Assignment 3 Handle Fractions,Text Surgeon and Print Patterns

Design Due: Sunday, 02/21/2020, 11:59pm on Canvas

Code Due: Sunday, 02/28/2020, 11:59pm on TEACH

Make sure you demo Assignment 2 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% of that assignment. If you fail to show up for your demo without informing anyone, then you will automatically lose 10% of that assignment.

This assignment consists of four parts. Out of which Part A and Part B are requirements of this assignment, whereas Part C and Part D are for extra credit. You will need to submit all programs to TEACH.

Part A : Fractions in Lowest Terms (60 points)

Problem Statement:

You are tasked to write a program which takes a numerator and a denominator as a user input and reduces the fraction to its lowest terms. For example, if a user inputs 48 as numerator, and 9 as denominator, then your program should output 16/3. This will require finding the greatest common divisor (gcd) for the numerator and denominator, then dividing both by that number. If the denominator is 0, an error message should be printed. At the end of each calculation, the user should be asked if they want to do another conversion or exit the program. The program must handle all types of bad inputs from users and recover from the errors.

Example run:

```
Enter your numerator: 4.6
Invalid input, enter whole numbers only!
Enter your numerator: -9
Enter your denominator: 45

The lowest terms of your fraction: -1/5

Do you want to do another conversion? (0-no, 1-yes): 1

Enter your numerator: 8
Enter your denominator: abc8
Invalid input, enter whole numbers only!
Enter your denominator: 0
Invalid input, denominator cannot be 0!
Enter your denominator: 2

The lowest terms of your fraction: 4

Do you want to do another conversion? (0-no, 1-yes): 0
```

Required function(s):

Your program must involve the following functions. You may not change the parameters or the return type!!!

```
//return false if denominator is 0, return true otherwise
//Reduce both numerator and denominator to lowest terms inside the
function bool to_lowest_terms(int &numerator, int &denominator);
```
OR
```
bool to_lowest_terms(int *numerator, int *denominator);
//return the greatest common divisor of num1 and num2
int g_c_d(int num1, int num2); - this must be a recursive function, and you
will need to explain the algorithm during demo
```
Note: If you use an algorithm that requires a third parameter, that is also fine.

Part B : Using C-style strings (40 points)

Problem statement:

You are tasked to write a program that reads in a sentence or paragraph (max length 1023 letters) from the user, store it as a C-style string, and gives the user several options for operations to perform on the text or quit. After performing an operation, the user should be prompted to select another option until he or she selects the option to quit. The program must handle all types of bad inputs from users and recover from the errors. Detailed requirements for each choice are listed below:

1. **Vowels vs. Consonants:** Check if the number of vowels equals the number of consonants. Ignore any non letter characters.
   Example: User enters "`banana`"; program prints " `# vowels = # consonants.`"
   Example: User enters "`ban ana44!`"; program prints " `# vowels = # consonants.`" Example: User enters "`zip`"; program prints "`# vowels != # consonants.`"

2. **Palindrome Detector:** Determine whether the input C-style string is palindrome or not. A palindrome is a word, phrase, or sequence that reads the same backwards as forward. E.g., racecar, madam.
   Examples of palindromes that your code needs to detect:
   User enters "`11Radar911`", program outputs "`This is a palindrome.`" (Ignore the numbers and case of letters)
   User enters "`top sports`", program outputs "`This is not a palindrome.`"
   User enters "`Was it a car or a cat I saw?`", program outputs "`This is a palindrome.`" (Ignore the space and special characters, '?')

3. **Words frequency**: Prompt the user for N words that the user wants to search for in the input string. Use a dynamic C++ string array allocated on the heap to store N words, and output the frequency of N given words in the input string.
   Example: User enters "`This is a test sentence, is this?`", then 3 words, "`this`", "`text`", and "`test`". The program outputs:
   ```
   this: 2
   text: 0
   test: 1
   ```
   (Note: Case insensitive, meaning "word" and "WoRD" are the same; All numbers and special characters in the string are ignored, meaning "wor32rd", "wo]]]rd343.34", and "word" are the same)

Suggested functions:

a. **`bool is_palindrome (char *str);`** //return true if str is palindrome, false otherwise b. **`char * purge_string (char *str);`** //accepts a c-string, and returns a version where all numbers, spaces, and special characters have been removed

(Note: the returned c-string must be on the heap, otherwise this c-string will be deleted once we leave this function)

OR you may use this:

`void purge_string (char *str, char* str_new);` //accepts two c-strings, remove all spaces and special characters in str, and store the new string into str_new

***Important note:**

You are not allowed to convert a C-style string to a C++ string object. In other words, you have to treat the input sentence (paragraph) as a C-style string and parse through it.

For frequency of given words, the N words must be entered at one time before searching to see if the words are in the sentence/paragraph. In other words, you cannot ask for a word, search its frequency and then ask for another word. You must ask for all words before searching for the words. You must use a dynamic array allocated on the heap!!! You will **NOT** be given credit for a variable length array, which is not dynamically allocated on the heap, i.e. `string array[num_words];` **is not accepted for choice 3 : Word Frequency!!!!** In addition, you must not have memory leaks (use valgrind to help)!

Extra Credits

Part C : Number Pattern Printing (20 points)

Write a program to print patterns. **You need to use loops or recursion to print out the following patterns.** The program will first ask the user which pattern to print, pattern of numbers, and then print the pattern accordingly (including the correct number of spaces in each line). At the end of each execution, the user should be asked if they want to print another pattern or exit the program. **No error handling is needed for this part**. Requirements for pattern are shown below:

The user will be asked for another input, indicating the number of rows of numbers (you can assume this value to be a positive odd number).

Example output: `row = 9`

```
      1
     123
    12345
   1234567
  123456789
   1234567
    12345
     123
      1
```

Part D : Use C-style strings for words (10 points)

For Word Frequency (choice 3) in Part B, use an array of C-style strings for the words, instead of C++ string objects. (**Note: Either make the words C-style strings for extra credit OR an array of C++ strings, not both)

Design Document – Due Sunday 2/21/20, 11:59pm on Canvas
Refer to the Example Design Document – Example_Design_Doc.pdf

You need to provide a design for both parts A and B! Design for extra credit Part C and Part D are not necessary but you are welcome to try out.

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?
    - How would you modularize the program, how many functions are you going to create, and what are they?
- What kind of bad input are you going to handle?

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 Monday, 2/28/20, 11:59pm on TEACH

Implementation Requirements:

- Must produce two working programs (Part A and Part B) that follows each rule stated above
- Your user interface must provide clear instructions for the user and information about the data being presented
- Your program should be properly decomposed into tasks and subtasks using functions. To help you with this, use the following:
    - Make each function do one thing and one thing only.
    - No more than 15 lines inside the curly braces of any function, including main(). Whitespace, variable declarations, single curly braces, vertical spacing, comments, and function headers do not count.
    - Functions over 15 lines need justification in comments.
    - Do not put multiple statements into one line.
- You are encouraged to use the functions in assignment 2 to do error handling.

- You may not use libraries i.e. <algorithm>
- You must not have any memory leaks
- You program should not have any runtime errors, e.g. segmentation fault
- No global variables allowed (those declared outside of many or any other function, global constants are allowed).
- Make sure you follow the style guidelines, have a program header and function headers with appropriate comments, and be consistent.

In addition to above requirements, implementation Requirements for Part B and Part D

- Use of references or pointers is required.
- Use of C-style string is required
- Use of 1D dynamic array is required
- Libraries allowed to use: <iostream>, <string>, <cstring>, <cstdlib>, <cmath>

Electronically submit your C++ program (.cpp file, not your executable!!!) by the code due date, on TEACH.

Remember to sign up with a TA on Canvas to demo your assignment. The deadline to demo this assignment without penalty is 03/12/2020.