COP 3014 Fall 2021
Homework 6

Total Points: 100 (and 50 points extra credit)
Due: Monday 11/22/2021 11:59:00 PM

1 Objective

The objective for this assignment is to make sure

- You understand and can work with strings in C++, both C-Strings and C++ string objects.
- You understand and can work with C++ arrays.
- You are comfortable with writing functions in C++.
- You are familiar with the concept of pointers
- You are familiar with repetitive structures (loops) and selection statements (if/else or switch) in any combination and can use them in functions and to manipulate array data.
- You can approach a complex problem, break it down into various parts, and put together a solution.

For this assignment, please make sure you conform to Output requirements. You should now be familiar enough with cout statements and output formatting that you should be able to EXACTLY match the sample output (other than certain exceptions you will be informed about beforehand).

You should also conform to the convention of declaring functions above `main` and defining them after `main`. And input/output statements should be confined to functions written for that purpose, and `main`.

This assignment requires you to submit multiple files on Canvas. Please do so in a SINGLE submission. Canvas allows you to turn in multiple flies in one submission. Once you have uploaded your first file, click "attach another file" to upload your second file, and so on.

Turn in your files `stringops.cpp, palindromes.cpp` and `stringPivots.cpp`, if you attempted it, through Canvas.

2 Problem 1 - String Operations - 50 points

Sheldon J. Plankton is on a mission. He is approaching people on the street and asking them to speak a sentence of their choice into a recorder. He is then going to generate statistics on the string. He wants to know the number of words, the number of punctuation characters, the number of uppercase characters immediately followed by lowercase characters, and to check if the string "end" is found in the string. However, he is completely out of his depth with string processing and has just hired you to write a C++ program for him.

This program has to use C++ string objects, and not C-strings.

Specifications

- Call this program `stringops.cpp`
- Write a function called `wordCount` that takes in a string as a parameter and returns the number of words in the string. (8 points)
- Write a function called `puncCount` that takes in a string as a parameter and returns the number of punctuation characters in the string. (9 points)
- Write a function called `upperLowerCount` that takes in a string as a parameter and returns the number of uppercase characters folowed immediately by a lowercase character in the string. (9 points)
- Write a function called `findSubString` that takes in a string as a parameter and returns 1 if the string "end" is a part of the string and 0 if it is not. (9 points)
- In the main function, accept a newline terminated string from the user. Then, print all the 4 statistics of the string. Repeat until the user enters "Done". (10 points)
- Please comment your code appropriately. (5 points)
- You are restricted to the `iostream, cctype` and `string` libraries for this program.
- You are not allowed to use the string class iterators like `begin, end, rbegin, rend,` etc.
- You may assume that the text entered will not contain stray whitespace. That is, if a space character is in the string, it will only be a single space between 2 words.
- You are not allowed any secondary/temporary strings. The checks have to be done with the C++ string object variable/parameter, without storing/copying the strings in temporary strings.

Sample Run

Regular text is what's printed by your program. Underlined text is user input, shown here as a sample. You will not be printing the underlined parts in your program.

```
Enter the string:  Maybe a story will cheer you up:  Once upon a time, there was an
Ugly Barnacle.  It was so ugly, that everyone died.  The end.
Number of words:  25
Number of punctuations:  6
Number of uppercase-lowercase character sets:  6
"end" is a part of this string.

Enter the string:  I have a square head and a real ghost has a round one.  All we
have to do is make my head round and boo, I'm scary
Number of words:  27
Number of punctuations:  3
Number of uppercase-lowercase character sets:  1
"end" is not a part of this string.

Enter the string:  Done
```

3 Problem 2 - Palindrome Strings - 50 points

Plankton has asked you to email your findings to him in the form of a report. However, he has a different style of communication. He likes to read his words backwards. However, you can only read text forwards. So, you decide to write up the report as usual, but you try to only use sentences or words that read the same forwards and backwards, and then send it through a program to reverse the words for you.

Write a C++ program that reads in several sentences terminated by newlines. Stop when the user enters a blank line. Then examine each string and check if it reads the same forwards and backwards (ignoring spaces, punctuation, and case), and then check if the string contains any individual words that read the same forwards and backwards even if the entire string doesn't.

For this program, you care required to use an array of C-Strings.

- Call this program `palindromes.cpp`
- Write a function called `checkPalin`. This function takes 1 c-string as a parameter and returns a boolean. It returns `true` if the entire c-string (ignoring spaces, punctuation and case) is a palindrome and `false` otherwise. (13 points)
- Write a function called `palWords`. This function takes 1 c-string as a parameter and returns an integer. It returns the number of words in the c-string that are palindromes, once again, ignoring spaces, punctuation and case. (15 points)
- In the main function, declare an array of 20 c-strings. Each c-string in the array could be 100 characters maximum. (3 points)
- Read in the newline terminated c-strings from the user. Stop when they enter a blank line. You may assume that the user would never enter more than 20 c-strings and that each c-string would not contain more than 99 characters. (7 points)
- For each c-string, call the above functions to determine if the entire string is a palindrome and if the c-string contains any words that might be palindromes by themselves. Print this information. (7 points)
- Please comment your code appropriately. (5 points)
- You are restricted to the `iostream, cctype` and `cstring` libraries.
- You are not allowed any secondary/temporary arrays/c-strings. The palindrome checks have to be done in-place, without storing/copying the c-strings in temporary c-strings.
- You may assume that your input would only consist on letters, numerals, punctuations and spaces.

Sample Run

Regular text is what's printed by your program. Underlined text is user input, shown here as a sample. You will not be printing the underlined parts in your program.

```
Enter the strings:
Swap, Paws
Rats Live on no evil Star!
Hannah likes the Racecar
```

```
20 Radar Kayak radar 02
A Toyota's a Toyota
Solos Deified Rotator Sagas
Is mayonnaise an instrument?

Palindrome strings:
Swap, Paws
Rats Live on no evil Star!
20 Radar Kayak radar 02
A Toyota's a Toyota

Swap, Paws - 0 palindrome words
Rats Live on no evil Star!  - 0 palindrome words
Hannah likes the Racecar - 2 palindrome words
20 Radar Kayak radar 02 - 3 palindrome words
A Toyota's a Toyota - 2 palindrome words
Solos Deified Rotator Sagas - 4 palindrome words
Is mayonnaise an instrument?  - 0 palindrome words
```

4 Extra Credit Problem - Pivoted Strings - 50 points

We get a pivoted string if we hold a mirror to the the top of a string and then reversing the string. For this program, the "pivotable" characters given in the table below are pivoted in the output. The rest are left as is.

Write C++ program where you read in multiple c-strings. As you read in each string, pviot the string and then accumulate the string into one C++ string object.

Specifications

- Call this program `stringPivots.cpp`
- Write a function called `readAndPviot`. This function takes no parameters and returns a C++ string object. (3 points)
- In the function:
  - Read in the number of c-strings from the user. (2 points)
  - Read in the newline-terminated c-strings. You may assume the c-strings will not be longer than 150 characters. (5 points)
  - For each c-string, reverse the c-string, and flip the pivotable characters as shown in the table below. (20 points)
  - As each c-string is pivoted, accumulate the results into a C++ string object. (5 points)
  - Return the C++ string object. (2 points)
- In the main function, call the function and print the returned C++ string object. (5 points)
- The main function should not contain more than 3 lines of code, including the return statement. (3 points)

- Please comment your code appropriately. (5 points)
- You are restricted to the `iostream, string` and `cstring` libraries for this program.
- You may assume that the text entered will not contain stray whitespace. That is, if a space character is in the string, it will only be a single space between 2 words.
- You are not allowed to use the string class iterators like `begin, end, rbegin, rend,` etc.
- You are not allowed any secondary/temporary arrays. You have 1 c-string for the input and processing and 1 C++ string object to accumulate the overall result. The reversing and pivoting has to be done in-place, without storing/copying the strings in temporary arrays.

Character	Pivoted character
M	W
W	M
b	q
d	p
h	y
m	w
n	u
p	d
q	b
u	n
w	m
y	h

Sample Run

```
Enter the number of strings: 3
Enter the strings:
No Patrick! Mayonnaise is not an instrument
The inner machinations of my mind are an enigma
Can I be excused for the rest of my life?

The mirrored strings are:
tuewnrtsui ua tou si esiauuohaW !kcirtaP oN
awgiue ua era puiw hw fo suoitauiycaw reuui eyT
?efil hw fo tser eyt rof pesncxe eq I uaC
```

5 Generic Guidelines

1. Include the header comment with your name and other information on the top of your files.
2. Please make sure you're only using the concepts already discussed in class. Please restrict yourself to input/output statements, variables and operators, selection statements, loops, functions, arrays, and strings. Using pointers, structs, generic classes, iterators or anything more advanced will result in a loss of 25 points.

3. If we have listed a specification and allocated point for it, you will lose points if that particular item is missing from your code, even if it is trivial.
4. No global variables (variables outside of main())
5. No use of the `auto keyword` or any other C++ 11 or higher features.
6. Functions have to be declared above main and defined below main. Not doing so will result in a loss of 10 points.
7. You need to ensure you do not go out of bounds in the arrays, even if CLion protects your program from crashing.
8. You need to follow the other programming conventions established in the course. This includes not using break or continue statements, unless the break statement is used to avoid the fall-through in a switch statement.
9. **This is individual work. You may NOT collaborate with other students in the course, former students, hire tutors to "help", copy solutions off the internet, or use pay-for solution websites, including but not limited to Chegg, CourseHero, WiseAnt, Bartelby, tutor.com, assorted Social Media groups, whatever GroupMe students might have created, etc.) This includes posting the problem statements on these websites even if you do not use the answer obtained. Doing so is a violation of the Academic Honor Code. Violation of the Honor Code will result in a 0 grade on the program, a reduced letter grade in the course and potentially more serious consequences.**
10. All input and output must be done with streams, using the library `iostream`
11. You may only use the libraries specified for each program (you do not need any others for these tasks)
12. Please make sure that you're conforming to specifications (program name, function names and parameters, print statements, expected inputs and outputs etc.). Not doing so will result in a loss of points
13. NO C style printing is permitted. (Aka, don't use printf). Use cout if you need to print to the screen.
14. When you write source code, it should be readable and well-documented (comments).
15. Make sure you either develop with or test with Jetbrains CLion (to be sure it reports no compile errors or warnings) before you submit the program.
16. Testing your program thoroughly is a part of writing good code. We give you sample runs to make sure you match our output requirements and to get a general idea of how we would test your code. Matching your outputs for JUST the sample runs is not a guarantee of a 100. We have several extensive test cases.
17. Please make sure you've compiled and run your program before you turn it in. Compilation errors can be quite costly. We take 5 points off per compiler error for the first 9 errors. The 10th compiler error will result in a grade of 0.
18. Only a file turned in through Canvas counts as a submission. A file on you computer, even if it hasn't been edited after the deadline, does not count.
19. The student is responsible for making sure they have turned in the right file(s). We will not accept any excuses about inadvertently modifying or deleting files, or turning in the wrong files.

20. **Program submissions** should be done through the Canvas class page, under the assignments tab (if it's not there yet I'll create it soon.) Do not send program submissions through e-mail – e-mail attachments will not be accepted as valid submissions.

21. The ONLY files you will submit via Canvas are `stringops.cpp, palindromes.cpp` and `stringPivots.cpp`.

22. **General Advice** - always keep an untouched copy of your finished homework files in your email. These files will have a time-stamp which will show when they were last worked on and will serve as a backup in case you ever have legitimate problems with submitting files through Canvas. Do this for ALL programs.