Aditya Agashe (ava9)
Susan Li (szl5) **Due: October 14, 2016**

CS 4701: Battleship Against Intelligent Adversary Project Proposal

Project Description:

1. **Goal:** We are looking to build a battleship game in Python. We plan on developing a text based REPL interface through the terminal for a user to play the game against the computer. We will write an AI algorithm to intelligently play against the user and make the best possible combination of moves.
2. **Game Play:** The user will start the game by running our program. The user will be asked to place their battleships on the board. The user will have 5 ships - one 5 block ship, one 4 block ship, two 3 block ship, one 2 block ship. The computer AI program will automatically randomly place the ships on its board. The user and the computer will then take turns guessing coordinates on the 10 x 10 board for locations of the opponent's battleships. Once a guess is made, the board will get updated. The REPL will display both boards - the user's board and the computer's board with the updated "hits". Once a player wins, the REPL will display the winner and ask the user if he/she wants to play again.
3. **Architecture:** The game consists of 4 main components
    a. The user - The user interacts with the Game State module through the REPL text interface
    b. Game state - This handles all attributes of game play, from managing the two boards, ships of the user and AI, to getting moves from the AI and calculating hits and misses. It interacts with the AI using the AI interface.
    c. AI - The AI handles making moves in response to user moves, the information of which is passed to it by the game state. It returns its moves to the game state through the AI interface.

General Approach:

The most basic battleship strategy is to select a random square. A heuristic that improves on this method is that you fire a shot at every nth square, where n is the length of the smallest ship still alive. We plan to improve upon this heuristic by using a game tree

Game tree search: Instead of simply picking the next n-parity square that is available, we will make our selection based on a game tree with one level. We will choose the n-parity square that if we miss, eliminates the maximum number of locations that the ship could still be.

We are willing to explore multiple levels of the game tree to see if this improves.

Aditya Agashe (ava9)
Susan Li (szl5) **Due: October 14, 2016**

System Evaluation Plan (Quantitative/Qualitative):

We will test the effectiveness of our Battleship AI by pitting it against increasingly more intelligent/difficult opponents. We test the Battleship AI by using naive algorithms as the easier opponents and intelligent human players as the most difficult opponents. We will then determine the probabilities of our Battleship AI:

1. Winning against naive (chooses next empty spot: A1 then A2 then A3)
2. Winning against random (chooses random empty spot)
3. Winning against weaker heuristic (chooses every second/third/fourth spot depending on which is the smallest ship still alive, chooses next available spot, e.g. A3, A6, A9, B2, B5….)
4. Winning against a combination of these approaches
5. Winning against human (We feel that Cornell CS students ought to be an intelligent human adversary)

Qualitative Evaluation: We will see what is the highest level of adversary our AI can beat, even one time. Will it be able to beat a human player even once?

Quantitative Evaluation: We will see what percentage of times our AI achieves a win against each level of adversary, e.g. does it beat a weaker heuristic 90% of the time but a human only 20% of the time?

Timeline:

- Oct 14 - Submit project proposal
- Oct 21 - Update project proposal based on professor's feedback
- Oct 24 - Decide on how we will traverse/utilize game tree to decide next move
- Oct 31 - AI fully built
- Nov 7 - Have naive, random, and one weaker heuristic AIs built
- Nov 14 - Test our AI against weaker AIs and win, test against humans
- Nov 21 - Test our AI against all levels of adversaries at least 100x each for computers, 10x for humans
- Nov 28 - Finish final product