General Instructions

- This lab test carries 20 marks.
- Program should be written in C language.
- Assume that all inputs are valid.
- Sample inputs are just indicative and need not handle every input case.
- Use of global variables is NOT permitted.
- The function prototypes given in the questions should not be changed.
- The input should be read from, and the output should be printed to, the console.
- **No clarifications regarding questions will be entertained. If there is any missing data you may make appropriate assumptions and write the assumptions clearly in the design sheet.**
- **For both Question 1 and Question 2, students have to first write the design and then only proceed with the implementation.**
- The duration of the Test is from **2:00 PM - 5:00 PM**.
- The test consists of two questions.
- At **3:00 PM**, the link for Question-1 submission will be replaced with a Question-1 **late submission link**, which would be active till 4:00 PM.
- Question-2 and its submission links will become available in the Eduserver at 3:00 PM.
- **The student has to decide whether to continue with the Question-1 or to proceed to the Question-2.**
- **If a student uses the Question-1 late submission link, then any submissions (both design and implementation) they make towards Question-2 will be ignored.**
- You must update your submissions on Eduserver at regular intervals, so that you don't end up having nothing uploaded at the end of the exam.
- **No email submissions will be allowed.** Marks will be given solely based on the **last submissions made on Eduserver within the deadlines**.
- There will be a viva voce during the test.
- **Design submission:**
    1. Read the question, understand the problem and write the design in pseudocode (in the shared Google document) for the indicated functions only. You are advised to **properly think about the solution before starting to write your design** to avoid wasting your time on rewrites.
    2. The design written should **clearly** convey your overall idea for the solution; the programming specific details may be changed over the course of the implementation. There will be a reduction in marks if the student writes C code instead of pseudocode.

3. **The edit permission of the shared document for writing the design of Question 1 and Question 2 will be revoked at 2:40 PM and 3:40 PM respectively.**
4. The pdf of the shared google document for writing the design of Question 1 and Question 2 should be uploaded in the EduServer before 2:50 PM and 3:50 PM respectively. **Only the designs uploaded in Eduserver will be considered for evaluation.**
5. Once designed, you should **write a program that implements your design**.

- **Mode of submission and timings:**

  1. Question 1 Design (upload the pdf of the shared google document in EduServer) - **2:50 PM**
  2. Question 1 Implementation (upload the zipped file of the source code in EduServer) - **3:00 PM**
     - At **3:00 PM** the submission link will be replaced with a Question-1 **late submission link**.
     - The **late submission link**, will be disabled at **4:00 PM**.
  3. Question 2 Design (upload the pdf of the shared google document in EduServer) - **3:50 PM**
  4. Question 2 Implementation (upload the zipped file of the source code in EduServer) - **4:50 PM**
     - The submission link will be disabled at **5:00 PM**.

- While implementing, you may use the source codes that you have previously submitted for the assignments, if you feel that it will be helpful.
- There will be some test cases that only test the correctness of specific functions and not the entire program. As such, if you are not able to complete your program, you should still **make sure that your submitted code will compile and run without errors** to get the marks for such test cases.
- The source code file should be named in the format

  `TEST<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c`

  (For example, TEST4_B190001CS_LAXMAN_1.c)

  The source file must be zipped and uploaded. Only zip files may be uploaded, even if they contain on a single .c file. The name of the zip file must be

  `TEST<NUMBER>_<ROLLNO>_<FIRST-NAME>.zip`

  (For example: TEST4_ B190001CS_LAXMAN.zip)

- Naming conventions **must** be strictly followed. Any deviations from the specified conventions, or associated requests after the exam, will not be considered.
- Any malpractice observed will lead to zero marks in the test. These will also be reported to the department for permission to award F grade in the course.

Mark distribution

Maximum marks – 20

- Question 1: 10 Marks (Design - 5 marks, Implementation and Test cases - 5 marks)
- Question 2: 8 Marks (Design - 4 marks, Implementation and Test cases - 4 marks)
- Viva voce - 2 marks

1. You are given the customer details of n customers of a bank. Each customer record contains the account number acc_no (integer), customer name $cust\_name$ (string) and the balance amount $balance$ (float) of a customer. The acc_no of each customer is unique.

   Write a program that implements the following functions as per the given function specifications. The function prototypes should not be changed.

   - $main()$: Repeatedly read a character 'r', 's', 'f' or 'd' from console and call the sub-functions appropriately until character 'e' is encountered.
   - $read\_and\_store(C, n)$: Read the account number acc_no, customer name $cust\_name$ and the balance amount $balance$ of n customers and store it in the array C.
   - $sort\_records(C, n)$: Sort the n customer records in the array C in the increasing order of their acc_no using the *Insertion Sort* algorithm.
   - $search\_record(C, a)$: Find the customer record with account number a from the set of n records in the array C using the *Linear Search* algorithm. If such a customer record is found, then print its index in the array C. Otherwise, print -1. (Assume that the array index starts from 0.)
   - $display(C, n)$: For each customer in the array C of length n, print the acc_no, $cust\_name$ and $balance$ (separated by a space), in a new line. (Print only the first two decimal digits of the float numbers.)

   Design Instructions

   - Write the design for the functions $read\_and\_store()$, $sort\_records()$, $search\_record()$, and $display()$ only.

   Input/Output Format

   The input consists of multiple lines. Each line may contain a character from { 'r', 's', 'f', 'd', 'e' } followed by zero or more integers. The integers, if given, are in the range $[1, 10^6]$.

   - Character 'r' : Character 'r'will be followed by an integer n. The next n lines contain the acc_no, $cust\_name$ and $balance$ of the n customers, separated by a space. Read the details using the function $read\_and\_store()$.
   - Character 's' : Sort the n customer records using the function $sort\_records()$.
   - Character 'f' : Character 'f' will be followed by an integer corresponding to the account number a. Print the position of the customer record whose account number is a using the function $search\_record()$.
   - Character 'd' : Print the details of all the customers using the function $display()$.
   - Character 'e' : End the program.

Sample Input

```
r 5
230124 James 2345.45
236572 Alen 34565.65
435678 John 78980.00
768432 Adil 45657.75
123432 Neena 57856.70
d
f 123432
s
d
f 345678
f 123432
e
```

Output

```
230124 James 2345.45
236572 Alen 34565.65
435678 John 78980.00
768432 Adil 45657.75
123432 Neena 57856.70
4
123432 Neena 57856.70
230124 James 2345.45
236572 Alen 34565.65
435678 John 78980.00
768432 Adil 45657.75
-1
0
```