



## Lab-7 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

- Dynamic memory allocation and de-allocation
- 2D arrays
- Public inheritance

## LAB EXERCISES

### Question # 1 [Marks: 30]

**Est. Time: 30 mins**

**NOTE:** You do not have to do this question using classes

In this question, you will learn how to implement 2D arrays in C++. A 2D array may be treated as a **matrix**. Your program must have the following functionalities:

1. The **size** of the 2D array must be **input by the user**. [5 marks]
2. The **values** in the matrix must be **input by the user**. Make a **display** function that prints the matrix. When the program ends **deallocate** any dynamically allocated memory. [10 marks]
3. Print the **sum of all elements** in the matrix. [5 marks]
4. Print the **row-wise** and **column-wise sums** of the matrix. [5 marks]
5. **Matrix addition** of 2 matrices. [5 marks]

Further guidelines:

- Your program will have **3 matrices (of the same size)** in total, each stored in a separate variable (original matrix, matrix that you will add into your original matrix, matrix obtained after addition)
- User input for matrix elements must have a **numerical** data-type
- Do not forget to check out sample output

## Sample Output

```
D:\>g++ lab_7_2D.cpp &a
Enter the number of rows: 2
Enter the number of columns: 3
Enter row number 0
3
2
2
Enter row number 1
1
1
2
Original Matrix:
3 2 2
1 1 2
Sum of all elements in the matrix: 11
Sum of elements in row number 0 is: 7
Sum of elements in row number 1 is: 4
Sum of elements in column number 0 is: 4
Sum of elements in column number 1 is: 3
Sum of elements in column number 2 is: 4
Lets create a new matrix of the same size as our original matrix.
Enter row number 0
1
2
4
Enter row number 1
5
1
3
New Matrix:
1 2 4
5 1 3
By adding the two matrices we get the following matrix:
4 4 6
6 2 5
D:\>_
```

## Question # 2 [Marks: 70]

Est. Time: 90 mins

**NOTE:** Read through the entire question specifically the notes' and the assumptions before writing the code

In this question you will be making your own version of the popular game **Among Us**. You will be making two classes “**spaceship**” and “**players**”. Class **players** (child) inherits publicly from class **spaceship** (parent). The code for **spaceShip** class is given below. It is to be copied as is in your code.

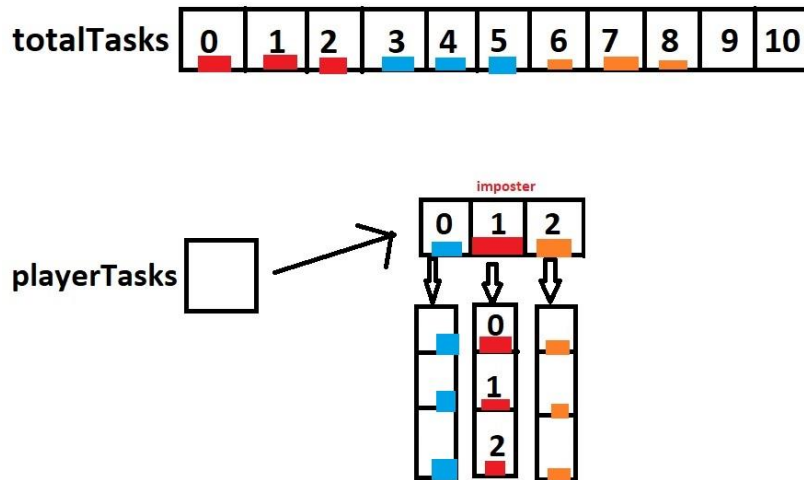
```
class spaceShip
{
    public:
        string totalTasks[10];
        spaceShip()
        {
            totalTasks[0] = "Sabotage";
            totalTasks[1] = "Vent";
            totalTasks[2] = "FakeTask2";
            totalTasks[3] = "Fix Wiring";
            totalTasks[4] = "Empty Trash";
            totalTasks[5] = "Download Data";
            totalTasks[6] = "Upload Data";
            totalTasks[7] = "Start Reactor";
            totalTasks[8] = "Shoot Asteroids";
            totalTasks[9] = "Redirect Power";
            totalTasks[10] = "02 Filter Clear";
        }
};
```

You are required to make the class `players` which has following **private** data members:

- **int numberOfPlayers**
- **int imposterCount**
- **string\* playerRole:** “imposter” or “crewmate”
- **string\*\* playerTasks:** points to an array of string pointers, each of which point to a string array of size 3 (which is to say that each player has 3 tasks)
- **int\* playerTaskCounter:** An integer array of size `numberOfPlayers`. Initialized with 3 for all indices indicating that all players have 3 tasks left to do (that is to say that none of the assigned tasks have been done yet)

You are also required to make the following **public** functions for this class:

- **Default Constructor:** sets default values for data members based on their data type
- **Parametrized Constructor:** takes **only one integer parameter** for `numberOfPlayers` and uses that to initialize the rest of the data members of this class
- **void setPlayerRoles:** Prompt the user to input `playerRole` for each player. A player can either be an “**imposter**” or a “**crewmate**”. If the user enters anything apart from these two inputs (case sensitivity and spelling wise as well) you should display this error message “**Invalid input. Check spelling**” and keep on taking inputs until all players have been entered correctly
- **void setPlayerTasks:** In class `spaceShip`, you have been provided with a list of total tasks that a spaceship has. The first three tasks (at index 0, 1 and 2) are reserved for the imposter so they have to be assigned to imposter’s `playerTasks`. The tasks for crewmates begin from index 3 of the `totalTasks` of `spaceShip`. You have to iterate through `totalTasks` and assign 3 tasks per crewmate. Refer to the following color coded picture to visualize the task mapping



- **int playGame:** Using the formula **rand() % numberOfPlayers** you have to generate a random number. This will be the player who has been selected randomly to do their tasks. If they have already completed all 3 of their tasks, then a new random number, that is, a different player needs to be selected to do their task. In case the selected player has not completed their tasks yet, they need to overwrite their current task with “DONE” and decrement their taskCounter. After that, you need to check for ending condition of the game that is: Either all crewmates have completed their tasks (in that case this function returns 2 and crewmates win) or imposter has completed its tasks before all crewmates did (in that case the function returns 1 and imposter wins the game)

int main()’s code is as follows (needs to be copied as is):

```
int main()
{
    srand(time(0));

    int n = 0;

    int returned = 0;

    cout << "Enter number of players: ";

    cin >> n;

    players obj(n);
```

```

    obj.setPlayerRoles();

    obj.setPlayerTasks();

    returned = obj.playGame();

    if(returned == 2)
    {
        cout << "Imposter won the game!" << endl;
    }

    else if(returned == 1)
    {
        cout << "Crewmates win the game!" << endl;
    }

    return 0;
}

```

**NOTE:** Refer to sample outputs for all the prompts and error messages to be displayed. KEEP THEM AS IS AS YOU WILL BE GRADED FOR THIS AS WELL. Provided code must NOT be changed!

**ASSUMPTION:**

- There must only be 1 imposter in a game
- Assume that maximum of only 3 players can play the game at a time

### Sample Output:

```
Enter number of players: 3
Enter player 1's role ("imposter" or "crewmate"): im
Invalid input. Check spelling
Enter player 1's role ("imposter" or "crewmate"): imposter
Enter player 2's role ("imposter" or "crewmate"): imposter
Imposter limit of 1 has been reached, please make all the other players crewmates!
Enter player 2's role ("imposter" or "crewmate"): crewmate
Enter player 3's role ("imposter" or "crewmate"): imposter
Imposter limit of 1 has been reached, please make all the other players crewmates!
Enter player 3's role ("imposter" or "crewmate"): crewmate
All players have been assigned their roles!
All players have been assigned their tasks!
The game has started!
crewmate on index 1 completed their task
imposter on index 0 completed their task
imposter on index 0 completed their task
crewmate on index 1 completed their task
crewmate on index 1 completed their task
imposter on index 0 completed their task
Imposter won the game!

Press any key to continue . . . ■
```

### Marks' Distribution:

Properly working default and parametrized constructors (includes initializations of 2D array as well): 15

Perfectly working assignRoles function: 10 (invalid input handling: 5, not adding more than 1 imposter: 5)

Perfectly working assignTasks function: 20 (imposter's: 10, all crewmates': 10)

Perfectly working playGame function: 20 (proper removal of done task: 10, proper implementation of winning conditions: 10)

Sample output (including same prompts): 5

**Best of Luck!**