CPSC 1020 **Computer Science I** Spring 2016
Programming Project 3 – 75 Points

Assignment Objectives
- Implement a project in C++ that processes command-line arguments
- Implement a project using arrays and string manipulation
- Implement a project that uses inheritance
- Utilize pair programming skills
- Practice problem-solving strategies to break down a problem

Delivery Instructions
- The following planning documents are due by 11PM on Friday, March 4th, to Canvas
  - Evidentiary support for steps 1 and 2 of Pólya's Method.
  - Rough outline of your strategy to solve the problem
- The project code files are due by 11:00PM on **Monday, March 21th**
- You should submit the specified files using Webhandin
  - You should have at least the following files to submit: `Hangman.cpp`, `HangmanBoard.cpp`, `HangmanGame.cpp`, `and makefile` with `clean`, `run`, and `compile` commands. You may opt to structure your classes using both `.h` and `.cpp` files
  - 12% off the maximum original point value is deducted for each 24-hour period the assignment is late for up to two days.

Approximate Grading Distribution
- Programming Style (10 points). Proper indentation, appropriate variable names, etc.
- Proper Documentation (10 point). Your file should include proper documentation as well as a descriptive comment header including your names.
- Planning Documents (15 points)
- Test Cases (35 points). Checking to see if your program works correctly for various input.
- Pair Programming Evaluation (5 points)

Collaboration Policy
For this assignment, you may not discuss any code with your classmates or any other human except your pair programming partner, course instructor, teaching staff, or ASC tutor. You may ask high-level conceptual questions via Piazza, but do not give direct answers to questions. You may use your textbook, class notes, lecture slides, and any materials from lab. You should not Google-search or look at any code on the Internet. All code should be written by both partners sitting side-by-side using proper pair programming method.

The Project
In this project, each team will be required to design and implement a souped-up version of the classic game, Hangman. Hangman is traditionally a pen-and-paper game where one player thinks of a word and the second person tries to guess it by suggesting letters. The word to guess is represented by "blanks" where the number of blanks is equal to the number of letters in the word. If the guessing player correctly guesses a letter in the word, the other player will write all occurrences of that letter in the word. If the

guessed letter is not in the word, the other player will draw one element in the hangman diagram, which is a rudimentary image depicting a hanging man.

The game ends when either 1) the guesser has exhausted the number of misses (6: head, torso, right arm, left arm, right leg, left leg) or 2) correctly guesses the word

You will implement capabilities for both human and computer players; that is, you can have one of the following combinations:

Computer (guesser)	Human (hangman operator)
Human (guesser)	Human (hangman operator)

Project Requirements

- Each team must design and implement the following classes named as they are stated below. You may choose to add more classes, if necessary:
    - `HangmanBoard`: a class that implements the functionality of updating/operating the board
    - `HangmanGame`: a class that implements the functionality of playing a game of Hangman
    - `Hangman`: a class that contains the main method to run Hangman Game
- At the initial game, you must display "Hangman++" and an empty gallows on the console using output statements. DO NOT USE ANY GUI COMPONENTS TO DRAW THE BOARD. Drawing the gallows and hanging man will be an exercise in your ASCII art creativity; feel free to be as creative (or not) as you want, as long as the image displays properly in a terminal running at 150x40 columns (note that's columns, not pixels)
- Since the human is the hangman operator, the human player needs to think of a word and input the number of blanks needed (if the human is the hangman operator, assume that no special characters aside from A-Z are used). Once the number of characters of the word the human has selected has been inputted, the gallows and blanks should be displayed as indicated above. In this mode, you will need to provide a prompt for the hangman operator to indicate whether a guess was correct (and if so, which blank(s)).
- You'll need to determine a strategy for the computer to guess the word correctly.
- If a human is the guesser, the human needs to be prompted to guess a letter. If the letter is a "hit", it is revealed in the blank where appropriate. If it's a miss, the man is further hanged.
- After a game is completed, you should prompt to see if another game is desired to be played. If so, you need to see if any roles are to be changed, and how many blanks are needed.
- When drawing your gallows, make sure that to the right (this makes it easier for you) of the hanged man, you indicate the missed letters. You may put the blanks anywhere you wish, as long as it looks "acceptable."
- The `Hangman` class must have a main method that can process input parameters.
    - The first argument (required) is for who is playing the initial role of a guesser: human or computer
    - The second argument is optional and it is an integer number indicating the number of blanks. If this is not indicated, you will need to prompt the human operator for the number of blanks.

Sample Program Calls

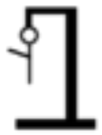
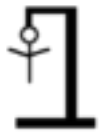
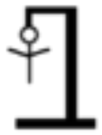
```
prompt% ./Hangman –L human
prompt% ./Hangman human –L
prompt% ./Hangman human 5
```

Extra Credit

Up to 20 points of extra credit may be awarded to the top 3 teams who develop the "best" successful computer guessing strategies. More details on this will be announced in class.

Appendix

Let's take a look at a sample run (which is excerpted from the Wikipedia entry on "Hangman", found at http://en.wikipedia.org/wiki/Hangman_(game)

Word: _ _ _ _ _ _ _ _ Guess: E Misses:	
Word: _ _ _ _ _ _ _ _ Guess: T Misses: e	
Word: _ _ _ _ _ _ _ _ Guess: A Misses: e,t	
Word: _ **A** _ _ _ **A** _ Guess: O Misses: e,t	
Word: _ **A** _ _ _ **A** _ Guess: I Misses: e,o,t	
Word: _ **A** _ _ _ **A** _ Guess: N Misses: e,i,o,t	

Word: _ **A N _ _ A N** Guess: S Misses: e,i,o,t	
Word: _ **A N _ _ A N** Guess: H Misses: e,i,o,s,t	
Word: **H A N _ _ A N** Guess: R Misses: e,i,o,s,t	
Word: **H A N _ _ A N** Guess: Misses: e,i,o,r,s,t GUESSER LOSES ☹	