CS 118 — Programming Fundamentals
Assignment #7

Due Date: Wednesday, November 20th at 11:55pm on Google Classroom.

Instructions: Assignments are to be done individually. **No late assignments will be accepted.** You must complete this assignment by yourself. You cannot work with anyone else in the class or with someone outside of the class. The code your write must be your own. You are encouraged to get help from the instructional staff. You may post general questions on Piazza. Do not post more than one line of code when using Piazza.

You must **submit a single zip file** containing your code and documentation on Google Classroom named ⟨*your_student_id*⟩*.zip* where ⟨*your_student_id*⟩ is something like *i19-XXXX*. This means that you must submit only **one file named *i19-XXXX.zip* containing only your source files**. Each file that you submit **must contain your name, student-id, and assignment#** on top of the file in comments. You submission must NOT contain multiple main() functions, otherwise it will not compile for grading. Test your program on a lab machine before submission.

Follow the instructions. Assignments not following the instructions will be awarded zero points.

Assignment Statement: In this assignment you will write a program to play Hangman. The program picks a secret phrase from a list of phrases or words. The human player guesses letters until they reveal the entire phrase or lose due to picking too many letters that are not in the phrase.

There are four sample files: *hangman.out1.txt*, *hangman.out2.txt*, *hangman.out3.txt* (shortest sample, demonstrates a loss), and *hangman.out4.txt* (various bad inputs). Use a diff tool to ensure your program produces the correct output. Even minor differences in output will cause you to fail grading tests and lose points.

Note, the phrase bank always picks topics in the random order. This is necessary for consistency when testing. Do not modify the header file.

Your program must:

1. Display an intro (This is already done.)
2. Create a new phrase bank. (This is already done in the init method of the hangman.h header file). The phrase bank can be accessed using two methods for you to use, *getGenre*() which returns a string containing the topic name and *getPhrase*() which returns a string that is the secret phrase for the game. Each secret phrase contains only letters and hyphens and spaces with only the letters missing.
3. The program then plays a round of Hangman until the user wins or loses. Use a while loop for this.
4. At the start of each game the program gets a secret phrase from the phrase bank.
5. In each game, the program displays the current form of the secret phrase. Spaces are shown as spaces and hyphens as hyphens. Letters that have not been guessed are shown as asterisks. Letters that have been guessed are shown.
6. The program then displays the letters that have not been guessed so far in alphabetical order in uppercase.

7. Next the program prompts the user for the next guess, a single letter. If the value entered is not a letter or it is a letter that has already been guessed the program prompts the user again until they enter a letter that has not been guessed yet. It is possible the user enters more than a single character or a character that has already been guessed. See output files for handling these cases.

You can assume the user will always enter at least one non whitespace character. If they enter more than a single character choose the first character entered. If the first character is a lower case letter, convert it to upper case. Ensure that the user input is valid.

The results of the guess are shown. If the guessed letter appears in the secret phrase all instances of that letter are revealed. If the guessed letter does not appear the number of wrong guesses is incremented.

If all of the letters in the secret phrase are revealed the player wins. If the player makes 5 wrong guesses they lose.

After each incorrect guess the program should print hangman using the method $showHangman()$ which takes the number of mistakes minus one as the parameter.

Approach: Divide the program into parts. Complete and test each part before moving on. Use the methods from string and ctype libraries to help make your job easier.

Divide the program up into methods to provide a structured solution. Some of the methods will be void and others will return values. You will have to make use of parameters, for loops, while loops, and if statements. Do not write all your code in main().

Honor Policy

This assignment is a individual learning opportunity that will be evaluated based on your ability to think independently, work through a problem in a logical manner solve the problems on your own. You may however discuss verbally or via email the general nature of the conceptual problem to be solved with your classmates or the course instructor, but you are to complete the actual assignment without resorting to help from any other person or other resources that are not authorized as part of this course. If in doubt, ask the course instructor. You may not use the Internet to search for solutions to the problem.