NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES
ISLAMABAD CAMPUS
COMPUTER PROGRAMMIN (CS103) - FALL 2018
ASSIGNMENT-1

Due Date: September 17, 2018 (05:00 pm)

Instructions:

1. *Make sure that you read and understand each and every instruction.*
2. *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.*
3. *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.*
4. *For all the problems you are required to user \char *", usage of string is strictly prohibited.*
5. *Please start early otherwise you will struggle with it.*
6. *You have to use pointer notation for all the questions. No subscript notation is allowed as it will result in a zero in the question.*
7. *You have to do proper allocation and deallocation of the memory.*
8. *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) **Phone Number List:** Write a program that has an array of at least 10 string objects that hold people names and phone numbers. You may make up your own strings, or use the following:

"Becky Warren, 555-1223"
"Joe Looney, 555-0097"
"Geri Palmer, 555-8787"
"Lynn Presnell, 555-1212"
"Holly Gaddis, 555-8878"
"Sam Wiggins, 555-0998"
"Bob Kain, 555-8712"
"Tim Haynes, 555-7676"
"Warren Gaddis, 555-9037"
"Jean James, 555-4939"
"Ron Palmer, 555-2783"

The program should ask the user to enter a name or partial name to search for in the array. Any entries in the array that match the string entered should be displayed. For example, if the user enters Palmer the program should display the following names from the list:

Geri Palmer, 555-8787
Ron Palmer, 555-2783

Write a function **searchDirectory (char **& directory, string name)** that identifies the name in the directory.

Q5) **Mode:** In statistics, the *mode* of a set of values is the value that occurs most often or with the greatest frequency. Write a function that accepts as arguments the following:

A) A pointer to an array of integers
B) An integer that indicates the number of elements in the array

The function should determine the mode of the array. That is, it should determine which value in the array occurs most often. The mode is the value the function should return. If the array has no mode (none of the values occur more than once), the function should return -1. (Assume the array will always contain nonnegative values.)

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.

I. 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.
   void countLetters(char *string, int *&array, int & size)

II. 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.
void countingUniqueWords(char *string, char **&uwords/*list of unique words;*/, int *&array/*to be allocated */, int & size/*updated array size*/)

III. Write a function that accepts a pointer to a C-string as its argument and returns the number of words contained in the string. For instance, if the string argument is “Four score and seven years ago” the function should return the number 6.
void countWords(char *string, int &count /*count of words;*/,)

IV. Write a program that accepts as input a sentence in which all of the words are run together, but the first character of each word is uppercase. Convert the sentence to a string in which the words are separated by spaces and only the first word starts with an uppercase letter. For example the string “StopAndSmellTheRoses”. would be converted to “Stop and smell the roses”.
void wordSeperator(const char *string, char *&newString/*converted String;*/,)

Q7) **String manipulation functions:** Write the implementation of the following functions

- **int Strlen(char *s1)**: Returns the length of string in number of characters
- **char *StrCat(char *s1, const char *s2):** Appends string s2 to array s1
- **char *StrCat(char *s1, int n, char z):** Appends n copies of z to array s1
- **void upper(char *&s1):** Step through each character and convert it to upper case
- **void *lower(char *&s1):** Step through each character and convert it to lower case
- **void *reverse(char *&s1):** Steps through the string, it should test each character to determine whether it is upper- or lowercase. If a character is uppercase, it should be converted to lowercase. Likewise, if a character is lowercase, it should be converted to uppercase.
- **char* strstr(char* s1, char* s2):** Searches for the first occurrence of s2 in s1. If an occurrence of s2 is found, the function returns a pointer to it. Otherwise, it returns a NULL pointer (address 0).
- **void strReplace(char *&s1, const char* tobeReplaced, const char * replaceWith):** Replace the tobeReplaced in s1 with the replaceWith string.
- **int StrCmp(const char *s1, const char *s2):** Compares the string s1 with the string s2 .The function returns 0, less than 0 or greater than 0 if s1 is equal to s2 less than or greater than s2 , respectively.
- **char **StrTok(char *s1, const char s2):** A call to StrTok breaks string s1 into ``tokens’’ (logical pieces such as words in a line of text) separated by character contained in char s2

Q8) **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 exisiting file via giving its name.
Press 4. To copy an exisiting file to a new file.
Press 5. To delete an exisiting 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 “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.