



## Lab-9 Manual

*for*

### Introduction to Programming (CS200)

**Dr. Mian Muhammad Awais**

#### LAB GUIDELINES

- **Make sure you submit the lab before 11:50 AM.** Any late submission will not be graded afterwards. In case of internet connectivity or electricity issues make sure to email your assigned TA before 2:00 PM. **You should email your assigned TA ONLY.**
- For every lab, there will be a folder created on LMS. You must submit your work in the respective folder during the lab time, you and only you are responsible for your submissions.
- You will be allowed to discuss the questions in the first half of the lab session for a few questions. After that, there will be a portion of lab where you cannot converse and must work for yourself. No discussion is allowed in later time period.
- You should do your work with utmost clarity and precision. Do not waste your time trying to do something you do not understand. Ask Lab instructors for help, that is what they are there for.

- Any legitimate cheating case can and will be reported to Disciplinary Committee without any leniency. Plagiarism Software make our task easier.
- Please follow the lab etiquettes and follow code of conduct in the session.
- Do not start Personal chat during your zoom meeting and raise your hands before asking questions.
- **Make separate .cpp files for each question. Naming convention for .cpp files is: YourRollNumber\_TaskX.cpp. Before submission, copy all the .cpp files in a folder named LabX\_AssignedTAName\_YourRollNumber.zip. Submit the .zip file only! (no .rar file submissions). X should be replaced by appropriate number.** Failure to follow the naming convention may lead to deduction of marks.

## OBJECTIVES

- Inheritance (protected and public)
- Dynamic memory

## LAB EXERCISES

### Question # 1 [Marks: 50]

Est. Time: 30 mins

In this question you are going to be tested on the concepts of public and protected inheritance. You are required to make 3 classes:

#### 1- class faculty:

- Data members:
  - **int numOfStudents** (private data member that stores number of students input by user)
  - **int\* marks** (protected data member that is an array of integers that will store marks of each student. The marks will be input by the user at run time)
- Member functions:
  - **Default and parametrized constructors** (prompt user to input marks of all the students in the parametrized constructor)
  - **Getters (getters) for BOTH the data members**

#### 2- class admin: (protected inheritance from class faculty)

- Data members:

- **String\* status** (private data member that is an array of strings that will store the grade (PASS/FAIL) of each student. The grades need to be computed based on the marks of the students)
- Member functions:
  - **Default and parametrized constructors** (you are supposed to calculate the grade (either “PASS” if marks  $\geq 50$  else “FAIL”) for each student based on the marks entered already by the user, in the parametrized constructor)
  - **void displayMarksAndStatus** (prints the marks and status/grade of all the students. Refer to the sample output for the formatting of the prompt. Needs to be followed exactly as is)

3- **class students:** (public inheritance from class admin)

- Data members:
  - **int num** (private data member that stores number of students input by user)
- Member functions:
  - **Default and parametrized constructors**
  - **void display** (makes a call to parent’s display function)

**NOTE:** You are required to make appropriate use of getters. If any data member can be accessed without its getter based on the type of inheritance relationship, you are not allowed to make use of a getter in that case.

Copy the following code as in int main():

```
int main()
{
    int n = 0;

    cout << "Enter number of students: ";

    cin >> n;

    students obj1(n);

    obj1.display();

    return 0;
}
```

## Sample Output

```
Enter number of students: 5
Enter student 1's marks: 12
Enter student 2's marks: 50
Enter student 3's marks: 98
Enter student 4's marks: 57
Enter student 5's marks: 25
Marks for all students have been set!
Grades for all students have been set!
Marks of student 1 are: 12. They FAIL
Marks of student 2 are: 50. They PASS
Marks of student 3 are: 98. They PASS
Marks of student 4 are: 57. They PASS
Marks of student 5 are: 25. They FAIL

Press any key to continue . . . ■
```

### Marks' distribution:

Correct skeleton for all 3 classes: 15 (5 for each)

Proper inheritance relationship setup between all the classes: 5

Use of inheritance relationships to make use of functions and to access data members: 20 (10 each)

Correct PASS/FAIL status setting: 5

Proper output formatting: 5

## Question # 2 [Marks: 50]

Est. Time: 40 mins

In this question, you will implement an **upper-triangular matrix**. An upper-triangular matrix is a matrix with **non-zero** entries **on the diagonal** and **non-zero** entries **above the diagonal**. The entries **below the diagonal** are **zero**. Since entries below the main diagonal are zero, **we don't want to store them**. This will save us memory.

1. **[20 marks]** Write a function that allocates memory to an upper triangular matrix. It will have **n** entries in the **first row**, **n-1** entries in the **second row**, **n-2** entries in the **third row** and so on (**n should be input from the user**). The function should also take as parameter a **pointer to a two-dimensional array**. You are **not allowed** to change the prototype of your function. Your function prototype **must** be as follows:

**Function prototype:** void AllocateTriangularMatrix (int \*\*\* arr, int n) where n is the number of rows and columns of the matrix as triangular matrices are square matrices

2. **[10 marks]** Write a function to **deallocate** this matrix.
3. **[10 marks]** Write a function to **input the contents** of this matrix from a **user**.
4. **[10 marks]** Finally write a function that **prints the matrix**.

Test all these functions by calling them in the **proper sequence (allocate, input, print, deallocate)** from **main ()**.

### Sample Output

```
C:\Users\rukhs\Lab8>g++ code.cpp &a
Enter matrix size: 5
1
2
3
5
1
2
7
9
4
6
3
4
5
2
1

Printing Contents!
1 2 3 5 1
0 2 7 9 4
0 0 6 3 4
0 0 0 5 2
0 0 0 0 1

Deallocating Memory!
C:\Users\rukhs\Lab8>
```

**Best of Luck!**