Design Document Assignment 2

Name: Tom Nyuma
ID: 934044293
CS161 Section 015
Date: 02/2/2021

Problem Statement: We are to write a C++ program that tests many of the common error handling functions that will be used throughout the CS16X series.

Understanding the Problem To successfully obtain the maximum amount of points on the assignment, our program must closely follow the outlined implementation requirements. get_int() will be the only function that takes user input, so I will be using a void() function to give the user input options, as it's the last function to be tested anyways. Everything else will be tested in main().

Pseudocode:

/*************************
Program Filename: assignment2.cpp
Author: Tom Nyuma
Date: 02/02/2021
Description: Program to design and test various error handling functions.
Input : int low, int high, int x, int letter, int num, int num1, int num2, float num1, float num2, ifloat precision, string num, string sentence, string sub_string, string prompt.
Output: A mix between Booleans, Integers, and strings.
*************************/

```
check_range(low, high, target) ->(return) a Boolean
      return (target<=low and target >= high)

// This function will return the result of an expression checking if our
target is between low and high.

is_capital(letter) -> a Boolean
      return (letter >= 'A' && letter <= 'Z')

// Since we know c++ uses ASCII, this function checks our input (char letter)
for being in the ASCII interval of uppercase characters.

is_even(num) -> a Boolean
      num % 2 ? Even or Odd
// Our is_even function takes advantage of the modulus operator to divide our
input by 2 and if the remainder is 0 we know it is even using basic math.
```

is_odd(num) -> a Boolean

// This function has the same structure as the is_even() function, but this time uses modulo 3 to check if the number is odd.

equality_test(num1 , num2) -> an Integer
	is num2 less than num1?
		return -1
	is num2 greater than num1?
		return 1
	otherwise?
		return 0

// This function executes an equality test. It returns -1 if num1 < num2, 0 if num1 == num2 and 1 if num1 > num2

float_is_equal(num1,num2,precision) -> Boolean
	int1
	int2

	if num1 is greater than 0
		int1 is (typecast int)(num1 * precision + .5)
	otherwise
		int1 is (typecast int)(num1 * precision - .5)

	if num2 is greater than 0
		int2 is (typecast int)(num2 * precision + .5)
	otherwise
		int 2 is (typecast int)(num2 * precision - .5)

	return (int1==int2)

// This function will check if num1 and num2 are equal to each other within a certain precision (decimal placing) by multiplying the floats (num1,num2) by the desired precision and casting them to ints. It also accounts for rounding hence the +- .5 in our expressions.

is_int(string num) -> Boolean
	for each character in num
		if num[0] == '-'
			continue
		if !(num[iterator] >= '0' and num[iterator] <= '9')
			return false

```
            return 1

// What this function does is check for a string being an integer. It
iterates over the input string using a for loop and checks each input for
being a character 0-9. if it is a negative integer and contains the character
'-', then we skip it and continue our for loop iteration.

number_present(string sentence) -> Boolean
      variable counter
      for character in sentence
            if sentence[character] >= '0' and sentence[character] >= '9'
                  counter += 1
            otherwise
                  return 0
      if counter >= 1
            return 1
      return 0

// This function, number_present() utilizes a counter to check if there is a
number present in the input. The for loop iterates through the string and if
there is a digit in between 0-9 then we increment the counter. If the counter
is ever greater than or equal to one we return 1 because a number is present.

letters_present(string sentence) -> Boolean
      int counter
      for char in sentence
            if sentence[character] >= 'A' and sentence[character] <= 'Z' or
sentence[character] >= 'a' and sentence[character] <= 'z'
                  counter += 1
            otherwise
                  return 0
      if counter >= 1
            return 1
      return 0

// Similar to the numbers_present() function, this function executes the same
set of statements and boolean logic but for the letters in the ascii table
instead of the numbers. The function has to check for lower and upper case
character inputs so we have to include the conditionals to check for it
appropriately.


contains_sub_string(string sentence, string substring) -> Boolean
      int x = 0
      for char in substring
```

```
            if sentence[x] == substring[char]
                  x = x + 1
            if x == sentence.length()
                  return 1
      return false
```

// What this function does is test for a substring being present in a string. it iterates through the substring and if the substring's current index equals the sentence index's character, it increments the pointer. If the pointer is equal to the length of the substring we know that the substring exist because we covered all possible characters in the substring.

word_count(string sentence) -> Integer

```
      int counter = 0
      bool in_whitespace = true
      for char in sentence
            if sentence[character] == ''
                  in_whitespace = true
            if sentence[character] >= 'A' and sentence[character] <= 'Z' or
sentence[character] >= 'a' and sentence[character] <= 'z'
                  counter = counter + 1
                  in_whitespace = false
            else
                  continue
      return counter
```

// This function checks for the number of words in a given string. It increments through our input string and checks on each iteration if a word is present. If there is no whitespace, which is under the boolean in_whitespace, then our iteration continues. At the end of our for loop, we return counter

to_upper(string sentence) -> String

```
      for i=0 str[i] != 0 i++
            if str[i] <= 'z' && str[i[ >= 'a'
                  str[i] += 'str[i]'-'str[i'
            else
                  continue
      return sentence
```

// This function iterates through our input string's characters and on each iteration converts those characters to uppercase. If there is a non character present at the current index, it continues.

to_lower(string sentence) -> String

```
      for i=0 str[i] != 0 i++
            if str[i] <= 'z' && str[i[ >= 'a'
                  str[i] += 'str[i]'-'str[i'
            else
                  continue
      return sentence
```

// What this function does is iterate through our input string's characters and on each iteration converts those characters to lowercase. If there is a non character present at the current index, it continues.

get_int(string prompt) -> Int

```
      int choice
      bool error
      do {
            error = false
            if (cin.fail()){error = true}
            cin.clear()
            cin.ignore(1000,'\n')
      } while (error)
      std::cout << choice
      cin.get()
```

// This function, get_int takes a string input and checks if the input is a valid integer by making use of the cin() function that normally would be checking for an integer, so if an error is found, we return the integer.

Testing

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