General Instructions

- This lab test carries 20 marks. The test consists of two questions numbered 1 and 2.
- Programs should be written in C language.
- Assume that all inputs are valid.
- Sample inputs are just indicative.
- Use of global variables is NOT permitted.
- 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.**
- Solve Part 2 only after submitting a solution (design and code) for Part 1.
- **The students must start with the design of Part 1 and upload the design of Part 1 in the EduServer before 3:30 pm.**
- After uploading the design, students can begin the implementation of Part 1. The source file of Part 1 should be uploaded in the EduServer before 5:00 pm.
- There will be a viva voce during the test.
- Design Submission:
  1. Read the question, understand the problem and write the design (in a sheet of paper) for the indicated function(s) as algorithm(s)(in pseudocode), as per the given prototype.
  2. Take a clear photograph of the handwritten design sheet and submit through the link in eduserver.
  3. The design must be written using pseudocode conventions. There will be a reduction in marks if the student writes C code instead of pseudocode.
  4. For Part 2 also, upload the design first and then start the implementation. Students can upload Part 2 files (design/implementation) till 5:30 pm.
- **The implementation *must be* completely based on the design already submitted.**
- The source code file should be named in the format

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

  (For example, TEST1_B190001CS_LAXMAN_1.c)

  The source file must be zipped and uploaded. The name of the zip file must be

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

  (For example: TEST1_ B190001CS_LAXMAN.zip)

Mark distribution

Maximum marks – 20

- Question 1: 14 Marks (Design - 7 marks, Implementation and Test cases - 5 marks, Viva voce - 2 marks)
- Question 2: 6 Marks (Design - 3 marks, Implementation and Test cases - 3 marks)

1. You are given the unique register number reg_no and mark m of students in a class. You are asked to store these details in two arrays A and B respectively, each of size n. Write a C program that implements the following functions as per the function prototypes given below:

   - $store\_details(n, A, B)$ – reads the register number reg_no and mark m of a student and stores it in the arrays A and B respectively as size n as follows:
     - Find the position p of the student in the arrays A and B as $p = (m + 10) \bmod n$.
     - If position p is vacant, then store the details of the student in arrays A and B, at position p.
     - If position p is not vacant (already allotted to some other student), then store the details of the student in the next vacant position in the array A.
   - $display\_marks(A, B, n)$ – prints the details of the students in the arrays A and B of size n.

   Input Format

   - First line contains an integer n which is the size of the array A and B.
   - Second line contains an integer $s \leq n$ which is the number of students.
   - Next s lines contain, reg_no (integer) and mark m (integer) of a student which is to be stored in the arrays, separated by a space.

   Output Format

   - First s lines contain the position p of each student in the arrays.
   - Next n lines contain the reg_no (integer) and mark m (integer) of each student in the arrays, separated by a space. If an array position is vacant, print -1.

   Sample Input and Output

   Input

```
5
5
180010 20
180011 14
180012 25
180013 12
180014 11
```

   Output

```
0
4
1
2
```

```
3
180010 20
180012 25
180013 12
180014 11
180011 14
```

2. Given two n size arrays A and B to store register number reg_no and mark m respectively. Modify the function $store\_details(n, A, B)$ in Qn. 1 to accommodate the following changes: (You can decide the function prototypes based on your design.)

   - $store\_details()$ - Reads the register number reg_no and mark m of a student, and :
     - Find the position p of the student as $p = (m + 10)$ mod n.
     - If position p is vacant, then store the details of the student at position p.
     - If position p is not vacant (already allotted to some other student), then insert the details of the student in a suitable way, such that it is accessible from position p.
   - $sort\_details()$ - Arrange the details of the student that are allotted to the same position in non-decreasing order of mark m.
   - $display\_marks()$ – Prints the details of the students as described in the output format.

   Input Format

   - First line contains an integer n which is the size of the arrays A and B.
   - Second line contains an integer s which is the number of students.
   - Next s lines contain, reg_no (integer) and mark m (integer) of a student which is to be stored in the arrays separated by a space.

   Output Format

   - For each array index p, print in a new line, reg_no and mark m (separated by a comma) of the student that are allotted to position p.
   - In a line, details of each student should be separated by a space.
   - If an array position is vacant, print NULL

   Sample Input and Output

   Input

```
5
10
180010 50
180011 34
180012 25
180013 59
180014 12
180015 29
180016 40
180017 24
180018 56
180019 47
```

   Output

```
180012,25 180016,40 180010,50
180018,56
180014,12 180019,47
NULL
180017,24 180015,29 180011,34 180013,59
```