Assignment 3
Due: Sunday, 4 February 2018, 11:59 PM on TEACH as a .cpp
Error Handling and Helper Function Library

Users are seldom perfect. It is our job as programmers to catch mistakes and correct them to prevent crashing the program. The table below illustrates common functions you will use for error handling and working with user in the CS 16X series.

Design (Recitation)
You will design solutions to each of the functions outlined in the table. Make sure to include a Understanding the Problem section and a Testing section in your design. Please consult the design syllabus for additional information. Tips for designing this assignment:

- Don't worry about functions to start
- Treat each function as a mini program
- Design the solution to each problem using loops, conditionals and what you know about strings
- Could be helpful to put each solution on a notecard to simulate a function
- During week 3 you do not need to know about functions to successfully design solutions to each problem

(50 pts) Implementation
Implement each function in the table. You are not permitted to use any built in functions. You may use the string library and string type but you are not permitted to use any of the other helper functions found in that library. You are encouraged to use the ASCII chart.
Grading:

- Each function is worth 3 points.
- You will receive 2 points by default.
- All functions must follow the outlined naming conventions, input and output values shown below in the table. Failure to do so will result in a zero for that function.

Function Name	**Input (Parameters)**	**Output (Return Type)**	**Description/Notes**
check_range	int lower_bound int upper_bound int test_value	Boolean	Indicates if the provided number is in the specified range
is_int	string num	Boolean	Indicates if a given string is an integer

is_float	string num	Boolean	Indicates if a given string is a float
is_capital	char letter	Boolean	Indicates if a given character is a capital letter
is_even	int num	Boolean	Indicates if a given number is even
is_odd	int num	Boolean	Indicates if a given number is odd
equality_test	int num1 int num2	Int	Tests num1 against num2 and returns -1 if num1 < num2, returns 0 if num1 == num2, returns 1 if num1 > num2
float_is_equal	float num1 float num2 float precision	Boolean	Tests if num1 and num2 are equal to each other within a certain precision
numbers_present	string sentence	Boolean	Indicates if the provided string contains numbers
letters_present	string sentence	Boolean	Indicates if the provided string contains letters
contains_sub_string	string sentence string sub_string	Boolean	Indicates if substring exists in sentence
word_count	string sentence	Int	Provides the number of words in a given string

to_upper	string sentence	String	Capitalizes all letters in a given string and leave all non-letter characters unchanged
to_lower	string sentence	String	Makes all letters lowercase in a given string
get_int	string prompt	Int	Takes a prompt from the user as a string literal, checks if input is a valid integer, returns the provided integer
get_float	string prompt	Float	Takes a prompt from the user as a string literal, checks if input is a valid float, returns the provided float

(35 pts) Testing

Main() will serve as the location where you test your functions. For each function you must show each return option is reachable. For example, is_int() should be run twice: once to show that it can successfully identify a string as an int and return true; once to show that is can successfully identify the string is not an int and return false. You must label your tests, indicating what you are testing, the value you are providing, the expected output and the actual output. If the expected output matches the actual output, the program should print "PASS" for the test and "FAIL" otherwise. Your print out should be readable. 28 points come from the actual tests (one for each return). The remaining 7 points comes from clean output (how readable the print statements are).

(15 pts) Program Style/Comments

In your implementation, make sure that you include a program header and function headers in your program, in addition to proper indentation/spacing and other comments. Below is an example header to include. Make sure you review the style guidelines for this class, and begin trying to follow them, i.e. don't align everything on the left or put everything on one line!

http://classes.engr.oregonstate.edu/eecs/winter2018/cs161-001/assignments/161_style_guideline.pdf

```
/*********************************************************************
** Function:
** Description:
** Parameters:
** Pre-Conditions:
** Post-Conditions:
*********************************************************************/
```

(10 points) Extra Credit (Choose One)

Option 1:

The above functions solve problems you will likely face in future assignments. One method for reusing these functions is to copy and paste them into each assignment. A better way is to make this set of functions into a library which you can include in future assignments but this is a concept we do not usually cover until CS 162.

Using the link below for reference, create your own library from these functions by separating your assignment into three files: helper_functions.h, helper_functions.cpp, and assignment3.cpp Include the helper_functions.h in your assignment3.cpp and compile using the following line:

```
g++ assignment3.cpp helper_functions.cpp -o assignment3
```

Resources (start on slide 7):
http://classes.engr.oregonstate.edu/eecs/spring2017/cs162-001/calendar/slide2-notes.pdf

Option 2:

Testing is a very important part of programming. Fully test your is_int and is_float functions. Test as many inputs as you can imagine. See how you can break your functions and if you do how are you going to fix it. Use the following hints to get you started:

- What happens if you have multiple digits?
- What happens if there is a sign at the beginning/end?
- What happens if it is a string with letters and numbers?
- What if you don't supply digits after a dot?

Record your tests cases the same way you did in main.

Electronically submit your C++ program (**.cpp** file, not your executable) by the assignment due date, using TEACH.
https://secure.engr.oregonstate.edu:8000/teach.php?type=want_auth

Make sure you demo Assignment #3 within two weeks of the due date to receive full credit. If you go outside the two week limit without permission, you will lose 50 points. If you fail to show up for your demo without informing anyone, then you will automatically lose 10 points.