National University of Computer and Emerging Sciences, Lahore Campus

Object Oriented Programming

Assignment 1

Total Marks: To be Decided by TA

Section: BCS-9A **Due Date:** 26th June 2023

Instructions:

Please read the following instructions carefully before submitting the assignment.

1. Save your file with your Roll Number and Name
2. Submit your assignment in Google Classroom
3. It should be clear that your assignment will not get any credit if:
    - The assignment is submitted after the due date.
    - Assignment is copied(partial or full) from any source (websites, forums, students, etc.)

Important Note:

1. In exercises given below, subscript operator [] is not allowed to traverse the array. Use only offset notation. i.e instead of using myArray[i] use *(myArray+i) to read/write an element.
2. There shouldn't be any memory leakage in your program.
3. Make separate functions for input and output of arrays. Your main should be a sequence of function calls only
4. You are not allowed to use global variables and goto instruction
5. All allocations of 1D/2D pointers should be dynamic.
6. Delete the array when it is no longer needed.
7. All the data will be given by user.
8. Pass the pointers to functions instead of [].
9. Make proper functions to solve the problems.
10. Debug your code to find errors/bugs.

Task 6 [Expand Array]:

Write a program that keeps taking integer input from the user until user enters -1 and displays the data in reverse order.

Your program should save the input in a dynamically allocated array. Initially create a dynamic array of five integers. Each time the array gets filled your program should double the size of array and continue taking the input. After receiving -1 (i.e. end of data input) your program should print the numbers in the reverse order as entered by the user.

Task 2 [Compress Array]:
Write a program that takes size of an array and its elements and removes consecutive occurrences of same number from the list. For the example given below, your program should have space of exactly 7 integers on heap after compression. Do not consume any extra byte on heap.

Sample Run:

```
Array Before Compression: 1,1,2,2,2,3,4,5,5,5,5,5,7,7,7,2,2,2

Array After Compression: 1,2,3,4,5,7,2
```

Task 3 [Find Common Elements]:
Implement a function that finds common elements in two arrays. If array1 = {1,2,3,4,5,6,3,2} and array2 is {1,3,5,7}, then array3 (common elements) should be {1,3,5}. Note array3 should not have any duplicate elements and at the end, array3 should not consume a single extra byte on the heap. You have to:

- Allocate the three arrays dynamically after inputting the size of array1 and array2 from the user. Statically allocated arrays are NOT allowed
- Initially, you can allocate elements = (size of array1 + size of array2) to array3. For example, you would allocate 8+4 to array 3 for the above example. After finding the common elements, the allocated size of array3 may be more than what you need. (In the above example you require 3 whereas you have allocated 12).

Task 4 [Merge Array]:
Write a program that takes two sorted arrays, one in ascending order the other one in descending order, and merge the arrays in ascending order without using any sorting algorithm.

```
Array 1: 2,5,9
Array 2: 6,3,2,1

Merged Array: 1,2,2,3,5,6,9
```

Task 5 [2D Array]

Exercise 1: Write a function **int** AllocateMemory(int& rows, int& cols)** that takes size of matrix (rows and columns) from user, allocates memory for the matrix and return its pointer.

What is the advantage of sending the two parameters by reference?

Exercise 2: Write a function **void InitializeMatrix(int** matrix, const int& rows, const int& rows)** that initializes the matrix elements to 0. You may use subscript operator to initialize elements of matrix (only for this exercise).

Why are we passing the parameters as const?

Exercise 3: Write a function **void DisplayMatrix(int** matrix, const int& rows, const int& cols)** that displays the matrix in proper format.

Exercise 4: Write a function **void DeallocateMemory(int** matrix, const int& rows)** that deallocates all the memory.

Test your program. An example run is given below.

```
Enter total rows:4
Enter total columns:3
The array is:
0 0 0
0 0 0
0 0 0
0 0 0
```

Task 6:

You are given a 2D array having some elements as shown below. Your task is to remove all zero elements from the array by making a new 2D array and assign it only the non-zero elements. Assume that rows and columns of the input array are defined by the user.

2	3	1	0	0	0
0	0	0	1	1	0
1	0	0	0	0	0
1	1	1	2	0	2
5	0	0	0	10	0

Output Array:

2	3	1	
1	1		
1			
1	1	2	2
5	10		

Task 7:

Write a program that creates 2D dynamic array in main function where the columns in each row should be of different length and both values should be positive. Create a function **fillArray**. This function should receive the 2D array from main function and prompt the user to provide data. Your program should only accept positive values to fill the array. Decide the remaining parameters and return type of this function at your own. . Create a function **twoDimToOneDim**. This function should receive the 2D array from main function and creates a dynamic 1D array long enough to store the data of 2D array into this 1D array. This function should return the

address of dynamically created 1D array to main function. Decide the remaining parameters of this function at your own. Create a function **SortArr**. This function should receive the 1D array from main function and sort its data in ascending order. Decide the remaining parameters and return type at your own. Create a function **showArr**. This function should receive the sorted 1D array and display its contents of console. Make sure that this function should not update the contents of array.

Sample Output:
Enter the size of rows: 3
Enter the columns for row#0: 4
Enter the columns for row#1: 2
Enter the columns for row#2: 3

Assume that following data is stored in 2D array:

87	61	92	14
56	29		
5	78	45	

Contents of Sorted 1D array are: 5, 14, 29, 45, 56, 61, 78, 87, 92

Task 8:

Write a function that will take a string and return a count of each letter in the string. For example, "my dog ate my homework" contains 3 m's, 3 o's, 2 e's, 2 y's and one each of d, g, a, t, h, w, r and k. 2 Your function should take a single string argument and return a dynamically allocated array of 26 integers representing the count of each of the letters a . . z respectively. Your function should be case insensitive, i.e., count 'A' and 'a' as the occurrence of the letter a. [Hint: use the letter to integer conversion functions.] Do not count non-letter characters (i.e., spaces, punctuation, digits, etc.) Write a program that will solicit a string from the user using getline, call your letter frequency function and print out the frequency of each letter in the string. Do not list letters that do not occur at least once.
Example:
Enter a string:
my dog at my homework
Letter frequency
a 1
d 1
e 1
g 1
h 1
k 1
m 3
o 3
r 1
t 1
w 1
y 2