user chose a pit that has stones, the function will keep increasing every next pit by one and decreasing the chosen pit by one.
   - 2.1. While the stone we are dropping is not the last stone in the chosen pit:
     - 2.1.1. If the next pit is the user's mancala, it will increase by one.
     - 2.1.2. If the next pit is the opponent's mancala, it is skipped and the first pit of the user is increased.
     - 2.1.3. If the next pit is not a mancala of either player, it will increase by one.
   - 2.2. If the stone we are dropping is the last stone in the chosen pit:
     - 2.2.1. If the next pit is the user's mancala, the mancala is increased by one and the user is granted another turn.
     - 2.2.2. If the next pit is the opponent's mancala, it is skipped and the first pit of the user will be increased, if stealing is activated and the first pit of the user has 0 stones, stealing takes place and the stone will be summed with the stones in the opposite side and added to the user's mancala.

2.2.3. If the next pit has 0 stones, in case of no stealing the next pit will increase by one, in case of stealing the next pit will not increase by one but dropped stone and the stones on the opposite side will be added to the user's mancala.

2.2.4. If the next pit is not a mancala or a pit with 0 stones, it is increased by one.

3. The function returns the new state, is_final which tells if the game is finished or not, and another_turn which tells if the user has another turn.

The function takes the player, to determine if it is a "user" or an "AI" because our game uses an array of 14 items with pits from 0-6 for the user and pits from 7-13 for the AI with the mancala the user at pit 6 and mancala of the AI at pit 13.

Check for final:

This function simply checks if it's the final game or not, and if it's all the stones on the other side of the board are captured in the other player's mancala. And that by checking:

1. If the state bins [0:6] are all zeros then the sum of all stones from bins [7:13] are assigned to bin 13
2. But If the state bins [7:13] are all zeros then the sum of all stones from bins [0:6] are assigned to bin 6

The function takes the current state of mancala and returns a boolean that indicates if it's the final game this boolean became true if any of the two previous cases happened and also returns the state as if it's the final the state will be modified.

Next moves:

This function is the bridge between make a move function and evaluate function as it makes all possible moves that AI can take changing the bin number and returns a tuple of all possible states in addition to the values of `is_final and another_turn` for each move. It takes the current state and the player which expected to be AI and return the `Next_moves` tuple that has 6 new possible states. And 6 possible values for `is_final and another_turn.`

2.0 Utility functions description

For this game, we used a simple utility function which is:

$$Number\ of\ stones\ in\ AI's\ Mancala\ -\ Number\ of\ stones\ in\ user's\ Mancala$$

This value will be at the leaf nodes when we hit the maximum depth. If the value of this function is positive, we assume that this is a good move for the AI. If it is negative, we assume that it is a good move for the user.

3.0 User guide

1. The game starts with asking the user if he wants to play a new game or load a previous one.
2. If the user chose a new game, the program asks him to select a difficulty (easy - medium- hard), otherwise, the difficulty is loaded from the savefile.
   2.1. The program asks the user if he wants steal mode or no steal mode.

    2.2. The user is then asked if he wants to be asked to save the game every turn or no.

    2.3. If the user chose to save the game, he will be asked every turn if he wants to save the game or not.

    2.4. If the user chose to not save the game, the program will not ask him again.

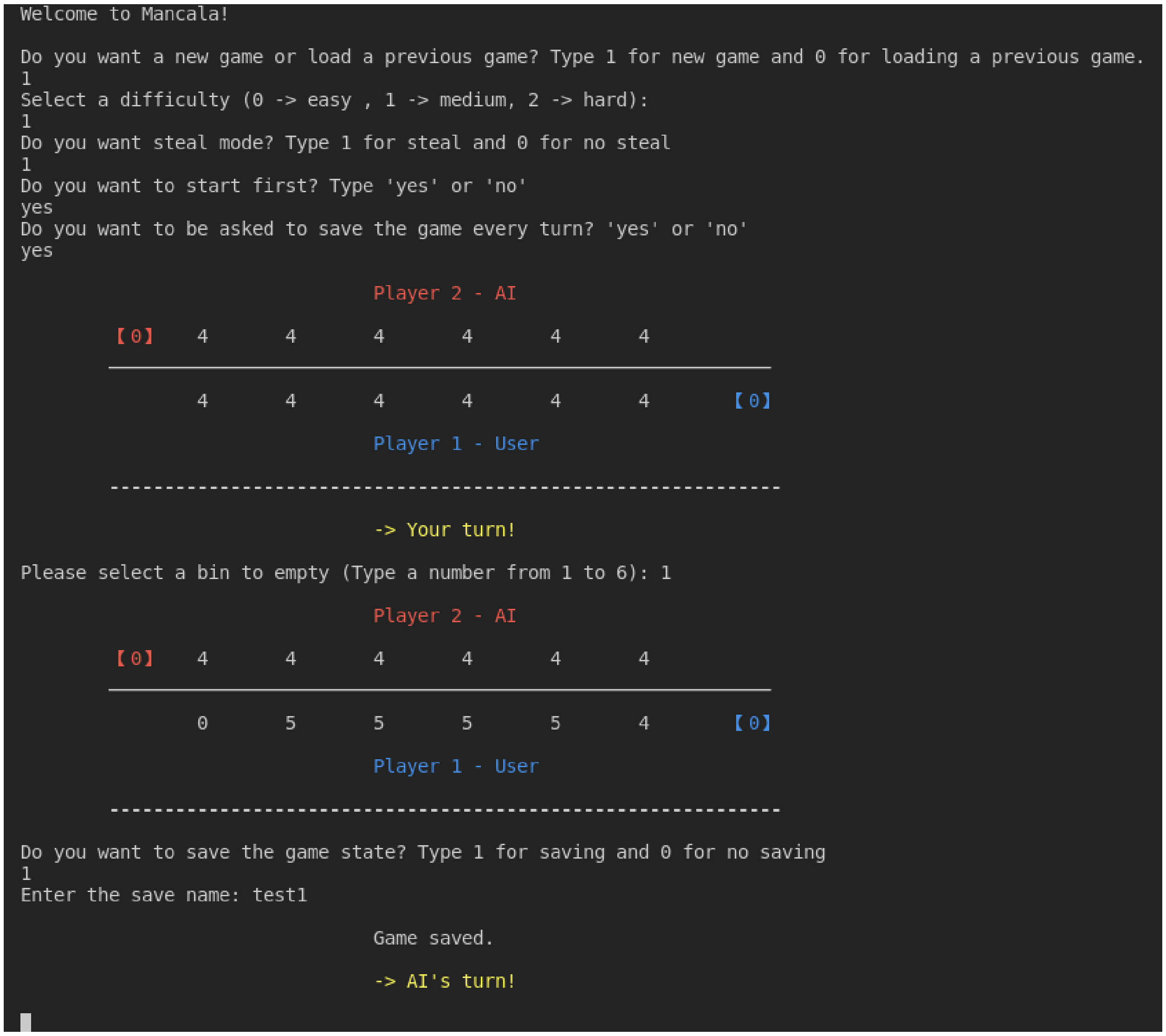
3. If the user chose to load the game, he will be asked to provide the save filename.

    3.1. The user will be asked if he wants to save the game every turn or no.

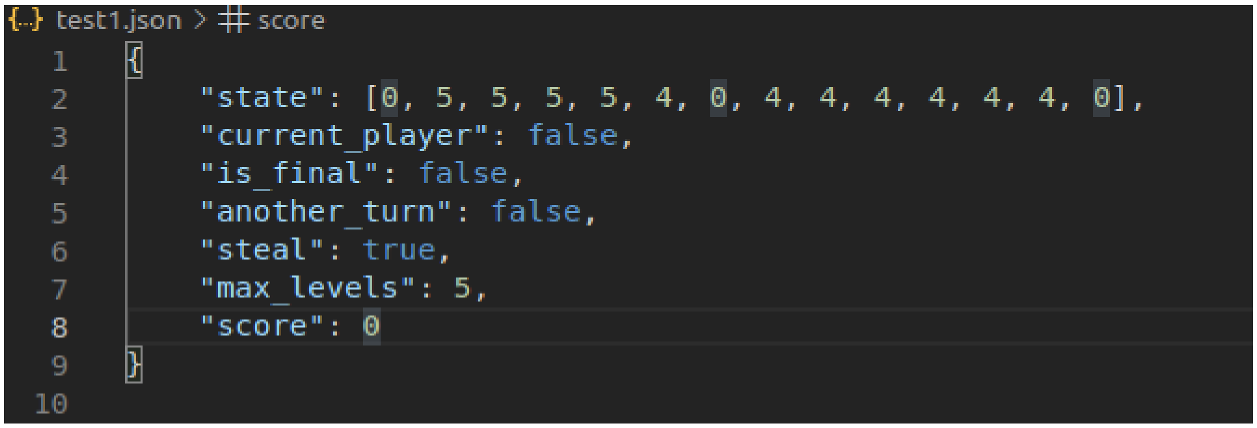
    3.2. Settings are loaded from the save game.

Screenshots:

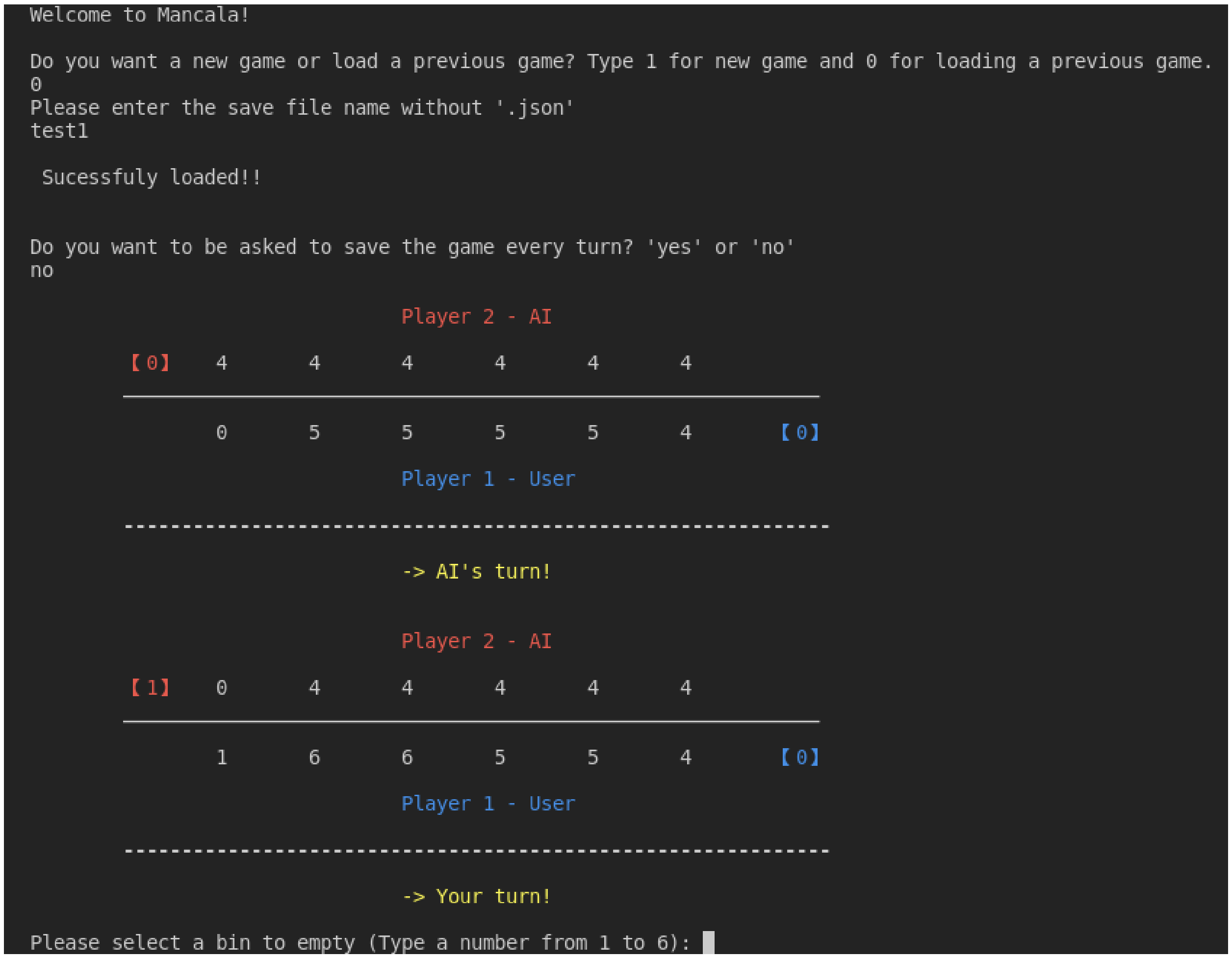
Sample of the game and usage of the save function



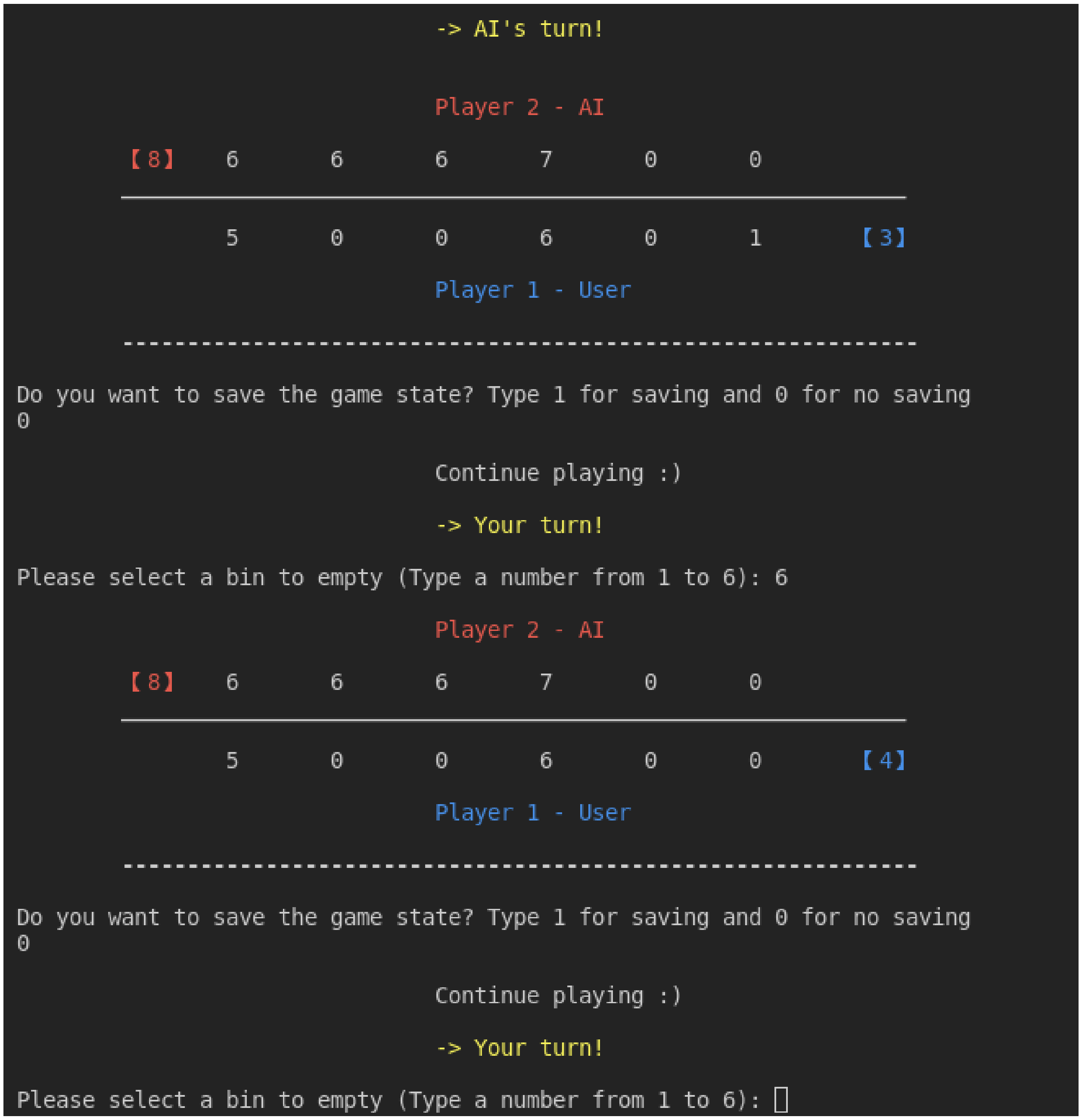
```
Welcome to Mancala!

Do you want a new game or load a previous game? Type 1 for new game and 0 for loading a previous game.
1
Select a difficulty (0 -> easy , 1 -> medium, 2 -> hard):
1
Do you want steal mode? Type 1 for steal and 0 for no steal
1
Do you want to start first? Type 'yes' or 'no'
yes
Do you want to be asked to save the game every turn? 'yes' or 'no'
yes

                                Player 2 - AI

        【0】   4       4       4       4       4       4
        ____________________________________________________________

               4       4       4       4       4       4       【0】

                                Player 1 - User

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

                                -> Your turn!

Please select a bin to empty (Type a number from 1 to 6): 1

                                Player 2 - AI

        【0】   4       4       4       4       4       4
        ____________________________________________________________

               0       5       5       5       5       4       【0】

                                Player 1 - User

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

Do you want to save the game state? Type 1 for saving and 0 for no saving
1
Enter the save name: test1

                                Game saved.

                                -> AI's turn!

```


```
{..} test1.json > # score
{
    "state": [0, 5, 5, 5, 5, 4, 0, 4, 4, 4, 4, 4, 4, 0],
    "current_player": false,
    "is_final": false,
    "another_turn": false,
    "steal": true,
    "max_levels": 5,
    "score": 0
}
```

Loading the game from a save file and AI makes a turn afterwise.

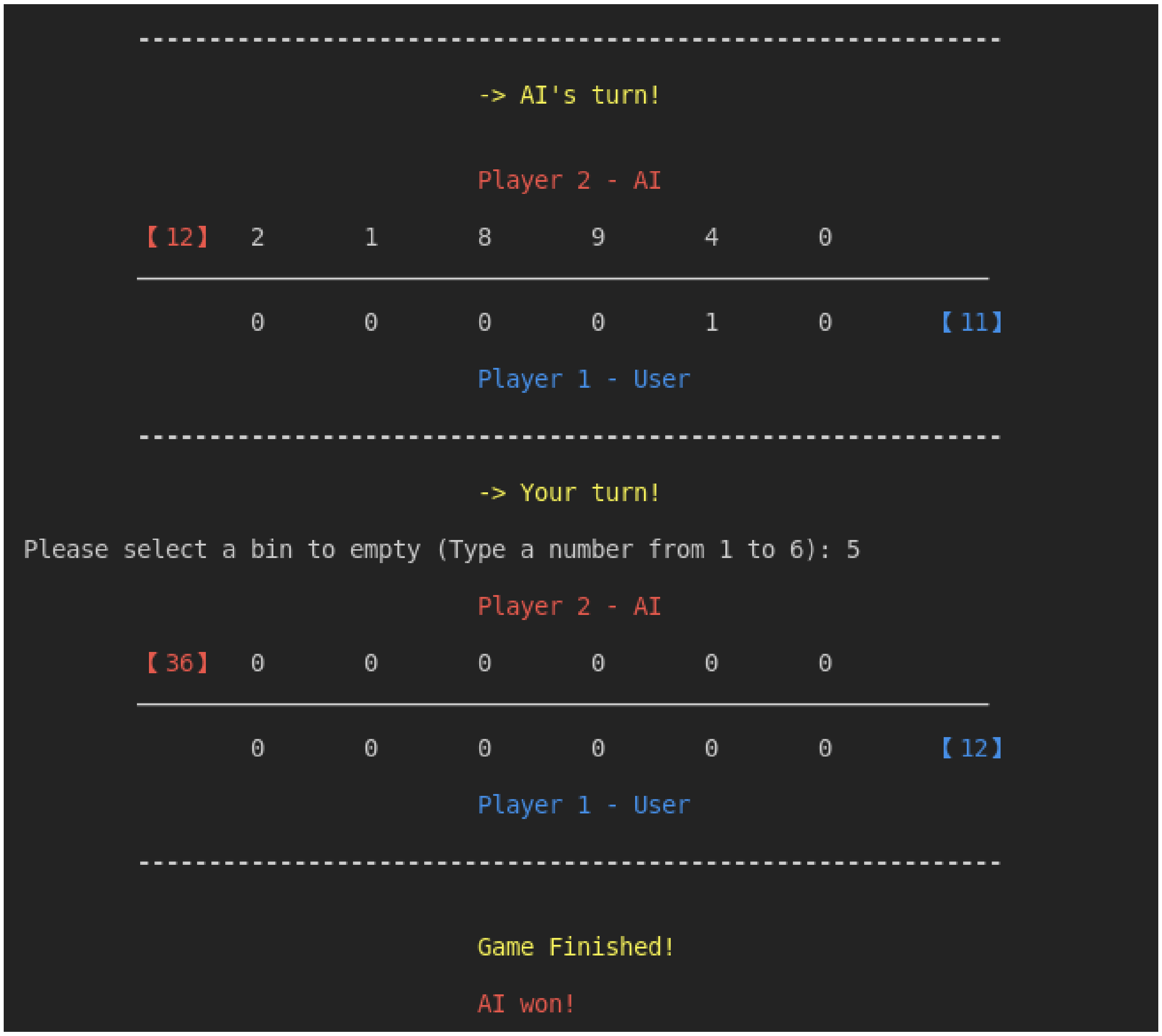

```
Welcome to Mancala!

Do you want a new game or load a previous game? Type 1 for new game and 0 for loading a previous game.
0
Please enter the save file name without '.json'
test1

 Sucessfuly loaded!!


Do you want to be asked to save the game every turn? 'yes' or 'no'
no

                              Player 2 - AI

         【 0】   4       4       4       4       4       4

         ____________________________________________________________

                 0       5       5       5       5       4      【 0】

                              Player 1 - User

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

                              -> AI's turn!

                              Player 2 - AI

         【 1】   0       4       4       4       4       4

         ____________________________________________________________

                 1       6       6       5       5       4      【 0】

                              Player 1 - User

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

                              -> Your turn!

Please select a bin to empty (Type a number from 1 to 6):
```

Screenshot of the user taking another turn after dropping the stone in the 6th pit in his mancala (as it is the last stone).



```
                                -> AI's turn!

                                Player 2 - AI

         【8】   6       6       6       7       0       0
         ____________________________________________________________

                 5       0       0       6       0       1       【3】
                                Player 1 - User

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

Do you want to save the game state? Type 1 for saving and 0 for no saving
0

                                Continue playing :)

                                -> Your turn!

Please select a bin to empty (Type a number from 1 to 6): 6

                                Player 2 - AI

         【8】   6       6       6       7       0       0
         ____________________________________________________________

                 5       0       0       6       0       0       【4】
                                Player 1 - User

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

Do you want to save the game state? Type 1 for saving and 0 for no saving
0

                                Continue playing :)

                                -> Your turn!

Please select a bin to empty (Type a number from 1 to 6):
```

Screenshot showing the ending of the game with the AI winning



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

                               -> AI's turn!


                               Player 2 - AI

          【12】  2       1       8       9       4       0
          ____________________________________________________________

                  0       0       0       0       1       0       【11】

                               Player 1 - User

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

                               -> Your turn!

Please select a bin to empty (Type a number from 1 to 6): 5

                               Player 2 - AI

          【36】  0       0       0       0       0       0
          ____________________________________________________________

                  0       0       0       0       0       0       【12】

                               Player 1 - User

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


                               Game Finished!

                               AI won!
```

4.0 Role of each member

- Abdallah Khaled Ahmed Mahmoud
    - Implemented the evaluation function.
- Omar Abdel Wahab Abdel Monem Ali
    - Implemented the main function.
- Omar Fathy Hamed Mahmoud
    - Implemented make_a_move function.
- Toka Abdul Hamied Abou-El Gheit
    - Implemented display_state, check_for_final, next_moves functions.
- Abdallah Nasser Abdallah
    - Implemented save_game, load_game functions.