



**COMSATS University Islamabad, Lahore Campus**  
Block–C, Department of Computer Science  
COMSATS University Islamabad, Lahore Campus 1.5KM Defence Road, Off Raiwind Road, Lahore

## Terminal Lab exam (100 marks)

**Lab Examination – Semester Spring 2022**

Course Title:	Programming fundamentals			Course code:	CSC201	Credit hours	4(3,1)
Course Instructor:	Ms. Mahwish waqas		Program Name:	BS Software engineering			
Semester:	1st	Batch:	SP22-BSE	Section:	C	Date:	03/06/2022
Time Allowed:	3 hours			Maximum marks:	100		
Student Name:	Aoun-Haider			Reg no:	FA21-BSE-133		
<b>Important Instructions/Guidelines:</b> <ul style="list-style-type: none"><li>• Save your work frequently. In case of power failure, code will be your responsibility.</li><li>• Compile your program to make sure it at least compiles without errors. If your program will not compile that will be an automatic zero mark.</li><li>• Check your System properly and make sure that the G++ compiler works properly. Do not complain at the end about system related problem.</li><li>• Write your name and SID as a comment in the first line of your program.</li><li>• After allocated time i.e., 3 hours, 1 mark will be deducted for each late minute.</li><li>• If I detect plagiarism, I will make a Unfair Means Committee case.</li><li>• Save your program in a file named as your SID with .cpp extension, i.e. <b>FA21_BSE_XYZ.cpp</b> and submit this file only, nothing else. Do not put any space or anything else besides your SID in the file name.</li><li>• Submit only one file</li></ul>							

**QNO: 01 CLO [03]**

**[50]**

- 1) Take an array as input from user and its size through a function named **CreateArray()** and apply the following operations using functions and switch cases:
  - a) Apply bubble sort and sinking sort
  - b) Create a function named **Greatest()** and **Smallest()** to find smallest element from array
  - c) Apply linear search through a function named **LinearSearch()** to find location of that element in array.
  - d) Apply binary search through a function named **BinarySearch()** to find location of that element in array.
  - e) Ask the user that whether he wants to delete an element from array through a function named **DeleteElement()** or not which will be one cases of switch cases.
  - f) Create a function named **FindDuplicate()** to find duplicate from array if exists, Otherwise show a prompt message. **"No duplicate found in this array!!!"**
  - g) Create a function named **MODE()** to find mode of array elements.
  - h) Create a function named **MEDIAN()** to find median of array elements.
  - i) Create a function named **BubbleSort()** and **SinkingSort()** to sort array elements according to their algorithms.
  - j) Create a function named **FindFrequency()** to find frequency of asked element ask user if he wants to know frequency of more than one element.

- k) Create a function named **CopyArray()** that will copy the array to another array( store the array elements to other array) and also display that copied array elements.
- l) Create a function to add a new Element in array through a function named **AddElement()** if array has vacancy to add otherwise prompt user.
- m) Create a function named **FindFactorial()** to find factorial of element provide by user. **Hint:** [call the function **LinearSearch()** or **BinarySearch()** which will return index of element and then find factorial through recursion easily.]
- n) Create a function named **RandomElementHistogram()** that will store new elements in array randomly through

array randomly through **rand()** function and print the array index, frequency( element stored in that index), number of histograms \*'s (i.e. if **Array[1]=4** then print four stars and vice versa.) Note: range of random numbers should not exceed from eight and print the data in tabular format. Means array index in first column, elements in second and histograms in third respectively.

```

C:\Users\students It\Desktop\C++\Task 02.exe
FACE FREQUENCY HISTOGRAM
1 14 *****
2 17 *****
3 8 *****
4 21 *****
5 17 *****
6 23 *****

-----
Process exited after 0.04882 seconds with return value 0
Press any key to continue . . .

```

- o) Create a function which will simply display

the array elements.

- p) Call all these functions in switch cases individually.

Ask the user that which operation he wants to apply like this:

**Enter your choice:**

- 1) Press one to create array.
- 2) Press two to display array.
- 3) Press three to sort array by bubble sort.
- 4) Press four to sort array by sinking sort.
- 5) Press five to search element through linear search.
- 6) Press six to search element through binary search.
- 7) Press seven to copy array.
- 8) Press eight to find factorial.
- 9) Press nine to add new element.
- 10) Press ten to clear and add new element and print Histograms.
- 11) Press eleven to delete some element.
- 12) Press twelve to find frequency of element.
- 13) Press thirteen to find mode.
- 14) Press fourteen to find median.
- 15) Press fifteen to find greatest and smallest element.

16) Press sixteen to find duplicate element.

17) Press seventeen to exit.

QNO: 02

[20]

## **TASK NUMBER 1**

Write a program that store 3 students grades (4 test marks for each student).

■ Print these marks .

■ Display Average of each student .

■ Calculate minimum and maximum marks using a single function.

■ For example store marks as follows.

```
int studentGrades[ 3 ][ 4 ] = { { 77, 68, 86, 73 }, { 96, 87, 89, 78 }, { 70, 90, 86, 81 } };
```

The output should be like this:

```
The array is:
studentGrades[0]  [0]  [1]  [2]  [3]
studentGrades[1]  96   87   89   78
studentGrades[2]  70   90   86   81

Lowest grade: 68
Highest grade: 96
The average grade for student 0 is 76.00
The average grade for student 1 is 87.50
The average grade for student 2 is 81.75
```

QNO: 03 CLO [04]

[30]

Create a Matrix by user and perform the following tasks:

- Take matrix and its size from user using 2D array through a function named **"CreateMatrix()"**.
- Create another void function **TransposeMatrix()**, **SquareMatrix()**, **AddMatrix()**, **SubtractMatrix()** to find transpose, square, addition or subtraction of the matrix by itself.
- Create another function **TypeOfMatrix()** to tell whether matrix is symmetric, square, scalar, null, skew-symmetric, row or column matrix which will return a string which will tell matrix is of which kind.
- Create a function named **NumOfOddAndEven()** to find number of odd and even elements in the matrix.
- Create a exponential function named **ExpOfMatrix()** of integer datatype which will take the power of a user entered element from the matrix and also take the base from the user.
- Create another function named **SumOfDiagonals()**, **SumOfRows()**, **SumOfColumn()**, **ProductOfDiagonal()**, **ProductOfRow()**, **Product of column()** to find sum and product of diagonal, rows and column respectively. Ask the user that he wants to find the operation of which row, column, or diagonal.

**G)** *Make a choice for user like 1<sup>st</sup> program or user switch case, while loop or if condition which suitable it's up to you.*

***Good luck!!***