CPSC 481-02 Final Project

Project Title: Mancala
Group Members: Serop Kelkelian (888119963), Sydney Sukhabut (888512910), Gautham Vegeraju (886713619), Cyrus Traeh Baybay (887636819)

Problem: Implementing an AI player into the game of Mancala. Mancala is a deterministic 2-player game, where players select “pockets of seeds” to spread across the board while trying to collect the most seeds in their assigned stockpile. Many implementations of Mancala require two players playing locally or online, so we plan to create our own implementation that allows users to play against AI.

Programming Language:

We plan to use Python in our project. Python is the leading programming language for AI, and is very popular in game development. It’s clear syntax, flexibility and simplicity allows for group members to collaborate on code, while minimizing coding style differences and syntax errors.

The documentation can be found here: https://docs.python.org/3/

Datasets (if required):

Datasets are not required but we can use:

Generate saves and turns with random players: You can write a program that plays Mancala randomly, generates random moves for both players, and records the game state and corresponding moves. You can also add randomness to your board's initialization. Vary the number of seeds in each pit.

Generate data using the existing mancala engine: There are several Mancala engines available online, such as Mancala-Online and Bao La Kiswahili, which can be used to generate game states and movements. You can record your score and moves and use it as your own record.

Collect data from human players: Mancala gameplay data can be collected from a human player by creating an online platform or running experiments in a lab environment. The collected data can then be used as datasets.

After creating the dataset, you can use it to train a machine learning model for playing Mancala. Reinforcement learning algorithms such as Q-learning and Monte Carlo tree search can be used to train models to perform optimal movements in different game states.

Existing code:

We currently have a couple options of existing code we may use to supplement our project. We may not use all of them, but will select them based on their compatibility with our project goals and requirements. The options are listed below:

- **Big Book of Small Python Projects, #43 Mancala.** Al Sweigart has several books related to Python and Game Development with Python. In one of his books, he has a simple version of the Mancala game that we can expand on. The code can be found at: https://inventwithpython.com/bigbookpython/project43.html
- **AIMA Code.** We have been using the code from “Artificial Intelligence: A Modern Approach” throughout the semester. It has many different modules that create a good foundation for implementing new games and algorithms. We can use AIMA Code to structure and test our algorithms and game.
  The repository can be found at: https://github.com/aimacode/aima-python

Extensions to Add:

We are using the AIMA code from their GitHub repository to structure the coding for the Mancala game. This will make it easier for us to test the minimax algorithm.

Algorithm/Approach:

We are planning on creating a minimax algorithm for the game Mancala. The minimax algorithm is recursive and it is used in decision-making and game theory. It provides an optimal move for the player by assuming the opponent will also choose the best move possible. With a full game tree, we will be able to see which initial move will give the player the best chance at winning.

Timeline:

Below is our general timeline for our project. We have a primary focus for each week that creates flexibility for the group’s schedules, but also keeps us concentrated on the key parts of the project. We also plan to adjust as needed and move ahead if we finish certain elements early.

MAY

	1 **Brainstorm Project**	2 **Establish Approach**	3 **Propose Project**	4 **Code (focus on game)**	5 **Code (focus on game)**	6 **Code (focus on game)**
7 **Code (focus on game)**	8 **Code (focus on algorithm)**	9 **Code (focus on algorithm)**	10 **Code (focus on algorithm)**	11 **Code (focus on algorithm)**	12 **Code (focus on algorithm)**	13 **Code (focus on algorithm)**
14 **Debug**	15 **Debug**	16 **Debug**	17 **Debug**	18 **Compile Materials**	19 **Submit Project**	

Special Computing Platform:

At this time, we do not have a need for a Special Computing Platform. Our project is expected to execute successfully on standard OS, like Windows, macOS, & Linux.

Roles & Responsibilities:

- Implementing Game Mechanics - **Sydney**
  - revising existing game code to fit the requirements to create AI player
- Creating the Algorithm - **2**
  - Developing the min-max/alpha-beta pruning algorithm
- Connect and Implement - **1**
  - Implementing the AI algorithm to the game and verifying with test cases
- Debug/Clean-up/Documentation - **ALL**