Caleb Bettcher and Kenneth Flowe | Intro to AI Project Report 1 | 11/4/2025

So far we built our python game with both the “MinMax”AI and an algorithm that picks a random valid move. To test that everything worked, we ran 100 games where both players picked random moves with no strategy at all. That gave us a baseline to see if there was any first-move advantage. The results were pretty even overall, with Player 1 winning a little more often. We also capped the games at 500 turns so they couldn’t go on forever, which actually worked out fine most games ended naturally way before that.

After that, we ran 100 games of Minimax versus Random, and the difference was more noticeable. Minimax won 88 percent of the time and usually by a lot, with an average score of 33 to 15. The games averaged around 29 turns, so they were pretty quick. This lined up perfectly with what we expected. The Minimax player basically crushed the random one because it actually looks ahead instead of just tossing stones anywhere. Overall, the test showed our Mancala logic and AI both work exactly how they should.

When we ran 100 games of random versus random, the results were pretty even overall. Player 1 won about 48 percent of the games, Player 2 won around 44 percent, and the rest ended in ties. On average, each game lasted about 31 moves, and both players finished with close scores roughly 26 to 24. This showed that our game logic worked correctly and that the only real edge came from going first, which is exactly what we expected.

	Random VS Random	**MiniMax va Random**
Player 1 Wins %	48%	88%
Player 2 Wins %	45%	10%
Ties %	7%	2%
Avg Turns Per Game	31	32.72
Avg p1 Score	26	15.28
Avg p2 Score	24	29.1

To no surprise, MiniMax dominated the random choice algorithm. However, I expected MiniMax to win almost 100% of the time. Maybe there’s a mistake hiding in the code somewhere. There is clearly a first move advantage, of about 3-4% based on the Random vs Random trial.

Next steps are to implement a modified MiniMax algorithm (AB pruning), and potentially another algorithm (maybe A*, or another, more complicated solution). Our code is also structured in a way so that it’s very easy to expand upon with a class based system and code comments.