



# Intelligent Mancala Game

Mohammed Emad Mahmoud

Mohamed Hussien Mostafa

Mohamed Amr Mohamed

Mohamed Amr Ahmed

Mohamed khaled rashad

Supervisor: Dr.Manal

**Department of Computer Systems Engineering**

Faculty of Engineering at Ain Shams University

Jun 2021

# Contents

<b>1</b>	<b>Brief Description of Mancala Game</b>	<b>3</b>
<b>2</b>	<b>How did we implement the game?</b>	<b>4</b>
<b>3</b>	<b>Utility Functions</b>	<b>5</b>
<b>4</b>	<b>A user guide with snapshots</b>	<b>7</b>
<b>5</b>	<b>Task Breakdown</b>	<b>8</b>

# List of Figures

4.1	cloning the repo . . . . .	7
-----	----------------------------	---

## Chapter 1

# Brief Description of Mancala Game

It's a board two-player game. The board is partitioned into 14 holes. Each player has 6 holes in front of them and a single hole for their score. Each player needs to collect as many stones as they can in their score hole to win the game.

## Chapter 2

# How did we implement the game?

We implemented each state of the game (number of stones in each hole) with a python tuple where `tuple[0]` is an array modeling the state of the board and the `tuple[1]` is the index of the hole played to get to that state.

## Chapter 3

# Utility Functions

1. `print_board(state)` :
  - prints the board to console based on passed state.
2. `let_user_player()` :
  - allows the system to get user play.
3. `convert_play_to_state(previous_state, play)` :
  - takes the state and the play chosen by human ,then convert the state to the new state according to the human play.
4. `get_node_children(in_state: list, stealing=True/False):`
  - it takes the current state of the game, and then calculating each possible state that can be got into from that current state (children of current state). Optional argument “stealing” if you pass it with true then stealing mode is activated, else its without stealing.
5. `evaluate(state)` :
  - evaluates the state by subtracting the number of stones in each side and the best state is the one that keeps your stones as much as possible.
6. `possibleMoves(state)` :
  - returns all possible plays that each player can play. In other words excludes the zero play (playing a hole which has no stones).
7. `extra_turn(state, selected_hole, player_type)` :

- checks if the player has to play again (the case when the last stone is dropped in the score hole).

8. `is_game_over(state)` :

- checks if the game is over by checking if there is a whole side with no stones.


9. `who_wins(state)` :

- checks who wins by subtracting the stones of each player and whoever has more stones wins.

## Chapter 4

# A user guide with snapshots

You can play our game by first cloning our repository [https://github.com/Mohammed-Hussien/Mancala\\_Ai.git](https://github.com/Mohammed-Hussien/Mancala_Ai.git). use the command `git clone` to clone the repo.

A terminal window with a dark background. The prompt is `C:\Mancala_Ai>` and the command entered is `git clone https://github.com/Mohammed-Hussien/Mancala_Ai.git`.

```
C:\Mancala_Ai>git clone https://github.com/Mohammed-Hussien/Mancala_Ai.git
```

Figure 4.1: cloning the repo

You will find a file called `mancala.exe`, just run that file and enjoy!



## Chapter 5

# Task Breakdown

1. Mohamed Hussien Mostafa Masaoud:
  - MiniMax Algorithm Implementation, Integration and game loop.
2. Mohamed Emad Mahmoud Abd Al Hamid:
  - AlphaBeta Algorithm Implementation, Integration and game loop.
3. Mohamed Amr Ahmed Taha:
  - Utility functions: who wins, is\_game\_over, convert\_paly\_to\_state, evaluate, possible\_moves.
4. Mohamed Amr Mohamed Hassan:
  - Utility functions: extra\_turn, possible\_moves, Integration and game loop.
5. Mohamed Khaled Rashad:
  - Utility functions: get\_node\_children, print\_board, Integration and game loop.