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

SET - ORANGE

==

13/04/2025 **Max. Marks: 90 M** **Duration: 180 mins**

==

General Instructions

- This question paper comprises three 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**", "**Q2.c**" and "**Q3.c**" from the domjudge portal. Copy the three **".c"** files into the directory **CPlabtest**. Rename your files to **"Q1_<Your_13_digit_ID>.c"**, **"Q2_<Your_13_digit_ID>.c"**, and **"Q3_<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 the 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"**, **"Q2_<Your_13_digit_ID>.c"**, and **"Q3_<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. [16M] [Expected time: 30 mins]

Given an array, **`arr`**, of size of 50 that stores unique integer values. **`arr`** presently contains 40 elements in its first 40 locations, and the remaining 10 locations don't store anything meaningful. The variable **`count`** stores the number of elements presently stored in **`arr`**, which is 40. Note that all 40 elements are sorted in descending order. Given an incomplete function **`insertInOrderDec()`** that takes an element **`x`** as a parameter and inserts **`x`** in **`arr`** such that the **`arr`** remains sorted. The function updates the count of valid elements in **`arr`** and returns the updated count. Complete the implementation of this function.

Sample Execution:

```
jagat@Jagats-MacBook-Pro-2 Set ORANGE % gcc q1.c
jagat@Jagats-MacBook-Pro-2 Set ORANGE % ./a.out
Updated array (count = 41):

100 98 96 94 92 90 88 86 84 83 82 80 78 76 74 72 70 68 66 64 62 60 58 56 54 52 50 48
46 44 42 40 38 36 34 32 30 28 26 24 22
```

Q2. [16+16=32M] [Expected time: 1 hour]

Given an array of 20 student records, where each record consists of a **`name`** (string), **`ID`** (integer), **`Math_Marks`** (integer) and **`Science_Marks`** (integer). (a) Write a function **`sortRecords()`** that takes the above array and sorts its elements in increasing order of marks attained by students in Science, using **Selection Sort**. This function also prints the sorted array. (b) Write a function **`searchRecord()`** that implements binary search on the above array to search for a record with given marks in Science. This function also prints the student record if it is present in the array and prints that the element is not found; otherwise. Complete the implementation of both functions.

Sample Execution:

```
jagat@Jagats-MacBook-Pro-2 Set ORANGE % gcc q2.c
jagat@Jagats-MacBook-Pro-2 Set ORANGE % ./a.out
Sorted Records by Science Marks (Ascending Order):
Name: Paul, ID: 116, Math: 73, Science: 65
Name: Steve, ID: 119, Math: 70, Science: 66
Name: Hannah, ID: 108, Math: 60, Science: 68
Name: Queen, ID: 117, Math: 93, Science: 70
Name: Ian, ID: 109, Math: 90, Science: 72
Name: David, ID: 104, Math: 80, Science: 75
Name: Oscar, ID: 115, Math: 81, Science: 76
Name: Liam, ID: 112, Math: 59, Science: 77
Name: Bob, ID: 102, Math: 92, Science: 79
Name: Jane, ID: 110, Math: 69, Science: 80
Name: Frank, ID: 106, Math: 83, Science: 82
Name: Nora, ID: 114, Math: 67, Science: 84
Name: Alice, ID: 101, Math: 78, Science: 85
Name: Kyle, ID: 111, Math: 74, Science: 86
Name: Tina, ID: 120, Math: 77, Science: 87
Name: Charlie, ID: 103, Math: 65, Science: 88
Name: Mona, ID: 113, Math: 88, Science: 89
Name: Eva, ID: 105, Math: 55, Science: 90
Name: Grace, ID: 107, Math: 72, Science: 91
Name: Rita, ID: 118, Math: 79, Science: 95
Record found:
Name: Charlie, ID: 103, Math: 65, Science: 88
```

Q3. [10+10+22=42M] [Expected time: 1 hour 20 mins]

Given 10 arrays of student records (**s1** to **s10**), each of **size 4**. Each array stores a list of students registered in its respective section (Sections 1 to 10). Each student record contains the **name** (string), **ID** (integer), **Math_Marks** (integer) and **Science_Marks** (integer). Write a function **createPointerArray()** that takes all these ten student record arrays as parameters and indexes them in an array of pointers **SectionArr**. All the arrays should be statically allocated. Write another function, **calculateSectionMax()**, that takes a student records array and computes the maximum **Math_Marks** students attain in that section. Now, using the above function, write a function **sortPointerArray()** that sorts **SectionArr** in the decreasing order of *the max* **Math_Marks** of each of the student arrays stored in it, using Selection Sort.

Sample Execution:

```
jagat@Jagats-MacBook-Pro-2 Set ORANGE % gcc q3.c
jagat@Jagats-MacBook-Pro-2 Set ORANGE % ./a.out
Max Math Marks of Sections (Before Sorting):
SectionArr[1]: 90
SectionArr[2]: 92
SectionArr[3]: 70
SectionArr[4]: 94
SectionArr[5]: 73
SectionArr[6]: 96
SectionArr[7]: 69
SectionArr[8]: 59
SectionArr[9]: 89
SectionArr[10]: 77

Max Math Marks of Sections (After Sorting):
SectionArr[1]: 96
SectionArr[2]: 94
SectionArr[3]: 92
SectionArr[4]: 90
SectionArr[5]: 89
SectionArr[6]: 77
SectionArr[7]: 73
SectionArr[8]: 70
SectionArr[9]: 69
SectionArr[10]: 59
```

==== End of document ====