CMPT 130 - FIC 2018-03 - Assignment 2

Due Date: Monday 26th November 2018 at 11:55PM

Instructor: Dr. Yonas T. Weldeselassie (Ph.D.)

Read this document in its entirety and carefully before you start anything and understand it. If you have any questions, don't hesitate to email me.

In this assignment, we will work with c-strings. In order to store c-strings, we will use C++ static or dynamic arrays of characters. **You are not allowed to use the C++ string data type in this assignment. Read the restriction section below.**

Consider the following main program together with function headers given below:

```
#include <iostream>

using namespace std;

typedef char* charPointer;

int stringLength(const charPointer s)
{
	//returns the number of printable characters in s
}

int countChars(const charPointer s, const char ch)
{
	//returns the number of times the character ch is found in s
}

int findChar(const charPointer s, const char ch, const int startIndex, const int lastIndex)
{
	/*
	returns the first index where the character ch is found in s starting from startIndex
	(inclusive) upto lastIndex (exclusive)
	If ch is not found in s in the interval, it returns -1
	This function must first validate both the startIndex and lastIndex.
	That is, if lastIndex > stringLength(s) or startIndex < 0 it must return -1
	*/
}

void rotateString(charPointer s, const int r)
{
	/*
	Rotate the characters of s by r units
	If r > 0, rotate the characters of s to the left
	If r < 0, rotate the characters of s to the right
	Please note the value of r can be any integer even larger than the length of s
	For example,
		"asmara" rotated to the left by 0 becomes "asmara"
		"asmara" rotated to the left by 1 becomes "smaraa"
		"asmara" rotated to the left by 2 becomes "maraas"
		"asmara" rotated to the left by 3 becomes "araasm"
		"asmara" rotated to the left by 4 becomes "raasma"
		"asmara" rotated to the left by 5 becomes "aasmar"
		"asmara" rotated to the left by 6 becomes "asmara"
		"asmara" rotated to the left by 7 becomes "smaraa"
		"asmara" rotated to the left by 8 becomes "maraas"
		and etc…
```

```
	*/
}
void append(charPointer &s, const char ch)
{
	/*
	Appends the character ch to the c-string s.
	That is ch is added to the end of s
	The parameter s is assumed to be a dynamic array (NOT a static one)
	*/
}

void append(charPointer &s1, const charPointer s2)
{
	/*
	Appends all the characters of s2 to s1
	The parameter s1 is assumed to be a dynamic array (NOT a static one)
	*/
}

void removeAll(charPointer &s, const char ch)
{
	/*
	remove all the occurences of the character ch from the c-string s
	The parameter s is assumed to be a dynamic array (NOT a static one)
	*/
}

charPointer zigzagMerge(const charPointer s1, const charPointer s2)
{
	/*
	create and return a new c-string by merging (putting in one) s1 and s2 in zigzag form.
	That is
			first character of the new c-string is the first character of s1
			second character of the new c-string is the first character of s2
			third character of the new c-string is the second character of s1
			fourth character of the new c-string is the second character of s2
			fifth character of the new c-string is the third character of s1
			sixth character of the new c-string is the third character of s2
			etc
		When either s1 or s2 reaches to its end, the remaining characters of the other are
		appended to the new c-string
		Example, zigzagMerge of "abc" and "defgh" will be "adbecfgh"
	*/
}

bool isAnagram(const charPointer s1, const charPointer s2)
{
	/*
	returns true if s1 and s2 contain same distinct characters appearing same number of times in
	both s1 and s2; otherwise returns false
	That is, this function returns true if s1 and s2 are permutations (re-arrangements) of each
	other.
	For example "CMPT" and "PCTM" are anagrams but "CPMT" and "CMPTM" are not anagrams
	*/
}

bool isSubString(const charPointer s1, const charPointer s2)
{
	/*
	returns true is s1 is a substring of s2 otherwise returns false
	Definition: s1 is a substring of s2 if s1 is found in s2.
```

```
	That is all characters of s1 are found TOGETHER in s2 in the SAME ORDER as they appear in s1
	Example "set" is a substring of "massachussettes" But "ets" is not substring of
	"massachussettes"
	*/
}

int countWords(const charPointer s)
{
	/*
		Given a c-string that contains some words separated by spaces,
		return the number of words in the c-string.
		In this case, a word means some characters with no space in between.
		Example: If the c-string parameter is "What    a     nice           problem".
		Then you see that there are FOUR words in this c-string, namely
		     1. What      2. a       3. nice      4. problem
		Your function then must return 4

		For simplicity,
		1. Assume that there are no spaces at the beginning or at the end of the c-string
		2. But a word can be separated from another word by one or more space
		2. Assume the parameter does not contain any punctuation marks such
		   as comma, semicolon or full stop.
	*/
}

int main()
{
	/*
	This main program is designed to test the functions you need to implement.
	You should NOT remove any line of code from this main program.
	But you may add more test code in the main program if you like.
	*/

	//Test stringLength function
	cout << endl;
	char s1[] = "massachussettes";
	cout << s1 << " has " << stringLength(s1) << " characters" << endl;

	//Test countChars function
	cout << endl;
	char ch = 's';
	int c = countChars(s1, ch);
	cout << ch << " appears " << c << " times in " << s1 << endl;

	//Test findChar function
	cout << endl;
	int index = findChar(s1, ch, 10, 14);
	if (index == -1)
		cout << ch << " is not found in " << s1 << " between indexes [10, 14)" << endl;
	else
		cout << ch << " is found at index " << index << " in " << s1 << " between indexes [10,
14)" << endl;

	//Test rotateString function
	cout << endl;
	char temp1[] = "massachussettes";
	rotateString(temp1, 2);
	cout << s1 << " rotated 2 units to the left becomes " << temp1 << endl;
	char temp2[] = "massachussettes";
	rotateString(temp2, -19);
	cout << s1 << " rotated 19 units to the right becomes " << temp2 << endl;
```

```
//Test append function (appending a character to c-string)
cout << endl;
charPointer s2 = new char[1];
s2[0] = '\0';
cout << "Given the c-string " << s2 << endl;
ch = 'a';
append(s2, ch);
cout << "\tAppending " << ch << " results to " << s2 << endl;
ch = 'b';
append(s2, ch);
cout << "\tAppending " << ch << " results to " << s2 << endl;

//Test append function (appending a c-string to a c-string)
cout << endl;
cout << "Appending " << s1 << " to " << s2 << ", we get ";
append(s2, s1);
cout << s2 << endl;

//Test removeAll function
cout << endl;
ch = 'e';
cout << "Removing all occurences of " << ch << " from " << s2 << ", we get ";
removeAll(s2, ch);
cout << s2 << endl;
ch = 't';
cout << "Removing all occurences of " << ch << " from " << s2 << ", we get ";
removeAll(s2, ch);
cout << s2 << endl;
ch = 's';
cout << "Removing all occurences of " << ch << " from " << s2 << ", we get ";
removeAll(s2, ch);
cout << s2 << endl;

//Test zigzagMerge function
cout << endl;
charPointer s3 = zigzagMerge(s1, s2);
cout << "The zigzag merge of " << s1 << " and " << s2 << " is " << s3 << endl;

//Test isAnagram function
cout << endl;
char s4[] = "htsemsaesuatscs";
bool flag = isAnagram(s1, s4);
if (flag)
	cout << s1 << " and " << s4 << " are anagrams" << endl;
else
	cout << s1 << " and " << s4 << " are not anagrams" << endl;

//Test isSubString function
cout << endl;
flag = isSubString(s1, s4);
if (flag)
	cout << s1 << " is a substring of " << s4 << endl;
else
	cout << s1 << " is not a substring of " << s4 << endl;

char s5[] = "abort";
char s6[] = "abcabodefaborhuhabortabunny";
flag = isSubString(s5, s6);
if (flag)
	cout << s5 << " is a substring of " << s6 << endl;
```

```
        else
                cout << s5 << " is not a substring of " << s6 << endl;

        //Test countWords function
        cout << endl;
        char s7[] = "";
        c = countWords(s7);
        cout << "There are " << c << " words in " << s7 << endl;
        char s8[] = "Test";
        c = countWords(s8);
        cout << "There are " << c << " words in " << s8 << endl;
        char s9[] = "Nice       one";
        c = countWords(s9);
        cout << "There are " << c << " words in " << s9 << endl;
        char s10[] = "This      is a nice     assignment and    hopefully an interesting    as well";
        c = countWords(s10);
        cout << "There are " << c << " words in " << s10 << endl;

        //Delete dynamically allocated memories
        delete[] s2;
        delete[] s3;

        system("Pause");
        return 0;
}
```

Your Task

You are required to copy the given main program together with the function headers exactly as they are with no change **whatsoever** and then implement the functions so that the program works correctly. Each function is described in detail in the comments given together with the function header. Some sample runs are given below for your reference. Of course you can write additional helper functions that you can call from within your functions; but you are not allowed to make any change to the given main program and function headers. Once you are done with your assignment, you can add more code to the main program if you would like to test your functions further. Of course I will only mark your functions; the main program is provided mainly to help you test your functions.

Restrictions

You are required to use C++ static or dynamic arrays of characters to store c-strings. You are NOT allowed to use any C++ string data type variable for any purpose. If you use a C++ string data type variable for any purpose in your program, you will automatically get zero mark. Moreover, you are NOT allowed to add any include directive. **Specifically you are not allowed to include string, cstdlib or math libraries. Also you are not allowed to use any built-in functions of c-strings.**

Submission Format

You are required to submit your program online through Moodle. You will find a submission button for **Assignment 2** on **Moodle on Week 12** and you are required to upload the source code of your C++ program. No assignment is submitted through email or hard copy; you must upload your work onto Moodle before the due date. Make sure to upload a CPP file.

Submission Due Date

The due date to upload your program online through Moodle is **Monday 26th November 2018 at 11:55PM**. Moodle will not allow you to upload after this date and time.

Marking

A non working program will automatically get zero. A program that works but doesn't give right output or gives partial right output will lose marks depending how severe its shortcoming is. This assignment carries 6% of the total course marks.

Sample Run of the program

massachussettes has 15 characters

s appears 5 times in massachussettes

s is not found in massachussettes between indexes [10, 14)

massachussettes rotated 2 units to the left becomes ssachussettesma
massachussettes rotated 19 units to the right becomes ttesmassachusse

Given the c-string
    Appending a results to a
    Appending b results to ab

Appending massachussettes to ab, we get abmassachussettes

Removing all occurences of e from abmassachussettes, we get abmassachusstts
Removing all occurences of t from abmassachusstts, we get abmassachusss
Removing all occurences of s from abmassachusss, we get abmaachu

The zigzag merge of massachussettes and abmaachu is maabsmsaaacchhuussettes

massachussettes and htsemsaesuatscs are anagrams

massachussettes is not a substring of htsemsaesuatscs
abort is a substring of abcabodefaborhuhabortabunny

There are 0 words in
There are 1 words in Test
There are 2 words in Nice one
There are 11 words in This is a nice assignment and hopefully an interesting as well