Programming Fundamentals *COURSE & LAB* – Spring 2021
(BS-IT-F20 Morning & Afternoon)

Lab # 10 (Question# 1,2)

Assignment# 04 (Question# 3,4)

Assigned on: ***Friday, August 27, 2021***

Submission Deadline for Lab questions: ***Saturday, August 28, 2021 (till 11:59 AM)***
Submission Deadline of assignment questions: ***Wednesday, September 01, 2021 (till 6:00 PM)***

Instructions:

- **This is an individual assignment. Absolutely NO collaboration is allowed. Any traces of plagiarism/cheating would result in an "F" grade in this course.**
- Do **NOT** copy even a single line of code from any other person or book or Internet or any other source.
- This assignment will ALSO be counted towards your marks in **PF-Lab**.
- Late submissions will NOT be accepted, whatever your excuse may be.
- This assignment needs to be submitted in **Soft** form. The **Submission Procedure** is given on last page.
- Clearly mention your **Name**, **Roll Number** and **Section** in comments at the **top** of **each CPP file**.

There are two parts in this assignment. You are required to submit C++ programs for the 4 questions given in **Part A**. Questions in **Part B** are for your practice and you are not required to submit them. But I will assume in quizzes and exams that you have solved all of these questions.

PART A

Question # 1 — 10 Marks

Define a C++ function having four parameters. This function should take an array of integers and its size as its first two parameters. This function should determine the most frequent element of the array, and it should return the **most frequent element** (i.e. the element which occurs the most often) and its **frequency** through its *last two reference parameters*. In your function, you are NOT allowed to declare or use any other array. Illustrate the working of your function (on various inputs) in a complete C++ program. See the following examples:

Example #	Elements of the array	Most frequent element	Frequency
1	**5 3 4 5 2 7 5 4**	**5**	**3**
2	**12 74 23 35 23 12**	**12** or **23**	**2**
3	**7 5 6**	**7** or **5** or **6**	**1**

As you can see from the last two examples, if more than one element is the most frequent element, then your function can determine any one of them as the most frequent element.

Illustrate the working of your function (on at least 5 different examples/inputs) in a complete C++ program.

Question # 2 — 10 Marks

Define a C++ function which should remove all NEGATIVE values from an unsorted array of integers. Your function should also preserve the order of remaining elements in the array. At the end, this function should also return the count of the NEGATIVE values that were removed from the array. The prototype of your function should be:

```
int removeNegatives (int arr[], int size, int& newSize)
```

Note that the third reference parameter is used to send the updated size of the array back to the calling function.

For example, if **arr** contains these **7** elements **{ 11, -15, -2, 7, 11, 6, -8 }**, then, after the function call to **removeNegatives** the **arr** should now contain **{ 11, 7, 11, 6 }** in its first four indices. The variable **newSize** (see the 3rd parameter) should contain **4** (since, the *partially filled array* **arr** now contains 4 valid elements) and the function should return **3** (because, 3 negative values were removed from the array).

Important Note:

- In your function, you are NOT allowed to declare or use any other array.

Illustrate the working of your function (on at least 4 different examples/inputs) in a complete C++ program.

Question # 3 — 10 Marks

Define a C++ function:

```
void cyclicRotate (int arr [], int n, int k);
```

This function takes an array (**arr**) and its size (**n**) as its first two parameters. Third parameter is an integer value **k**. This function should *cyclically rotate* all the elements of the array **arr**, by **k** positions to the right.

For example,

if **arr** contains these **8** elements **{ 3, 4, 5, 8, 7, 2, 9, 1 }**,

then, after the function call **cyclicRotate (arr, 8, 3)**

the **arr** should contain these **8** elements **{ 2, 9, 1, 3, 4, 5, 8, 7 }**.

In your function:

- You can assume that **k** is greater than **0** and less than **n**
- You are NOT allowed to declare or use any other array.

Illustrate the working of your function (on at least 4 different examples/inputs) in a complete C++ program.

Hint 1: What happens if **k** is **1**?

Hint 2: You may implement another helper function to accomplish this task.

Hint 3: Do some paperwork.

Question # 4 — 15 Marks

After your graduation from PUCIT, you have been hired by Facebook as a Software engineer. On the first day of your job, you are required to write a C++ program for the following problem:

You are given an input file (**friends.txt**) containing data about friendship status of some persons. The first line in the input file contains a positive integer N (you can assume that $2 \leq N \leq 50$) indicating the number of persons in a small social network. The second line of the input file contains another positive integer F (indicating the number of friendships). After that there are F lines, each line containing a pair i, j of integers indicating that person i is friends with the person j.

Example input file: **friends.txt**

```
6
7
0,1
1,2
3,1
1,5
4,5
3,0
4,3
```

First of all, you are required to store the friendships in the social network using a $N \times N$ two-dimensional array of **short**s. In the case of above sample input file, you can visualize the friendships in the form a 6×6 array. Your code must be **generic** i.e. it must work for all values of N in the range $2 \leq N \leq 50$. For this, you can define a 50×50 two-dimensional array of **short**s, and use only the first N rows and first N columns of this two-dimensional array depending upon the value of N read from the input file.

Note that, friendship is a two-way phenomenon. Each pair i, j means that: person i is friend with person j, and also that person j is friend with person i. So, each pair i, j will cause two entries in the two-dimensional friendship array to be set to **1**. For example, the friendships given in the above example text file will be stored like this:

	0	**1**	**2**	**3**	**4**	**5**
0	0	1	0	1	0	0
1	1	0	1	1	0	1
2	0	1	0	0	0	0
3	1	1	0	0	1	0
4	0	0	0	1	0	1

5	0	1	0	0	1	0

Also note that all entries on the *main diagonal* (which runs from top-left to bottom-right corner) will always be ZERO i.e. a person cannot be friend with himself/herself.

The flow of your program will be as follows:

1. Read the data from the input file, and store it in a two-dimensional array (as described above).
2. Display the friendship table on screen, in a neat and readable way. All rows and columns should be properly labeled in the output.
3. Repeat the following logic until the user chooses to quit the program:
   - **a)** Ask the user to enter two different integers (let's say ***A*** and ***B***).
     Perform **input validation** to make sure that ***A*** and ***B*** are NOT same, and both are valid i.e. $\mathbf{0} \leq \boldsymbol{A} < \boldsymbol{N}$ and $\mathbf{0} \leq \boldsymbol{B} < \boldsymbol{N}$.
     Note: You MUST use more meaningful variable names.
   - **b)** Your program should determine and display the count and list of all persons which are *COMMON FRIENDS* of persons ***A*** and ***B***. See the following sample run to understand the working of this step.

Sample run for **Step # 3** in the above list is shown below. Note that this sample run corresponds to the example input file shown on the previous page. Text shown in **red** in entered by the user:

```
......
......

Enter two integers: 1 4
Person # 1 and 4 have 2 common friend(s) which is/are: 3 5

Continue (Y/N)? Y

Enter two integers: 4 0
Person # 4 and 0 have 1 common friend(s) which is/are: 3

Continue (Y/N)? Y

Enter two integers: 1 3
Person # 1 and 3 have 1 common friend(s) which is/are: 0

Continue (Y/N)? Y

Enter two integers: 4 5
Person # 4 and 5 have NO common friend(s)

Continue (Y/N)? N
```

Important Note: You MUST implement the logic of your program using at least **4 different functions** (apart from the **`main`** function).

PART B

Question # 5

For research purposes and to better help students, the academic office of PUCIT wants to know how well female and male students have performed in their degree programs. You receive a file that contains female and male student CGPAs. The letter code **`F`** is used for female students and the letter code **`M`** is used for male students. Every file entry consists of a letter code followed by a CGPA. Each line has one entry (see sample input file shown below). The number of entries in the file is unknown. The name of the input file should be **`CGPAs.txt`**.

Write a program that computes and outputs the average CGPA for both female and male students. Format your results rounded off to two decimal places (see sample output file shown below). The name of the output file produced by your program should be **`Summary.txt`**.

Sample input file:

`CGPAs.txt`

```
M 3.6
F 3.4
F 2.15
M 2.98
M 4
F 3.53
M 2.05
```

Sample output file:

`Summary.txt`

```
Average CGPA of 4 male students: 3.16
Average CGPA of 3 female students: 3.03
```

Your program should have at least the following four functions (apart from the **`main`** function):

- **`bool openFiles (ifstream&, ofstream&);`**
  This function opens the input and output files, and sets the output of the floating-point numbers to two decimal places in a fixed decimal format with a decimal point and trailing zeros. This function will return **true** if both files are successfully opened, otherwise it should return **false**. If both files are not opened successfully, the program should be terminated (from the *main* function) using the *exit* function.
- **`void readAndSumGPAs (ifstream&, double&, double&, int&, int&);`**
  This function reads the data from the input file (whose handle it receives as the first parameter), and determines and stores the *sum of male CGPAs*, *sum of female CGPAs*, *count of male students*, and *count of female students* in the next four reference parameters.
- **`void averageGPA (double, double, int, int, double*, double*);`**
  This function finds the average CGPA for male students and the average CGPA for female students and stores them in the variables pointed to by the last two parameters. The first four parameters are used to receive the *sum of male CGPAs*, *sum of female CGPAs*, *count of male students*, and *count of female students*.

- **`void outputResults (ofstream&, int, double, int, double);`**
  This function outputs the relevant results (see the sample output file) in the correct format to the output file (whose handle it receives as the first parameter). Next four parameters are used to receive the *count of male students*, *average CGPA of male students*, *count of female students*, and *average CGPA of female students*, respectively.

Question # 6

Write a C++ program for solving **Programming Challenge # 18** (*Tic-Tac-Toe Game*) on **Page 460** of your textbook (Gaddis, 9th Edition).

Question # 7

Write a C++ program for solving **Programming Challenge # 22** (*Theater Seating*) on **Pages 461-462** of your textbook (Gaddis, 9th Edition).

Submission Procedure:

i. You are required to submit C++ programs for the **first 2 questions** (given in **Part A**) are the lab tasks, where as **questions 3, and 4** (given in **Part A**) are considered as an assignment tasks.. The last 3 questions (given in Part B) are for your practice and you are not required to submit them. But I will assume in final exams that you have solved all of these questions/tasks ☺

ii. Put the Four (4) ***.CPP*** files containing your C++ code (names of the files **MUST** be **q1.cpp, q2.cpp, q3.cpp, q4.cpp**) in a folder (do NOT include any other files in your submission, otherwise your submission will NOT be graded). Don't forget to mention your **Name**, **Roll Number** and **Section** in comments at the top of each CPP file.

iii. Now, compress the folder (that you created in previous step) in **.RAR** or **.ZIP** format. The name of the RAR or ZIP file should be according to format:

`Mor-A3-BITF20M123` (for Morning section) OR
`Aft-A3-BITF20A456` (for Afternoon section),

where the text shown in **BLUE** should be your **complete Roll Number**.

iv. Finally, submit the (single) **.RAR** or **.ZIP** file through **Google Classroom**. Make sure that you press the ***SUBMIT*** button after uploading your file.

Note: *If you do not follow the above submission procedure, your Assignment will NOT be graded and you will be awarded ZERO marks.*

These *good programming practices* will also have their (significant) weightage in the marking of this assignment:

- Comment your code intelligently. **Uncommented code will NOT be given any credit.**
- Indent your code properly.
- Use meaningful variable names.

- Use meaningful prompt lines/labels for input/output.

If any program that you submit **does NOT compile correctly** or it **generates any kind of run-time error** you will get ZERO marks in the whole assignment.

☺ GOOD LUCK! ☺