* Final Project Description

We present an implementation of algorithm described by AlphaGo Zero paper(1) which utilizes one single neural network and Monte Carlo Tree Search(2) to achieve superhuman proficiency in complete information games without prior human knowledge. We apply this approach to Connect 4(3), a popular game in Carnegie Mellon University Qatar’s food court. Furthermore, when we search the move that leads to greater or equal to 50 percent win rate for the A.I. at each game step, it closely simulates a perfectly balanced opponent.

* Description of features that will be demoed to the TA at the first milestone
  + At first milestone:
    - Connect 4 CLI implementation in Python 3.6 (for better support with deep learning frameworks)
    - GUI for Connect 4 with Tkinter
    - Implementation of classic Monte Carlo Tree Search algorithm
  + Final Demo:
    - Reinforcement Learning algorithm of AlphaGo Zero using Keras or Tensorflow
    - Algorithm to simulate a perfectly balanced opponent

1. AlphaGo Zero paper: <https://www.nature.com/articles/nature24270.epdf?author_access_token=VJXbVjaSHxFoctQQ4p2k4tRgN0jAjWel9jnR3ZoTv0PVW4gB86EEpGqTRDtpIz-2rmo8-KG06gqVobU5NSCFeHILHcVFUeMsbvwS-lxjqQGg98faovwjxeTUgZAUMnRQ>
2. MCTS (Monte Carlo Tree Search) is a relatively old heuristic search algorithm for decision processes. It is commonly used to create AI for games with incomplete information such as bridge, poker, and Go. In the context of AlphaGo Zero, MCTS can be viewed as a powerful policy improvement operator in addition to the policy neural network.
3. Connect 4 is a popular two player connection game with complete information on a seven-column, six-row game board. The objective of the game is to form a vertical, horizontal, or diagonal line with four pieces of the same color. If neither of the players win when all 42 slots are filled, the game is a draw.

Libraries expected to be used:

Tkinter (to make GUI of the game), Tensorflow/Keras and Numpy (to implement neural networks and reinforcement learning algorithm)