**CMPUT 355 2020**
**Assignment #4**

I started this assignment rather late, and didn't have a group, so all of the work was done by myself.

1. SuperHex!
2. Grayson Germsheid – 1545413 – germshei (Just Me)
3. N/A – All contributions by Grayson

4. Project GitHub: https://github.com/germshei/cmput355-assn4
   There should be a sample video in the submission folder

5. I decided to make a hex player / visualizer. The user is able to specify the dimensions of the board, as well as if they would like to play against a computer opponent or another player. Users can then take turns entering a cell they would like to mark until one has won the game. When playing against the computer, the player and computer alternate turns until one has won.
   https://www.hexwiki.net/index.php/Rules

6. My original goals were to:
   - Implement a Y / Hex player (Program would turn the Y-board into a Hex-board)
   - Implement an alpha-beta search algorithm to allow the program to determine the optimal move for a given player from a given state
   - Let the game be modular – the user should be able to choose the board size, whether they're playing against the computer or another player, as well as various other options

   Overall I would say that I was successful in everything but the first part – however implementing a Y board (or a geodesic-Y board as I tried) proved too difficult, so I settled on simply implementing Hex. Otherwise, all of the goals were achieved. The most disappointing part was not being able to implement or improve on more features (i.e. the alpha-beta pruning heuristic, algorithm complexity), but overall what was accomplished was satisfying. If I were to extend this program, I would implement it in a more board-agnostic way – making it a Y player – as well as improving usability and adding more options.

7. Various sample runs are included in the file statistics.txt with a variety of parameters.

8. Overall, I am generally happy with the quality of the project. One issue that I found was that the minmax search to find the optimal computer move does not always perform amazing – the search depth is too limited on larger boards for it to be a strong player, and I would have like to implement a better heuristic for scoring states (I went with the difference between shortest paths) as well as some sort of limitation and bias for selecting child nodes. As it stands, the only way to change the alpha-beta search is to limit its depth.

**Diary of work:**

I decided to pick this project because I found the game of Hex interesting and wanted to work on implementing a computer player that could play the game. Initially, I wanted to go with geodesic-Y, however I ran into too much trouble finding a way to represent the game state, so I settled on Hex since it could be held in a simple 2D array.

**Monday Nov. 16 – Approx. 8 hrs (Only started on the 18th)**
Most of this week was spent implementing the underlying game structure:

- Creating the data structures to track the game – state, players, turn #, etc…
- Implementing functionality to allow players to make moves
- Implementing turn order

What took a lot of time was implementing the functionality to print the board to the screen. It took me ages to work out an algorithm to turn a rectangular grid into a trapezoid composed of hexagons.

**Monday Nov. 23 – Approx. 24hrs**
This week was spent implementing the functionality to have a computer opponent. This was done via an alpha-beta minmax search of each possible move from the current state. The heuristic used is the difference in shortest-paths from the leaf state for the minimizing and maximizing players. Implementing the aB pruning was fairly straightforward, most of my work was on implementing a function to convert the array-representation of the board (that knows nothing about connections between cells) into a graph that could be used for Djikstra's algorithm. This algorithm is also used to check the win condition.

I also spent this week cleaning up the code, and implementing some options that can be set by the user (to set the board dimensions, the computer player, the use of the pie rule, as well as the maximum depth used by the alpha-beta pruning)