

CS 1004 Object Oriented Programming

Assignment

Number 1

February 6, 2023

Deadline: Monday 15th February, 2023 before 23h00

Attention

Make sure that you read and understand each and every instruction. If you have any questions or comments you are encouraged to discuss your problems with your colleagues (and instructors) on google classroom.

Plagiarism is strongly forbidden and will be very strongly punished. If we find that you have copied from someone else or someone else has copied from you (with or without your knowledge) both of you will be punished. You will be awarded straight zero in this assignment or all assignments.

Submit a single '.zip' file for your assignment and each problem solution must be provided in a separate CPP file. For instance, you must name the file containing solution of first file as 'q1.cpp' and second as 'q2.cpp' and so on.

For all the problems you are required to use "char *", usage of string is strictly prohibited.

Please start early otherwise you will struggle with it.

Note: You have to follow the submission instructions to the letter. Failing to do so can get a zero in assignment.

Q1) Password Verifier: Imagine you are developing a software package that requires users to enter their own passwords. Your software requires that users' passwords meet the following criteria:

The password should be at least six characters long.

The password should contain at least one uppercase and at least one lowercase letter.

The password should have at least one digit.

Write a function that accepts a pointer to a C-string as its argument for a password and then verifies that it meets the stated criteria.

If it doesn't, the program should display a message telling the user why.

Q2) Magic Squares: In this question your goal is to write code for solving magic square (for details read the attached file magic-square.pdf) using 2-D pointers. Your program should ask the user for the dimension of magic square and then solve the magic square for given dimension. You must do the proper allocation and deallocation of the memory.

Q3) Connect4: Write a two-player game of "connect 4" where the user can set the width and height of the board and each player gets a turn to drop a token into the slot. Display the board using + for one side, x for the other, and _ to indicate blank spaces. You have to write a function that accepts a 2D pointer to a char array that represents the board. For further information on Connect4: https://en.wikipedia.org/wiki/Connect_Four. You will play the game till the board gets filled or either one of the player wins.

Q4: String manipulation functions Write the implementation of the following functions

```
1  int Strlen(char *s1)
2  /*returns the length of string in number of characters...*/
3  {
4  }
5  char *Strcpy( char *s1, const char *s2 )
6  /*Copies string s2 into array s1.
7  The value of s1 is returned.
8  char *strncpy( char *s1, const char *s2, size_t n )
9  Copies at most n characters of string s2 into array s1.
10 The value of s1 is returned.*/
11 {
12
13 }
```

```

1  char *StrCat( char *s1, const char *s2 )
2  /*Appends string s2 to array s1.
3  The first character of s2 overwrites
4  the terminating null character of s1.
5  The value of s1 is returned.*/
6  {
7
8  }
9
10 char *StrnCAt( char *s1, const char *s2, size_t n )
11 /*Appends at most n characters of string s2 to array s1.
12 The first character of s2 overwrites the terminating
13 null character of s1. The
14 value of s1 is returned.*/
15 {
16 }
17
18
19 int StrCmp( const char *s1, const char *s2 )
20 /*Compares the string s1 with the string s2 .
21 The function returns 0,
22 less than 0 or greater than 0 if s1 is equal to,
23 less than or greater
24 than s2 , respectively.*/
25 int StrnCmp( const char *s1, const char *s2, size_t n )
26 /*
27 Compares up to n characters of the string
28 s1 with the string s2 .
29 The function returns 0 ,
30 less than 0 or greater than 0 if s1 is equal
31 to, less than or greater than s2 , respectively.
32 */
33 {
34 }
35 char **StrTok( char *s1, const char s2)
36 /*A call to StrTok breaks string s1 into
37 ``tokens'' (logical pieces such as words
38 in a line of text) separated by character
39 contained in char s2*/
40 {}
41
42 int StrFind(char *s1, char *s2)
43 /*
44 Searches the string s1 for the first occurrence
45 of the string s2 and returns its starting index,
46 if s2 not found returns -1.
47 */
48 {}
49
50 char * SubStr (char *, int pos, int len)
51 /*
52 This function returns a newly constructed
53 string variable with its value initialized
54 to a copy of a substring of this variable.
55
56 The substring is the portion of the string
57 that starts at character position ``pos'' and
58 spans ``len'' characters (or until the end
59 of the string, whichever comes first).*/
60 {}

```

Q5: String Text Editor

Here your goal is to build a string text editor, which will allow its user to create simple text files and store them in primary memory (Remember you will be storing data in primary memory RAM not on hard disk, so you will not need file handling). Your text editor will have following main menu:

```
Press 1. To create a new file.
Press 2. To view an existing file via giving a file name.
Press 3. To edit an existing file via giving its name.
Press 4. To copy an existing file to a new file.
Press 5. To delete an existing via its name.
Press 6. To view list of all files with the names.
Press 7. To Exit.
```

File Creation Whenever user will press 1. your program will ask for the new file name and number of text lines in the file. You will also tell the user that in any single line of file there can be maximum of 60 characters, if there are more than 60 characters in a single line the remaining characters should be moved to next line. While taking input from user your program should automatically display each line number on the terminal, you will be using this number during file editing for updating a particular line. For example, this is how your program output will editing a file.

```
1. I am writing a text file.
2. This is second line of my file and i am storing data
3. on line by line basis.
```

File Editing User should be able to edit a file by giving its name. When editing a file user can upgrade any single line by giving its number and your program will update the content of that line with the newly provided text by user. Furthermore, use can ask to replace a specific word, e.g . user can ask to replace all the instances of word “work” with “enjoy”. Your program should find all the instances of word “work” and should replace it with “enjoy”.

File Copying Your program will allow its user to create a copy of an existing file and he should be asked to provide a valid new file name.

File Deleting User should be allowed to delete a file via its name.

For completing this task you will need to make use of dynamic memory and multi-dimensional pointers. So please start early otherwise you will struggle with it.

Q6:Text Analysis The availability of computers with string-manipulation capabilities has resulted in some rather interesting approaches to analyzing the writings of great authors. This exercise examines three methods for analyzing texts with a computer. **You have to use char * for following exercises.**

1. Write a function that receives a string consisting of several lines of text and returns an array indicating the number of occurrences of each letter of the alphabet in the text. For example, the phrase “To be, or not to be: that is the question”: contains one “a,” two “b’s,” no “c’s,” and so on.

```
1 void countLetters(char *string, int *&array, int & size)
2 /*Parameters:
3 Input:
4 char * : a multiline string
5 Output:
6 int *: an array containing counts of each letter,
7 to be allocated in function
8 int : array size */
9 ;
```

2. Write a function that receives a string consisting of several lines of text and returns an array indicating the number of one-letter words, two-letter words, three-letter words, and so on, appearing in the text. For example, the phrase “Whether this nobler in the mind to suffer” contains 2, 3, 4, etc. length words.

```

1 void countWordsBasedOnLength(char *string, int *&array/*to be allocated */,
2 int & size/*updated array size*/)
3 /*Parameters:
4 Input:
5 char * : a multiline string
6 Output:
7 int *: an array containing counts of each different length words,
8 to be allocated in function
9 int : array size */
10 ;

```

3. Write a function that receives a string consisting of several lines of text and returns arrays indicating unique words and the number of occurrences of each unique word in the text along with their size.

```

1 void countingUniqueWords(char *string, char **&uwords/*list of unique words;*/,
2 int *&array/*to be allocated */, int & size/*updated array size*/)
3 /*Parameters:
4 Input:
5 char * : a multiline string
6 Output:
7 char **: an array of unique words
8 int *: their counts
9 int : number of unique words*/
10 ;

```