Birla Institute of Technology and Science, Pilani, Rajasthan, India
Second Semester 2024-25
CS F111 – Computer Programming
Online Programming Test (Makeup)

SET - Makeup

==========

25/04/2025 **Max. Marks: 90 M** **Duration: 180 mins**

==========

General Instructions

- This question paper comprises two problems, whose details are described in the problem statements given on the next page. Read all the instructions and the problem statements carefully before attempting the test.
- Carefully follow the submission instructions **mentioned at the end of this page** before uploading your solutions on the **Dom Judge** portal.
- For a given problem, if you submit multiple submissions, only the latest one will be considered for evaluation. Whatever you submit on the Dom Judge portal will be considered final. **It is your responsibility to make sure that you are submitting the correct file against each problem**. **Also, be sure to save your files before you submit them.**
- **You are responsible for ensuring your solution is correctly submitted to the Dom Judge portal.** Later, if some student claims that he/she has submitted the solution and it doesn't appear on the Dom Judge portal, we won't entertain or listen to such issues.
- Programs that only have the input/output functions (e.g., *printf*()/*scanf()*) without the key programming logic, would **NOT** be considered for evaluation.

Instructions to attempt the test

- Create a directory in your home directory with the name **CPlabtest**.
- Download the files: "**Question Paper.pdf"** (PDF which you are reading currently), "**Q1.c**" and "**Q2.c**" from the domjudge portal. Copy these files into a directory **CPlabtest**. Rename them to **"Q1_<Your_13_digit_ID>.c"** and **"Q2_<Your_13_digit_ID>.c"**. For example, if your ID is **2024A1PS1234P**, rename Q1.c to **"Q1_2024A1PS1234P.c"**.
- The above ".c" files are the files you will be working on. You will have to complete the implementation of the incomplete functions in these files according to the problem statements.
- You should **NOT** do the following in the ".c" files:
  - **change anything in the main() function**
  - **change the function parameters or the return types of the functions**
  - **change or create new structure definitions**
  - **create new global variables**
  - **create new functions**
- Carefully observe how each function (that you will implement) is called in the **main()** function. Also, observe the sample execution shown at the end of each question.
- Follow the submission instructions properly while submitting your solutions on the portal.
- Evaluation of your programs will be based on the following factors: ***Correct Execution***, ***Correctness of Logic***, and ***Presentability of the Code***. Presentability includes usage of proper indentation, comments, etc.

Submission Instructions

The CPlabtest directory contains the following files: **"Q1_<Your_13_digit_ID>.c"** and **"Q2_<Your_13_digit_ID>.c"**. Please upload these files separately to their respective submission links on the portal. Make sure to save your files before uploading. You **don't** need to convert them into zip files.

After submitting, you can download each file to verify whether they were uploaded correctly. ***After verifying, put an additional signature on the attendance sheet before***

leaving the room. Any submission without this additional signature will NOT be evaluated.

Q1. [20+15+15=50M] [1 hour 30 mins]

You are given a 2D matrix of fixed size, where each cell stores a **singly linked list of integers**. The matrix is statically declared in the main function as:

```
Node* grid[ROWS][COLS];
```

Each cell in the matrix can contain **zero or more integers**, stored as a singly linked list. Your task is to implement the following functionalities:

1. Insert an integer into the linked list of a given cell **`(row, col)`** of the matrix. Insertion in a linked list can be done at the beginning of the list.
2. Compute the **sum of all integers** in a given row.
3. Compute the **sum of all integers** in a given column.

The above tasks are to be achieved by implementing the following functions:

- ***`void insertValue(Node* grid[][COLS], int row, int col, int val);`***

This function Inserts the value **`val`** at the **beginning** of the linked list located at **`grid[row][col]`**. The parameters taken as input include - **`grid`**: 2D matrix of linked list pointers; **`row, col`**: Coordinates of the matrix cell; and **`val`**: Integer to insert.

- ***`int rowSum(Node* grid[][COLS], int row);`***

This function returns the **total sum** of all integers across all columns in the specified row. The parameters taken as input include - **`grid`**: 2D matrix of linked list pointers; and **`row`**: Row number (0-indexed)

- ***`int colSum(Node* grid[][COLS], int col);`***

This function returns the **total sum** of all integers across all rows in the specified column. The parameters taken as input include - **`grid`**: 2D matrix of linked list pointers; and **`col`**: Column number (0-indexed)

User the structure definition of a node provided in Q1.c to implement the linked list. You don't need to explicitly store a header for the linked list or the count on number of elements stored in it.

Sample Execution:

```
jagat@Jagats-MacBook-Pro-2 Set PINK % gcc Q1.c
jagat@Jagats-MacBook-Pro-2 Set PINK % ./a.out
Updated array:
Sum of row 0: 60
Sum of col 0: 35
```

Q2. [15+25=40M] [1 hour 15 mins]

You are building a **student registry** for a class. The student registry is stored in the form of an array of structures. Each student record has:

- A name
- A roll number
- A **dynamically allocated array** of marks
- The number of marks they have

After populating this registry, you are required to:

1. Add a student's details and their dynamic marks array.
2. Filter out students whose average marks are **above a given threshold**, and store them in a **linked list of top performers**.

The above can be achieved by implementing the following functions:

- `void addStudent(Student* arr, int index, char* name, int roll, int* mArr, int n);`

The parameters taken as input for this function include: **`arr`**: The array of **`Student`** records; **`index`**: The index at which to add this student; **`name`**: Student's name; **`roll`**: Student's roll number; **`mArr`**: An array of marks; **`n`**: The number of marks in the **`mArr`**. Essentially, this function adds a student at the given **`index`** of the **`arr`**, using **`name`**, **`roll`**, **`mArr`** and **`n`**. It must allocate memory for **`marks`** dynamically and copy the values from the **`mArr`**. **`marks`** is a field in a student record.

- `Node* filterTopStudents(Student* arr, int n, float threshold);`

This function returns a **linked list of students** whose average marks exceed **`threshold`**. The input parameters include: **`arr`**: Array of students; **`n`**: Number of students in the array; **`threshold`**: Minimum average required for a student to be included in the linked list.

Implement both the above functions. Their declarations are present in "Q2.c". Use the structure definitions given in "Q2.c". Don't create any new structure definitions. You can implement the linked list without an explicit structure for the header.

Sample Execution:

```
jagat@Jagats-MacBook-Pro-2 Set PINK % gcc Q2.c
jagat@Jagats-MacBook-Pro-2 Set PINK % ./a.out
Top Students:
Alice (Roll: 1)
Carol (Roll: 3)
```

==== End of document ====