Assignment 2 Error Handling and Helper Function Library

Design Due: Sunday, 2/7/2021, 11:59pm on Canvas

Code Due: Sunday, 2/14/2021, 11:59pm on TEACH

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.

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_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 (hard coded)
is_int	string num	Boolean	Indicates if a given string is an integer
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 and leave all non letter characters unchanged
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

Design Document – Due Sunday 2/7/2021, 11:59pm on Canvas
Refer to the Example Design Document – Example_Design_Doc.pdf

Design solutions to each of the functions outlined in the table. Make sure to include an 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 4 you do not need to know about functions to successfully design solutions to each problem

Guiding questions:

Understanding the Problem/Problem Analysis:

- What are the user inputs, program outputs, etc.?
- What assumptions are you making?
- What are all the tasks and subtasks in this problem?

Program Design:

- What does the overall big picture of each function look like? (Flowchart or pseudocode) ○ What data do you need to create, when you read input from the user?
  - How to name your variables?
  - What are the decisions that need to be made in this program?
  - What tasks are repeated?

Based on your answers above, list the **specific steps or provide a flowchart** of what is needed to create. Be very explicit!!!

Program Testing:

Create a test plan with the test cases (bad, good, and edge cases). What do you hope to be the expected results?

- What are the good, bad, and edge cases for ALL input in the program? Make sure to provide enough of each and for all different inputs you get from the user.

Electronically submit your Design Doc (.pdf file!!!) by the design due date, on Canvas.

Program Code – Due Sunday, 2/14/2021, 11:59pm on TEACH

Implementation Requirements:

- Implement each function in the table
- No built-in functions are allowed, except for:
  - From <iostream>: cin.clear(), cin.ignore(), cin.fail()
  - From <string>: getline(), .length(), size(), [], .at(), +=
  - From <cmath>: pow(), abs()

*Note: Implement your own version of other built-in functions is allowed. i.e. a_to_i()

- Typecasting allowed only if the character being converted fits the stated criterion (i.e. char was confirmed as an int, letter, etc.)
- You are encouraged to use the ASCII table: http://www.asciitable.com/
- Your user interface must provide clear instructions for the user and information about the data being presented
- No global variables allowed (those declared outside of many or any other function, global constants are allowed).

Note: get_int() is the only function which will get input from the user. The rest will be hard coded arguments to your functions.

Testing Requirements:
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. Below is the example code to test is_int():

```
cout << "\nTesting is_int(\"34\")...\n";
cout << "Expected: 1";
cout << "\tAcutal: " << is_int("34");
 (is_int("34")==1)? cout << "\tPASSED\n" : cout << "\tFAILED\n";

cout << "\nTesting is_int(\"abc\")...\n";
cout << "Expected: 0";
cout << "\tAcutal: " << is_int("abc");
 (is_int("abc")==0)? cout << "\tPASSED\n" : cout << "\tFAILED\n";
```

Program Style/Comments Requirements:
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 program header and function 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!

```
/*********************************************************************
* ** Program: assignment2.cpp
** Author: Your Name
** Date: 01/01/2020
```

```
** Description:
** Input:
** Output:
*********************************************************************/

/*********************************************************************
* ** Function: is_int()
** Description: Indicate if a given string is an integer
** Parameters: string num
** Pre-conditions: take a string parameter
** Post-conditions: return a value to tell if the string is an int
*********************************************************************
/
```

(Extra credit: 10 pts)

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 `assignment2.cpp`.

Include the `helper_functions.h` in your `assignment2.cpp` and compile using the following line: `g++ assignment2.cpp helper_functions.cpp -o assignment2`

Electronically submit your C++ program (.cpp file, not your executable!!!) by the code due date, on TEACH. The rubric worth 100 points will be released after design due date.

Remember to sign up with a TA on Canvas to demo your assignment.

- **If you go outside the two-week limit without permission, you will lose 50% of the points of this assignment.**
- **If you fail to show up for your demo without informing anyone, then you will automatically lose 10% of the points of this assignment.**