Programming Fundamentals COURSE & LAB – Spring 2022 (BS-IT-F21 Morning)

Note: For Character Arrays, you must not use size in loop, you must loop until the Null (‘\0’) character is found.

Task # 1:

Write a function replace(char str[], char c1, char c2) which takes 3 parameters: a character array, a character to find and a character to replace. Your function needs to find all occurrences of a character (taken as the 2nd parameter) and replace it with another character (taken as 3rd parameter) and display it at the end.

Example:

```
char str[] = “This_is_a_line”
char c1 = ‘_’
char c2 = ‘ ’      // This is a space character (spacebar)
replace(str, c1, c2)
```

Output: This is a line

Task # 2:

Write a function count(char str[]) to find the number of alphabets, digits and in a given string.

Loop over the character array and check if each character is an alphabet or digit.
A character c is an alphabet if it is **c >= ‘a’ and c <= ’z’**.
A character c is digit if it is **c >= ‘1’ and c <= ’9’**

Example:
“BITF21M500” has 5 alphabets and 5 digits
“Something” has 9 alphabets and 0 digits
“1234” has 0 alphabets and 4 digits

Task # 3:

Implement the functionality of caesar cipher in your program.

Caesar cipher is a cipher algorithm that switches characters in a string with some other character after it in the alphabet based on the given key.

For example if we have a string “**abcde**” and a **key of 3** then what will happen is
a will become **d**
b will become **e**
c will become **f**
d will become **g**
e will become **h**
Finally the string will become “**defgh**”.

A problem arise if we go out of bounds of alphabet

For example “**wxyz**” with a **key of 2**

w becomes **y**
x becomes **z**

What about y and z? Adding 2 to them is greater than 26 and out of alphabets. In this case, we loop around the alphabet.

y becomes **a** (y + 1 becomes z, and z + 1 is a)
z becomes **b** (z + 1 becomes a, and a + 1 is b)

To always achieve accurate results which loops around, you can use the formula below

c = (p + k) % 26
 where p is the original character, k is the key and c is the new character. The % by 26, ensures that the new character is within the alphabet range.

You can also watch this short video for better understanding of the algorithm:
https://youtu.be/o6TPx1Co_wg

Before Doing tasks 4 and 5 read the notes at the end. It's very helpful.

Task # 4:

Given a matrix, you are required to take the transpose of it and store it in the same matrix. Implement the function with the following header:

-> void transpose(int[][COL], int, int); //a matrix, number of rows, number of columns

Where rows and cols values should be less than or equal to 10 and greater than 0. Display an appropriate message if any of the values are out of bounds. Display the matrix in this function before returning. Be careful; the order of the resulting matrix may be different but surely less than or equal to 10x10.

Task # 5:

Given a matrix A of order MxN of maximum size 10x10 and strictly no more. And another matrix B of order PxQ. And two integers I and J. Where P+I should be strictly less than M and Q+J should be strictly less than N. If the conditions are not satisfied display an error message and return.

You are required to insert matrix B into matrix A starting at index I and J.

Function header:

***->** void insert(int[][N], int[][Q], const int, const int, const int, const int, const int, const int);*

The arguments list are in the following order: M, N, P, Q, I, J.

$$\begin{bmatrix} 0 & 1 & 5 & 3 & 4 \\ 5 & 3 & 7 & 8 & 3 \\ 8 & 1 & 7 & 6 & 5 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix}$$

MxN = 4x5

$$\begin{bmatrix} 9 & 999 & 9 \\ 9 & 999 & 9 \end{bmatrix}$$

PxQ = 2x3

After inserting PxQ ordered matrix at I = 2 and J = 1

$$\begin{bmatrix} 0 & 1 & 5 & 3 & 4 \\ 5 & 3 & 7 & 8 & 3 \\ 8 & 9 & 999 & 9 & 5 \\ 4 & 9 & 999 & 9 & 0 \end{bmatrix}$$

MxN = 4x5

Note for task 4 and 5:

- Best approach for this is to declare the matrix of size 10x10 at start. And then take input the number of rows (let's say x) and the number of columns (let's say y). Validate the x and y to be less than or equal to 10. And initialize only the x rows and y columns of the matrix by taking input from the user. Then work with the initialized part only. The rest of the rows and columns (memory) will not be used.
- Make a display function to display the 2d array in the form of a matrix. And call that function every time the 2d array is updated. This will help recognize the problems in the code.
- Here's an example: the MxN matrix will actually be like the following. Where n is any garbage value. You only need to work with the first 4 rows and 5 columns of it.

```
[[0 1 5 3 4 n n n n]
 [5 3 7 8 3 n n n n n]
 [8 1 7 6 5 n n n n n]
 [4 3 2 1 0 n n n n n]
 [n n n n n n n n n n]
 [n n n n n n n n n n]
 [n n n n n n n n n n]
 [n n n n n n n n n n]
 [n n n n n n n n n n]
 [n n n n n n n n n n]]
```

After taking a transpose of it.

```
[[0 5 8 4 4 n n n n]
 [1 3 1 3 3 n n n n n]
 [5 7 7 2 5 n n n n n]
 [3 8 6 1 0 n n n n n]
 [4 3 5 0 n n n n n n]
 [n n n n n n n n n n]
 [n n n n n n n n n n]
 [n n n n n n n n n n]
 [n n n n n n n n n n]
 [n n n n n n n n n n]]
```

Now the order is changed from 4x5 to 5x4 and you only have to display this submatrix. Note that the column 4 3 5 0 will remain unchanged but it is now not a part of the matrix and not required to take care of.