1. **The names of the members of your group**

Team Members: Andrew Rottier, acrottier@wpi.edu Barrett Wolfson, bwolfson@wpi.edu Kenneth Colpritt, [kcolpritt@wpi.edu](mailto:kcolpritt@wpi.edu)

Groupname: groupZ

1. **Instructions on compiling and running your program**

INSTRUCTIONS On how to run our AI program.

to compile: Open terminal and navigate to src folder in the project and use command:

javac GomokuAI.java

NOTE, if there is a warning, recompile with this command

javac -Xlint:unchecked GomokuAI.java

Once compiled, run the program with:

java GomokuAI

This will leave the program hanging in a loop, waiting for the groupZ.go file to be created. It will then read any moves in the move\_file and begin calculating its own move.

1. **The utility function that your program uses**

Our utility function takes into account our calculated heuristics from the board and the depth ahead. It looks at all possible states in the world and chooses the state which scored the best in our heuristic methods.

1. **The evaluation function that your program uses**

Our evaluation function were based on defensive moves. It checked the board for the opponent’s streaks and specific patterns. It look at all neighboring positions and evaluated the possible moves looking vertically, horizontally, and in both diagonal directions. It then counted how many patterns were matched in each direction. Some examples of patterns were \_XXXX, X\_XXX, XX\_X, etc where we looked for dangerous combinations of opponent moves in the linear direction. Depending on whose move it was, we checked for X’s or O’s. These states were then weighted based on the severity of matched patterns, where patterns that were 1 away from winning were weighted at 100, 2 away were weighted at 5, and 3 away were 1.

1. **The heuristics and/or strategies that you employed to decide how to expand (pieces of) the minimax tree without exceeding your time limit**

The main heuristic we employed to avoid expanding the entire minimax tree was alpha beta pruning within the minimax algorithm. If the algorithm sees certain scenarios where alpha > beta it can break prematurely with an alpha/beta cut-off and subsequently avoid expanding multiple subtrees of the entire minimax tree. Another strategy we used was setting the depth searched by the minimax algorithm as a parameter it consumes. During testing we could increase this number to search more combinations or decrease it to have the computer return a move decision more quickly.

1. **Results:**
   * **describe which tests you ran to try out your program. Did your program play against human players? Did your program play against itself? Did your program play against other programs? How did your program do during those games?**
   * **describe the strengths and the weaknesses of your program.**

Throughout the process of creating a functional AI for Gomoku we tested against multiple humans and different iterations of itself. Against humans it created logical moves that attempted to block and create ‘offensive’ moves to attempt to win the game. While playing against itself it started to be strictly defensive depending on the weight we had for our heuristics. It would attempt to block even if there wasn’t a major threat currently.

Strengths: The defensive capabilities of the program are definitely one of its strengths. When playing against human opponents, the program would continuous attempt to block streaks the human was building. Additionally, the variability of patterns included in our heuristic evaluation functions allows the program to evaluate the state of a board quite well. It can identify variations of patterns that are some distance away from a 5-piece streak win. For example, evaluating patterns that are one move away from a win it can identify “XXXX”, “XX\_XX”, “X\_XXX”, “XXX\_X” on any row, column, forward diagonal, or backward diagonal.

Weaknesses: It didn’t have any concept of offensive moves when playing against itself. The program would continue blocking and a string of 3 ‘X’s appeared yet it didn’t attempt to complete the row, column, or diagonal in its own favor. Another weakness was related to the blocking or defensive moves. The program would block well in a diagonal way, but it would often miss a horizontal or vertical string of the opponents.

1. **A discussion of why the evaluation function and the heuristic(s) you picked are good choices.**

Our evaluation function and choice of heuristics was a good choice because along we created weights which allowed us to make certain heuristics more important than others. This attempted to make sure that the opponent couldn’t win within the given depth of the minimax algorithm. The weights also contributed to deciding which next move would be most valuable if there were no immediate threats. The heuristics took into account a depth of how many moves to look ahead.