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. Consider the scenario of processing different transactions in a bank, based on the token system. A token is given to the customer in the order of their arrival time and the transaction is done in the order of the token number.

   In the waiting area, n seats, numbered 0 to $n-1$, are placed in a row for seating the customers. A customer enters the waiting area only when there is a vacant seat available. When a customer enters the waiting area, he/she gets a token and is given the next vacant seat to the right of the last customer who entered the waiting area. If there is no such seat available, he/she should be seated in the first vacant seat from the front of the row. Once a customer is allotted a seat, he/she remains in the same seat until he/she is called for the transaction.

   Please refer the example below with n=4.

allot seat for 101	101			
allot seat for 102	101	102		
allot seat for 103	101	102	103	
process transaction		102	103	
process transaction			103	
allot seat for 104			103	104
allot seat for 105	105		103	104
process transaction	105			104
allot seat for 106	105	106		104

   Write a C program that implements the following functions as per the function prototypes given below. (You have to store the seating information of the customers in an array A of size n).

   - $allot\_seat(A, n, t)$: Allot a seat to the customer with token number t, as described above. Also, print the seat number allotted to the customer, in a new line.
   - $process\_transaction(A, n)$: Call the next customer for the transaction in the order of their token number and delete the corresponding entry in array A. Also, print the seat number that was vacated, in a new line.

Input/Output Format

First line contains an integer n which is the number of seats.

Subsequent lines contain at most two integers where the first integer always represents the operation to be performed:

- If first integer is 0, read the second integer which is the token number of the new customer to be seated. Allot seat for the new customer using the $allot\_seat()$ function.
- If the first integer is 1, call the next customer for the transaction using the $process\_transaction()$ function.
- If the first integer is -1, then terminate the program.

Sample Input and Output

Input

```
4
0 101
0 102
0 103
1
1
0 104
0 105
1
0 106
-1
```

Output

```
0
1
2
0
1
3
0
2
1
```

2. The bank decides to give a preference for each customer based on their account levels. A customer with higher preference is called for transaction, before a customer with lower preference. If two of them have the same preference, they are called in the order of their token number.

   A customer enters the waiting area only when there is a vacant seat available. When a customer enters the waiting area, he/she is seated to the first vacant seat available. Once a customer is allotted a seat, he/she remains in the same seat until he/she is called for the transaction.

   The functions in Qn.1 should be modified to include the following: (You can decide the function prototypes based on your design.)

   - $allot\_seat()$: Allot a seat to the new customer and store his token number and preference. Also, print the seat number allotted to the customer, in a new line.
   - $process\_transaction()$: From the customers in the waiting area, call the customer with the highest preference for the transaction and delete the corresponding entry. Also, print the token number and preference of the customer who was called for the transaction, in a new line, separated by a space.
   - $update\_preference()$: Update the preference of the customer with given token number to t.
   - $sort\_customers()$: Using function $process\_transaction()$, print the details of the customers present in the waiting area, in the non-increasing order of their preference. The details of each customer should be printed in a new line.
   - $print\_customers$: Print the token number and preference of the customers in the order of their seat number, as described in the output format.

Input/Output Format

First line contains an integer n which is the number of seats.

Subsequent lines contain at most three integers where the first integer always represents the operation to be performed:

- If the first integer is 0, read the second integer corresponding to token number and third integer corresponding to preference of the new customer and and allot seat for him using the $allot\_seat()$ function.
- If the first integer is 1, call the next customer for the transaction using the $process\_transaction()$ function.
- If the first integer is 2, the second integer corresponds to the token number of a customer and the third integer corresponds to that customer's new preference p. Update the preference of the given customer using $update\_preference()$ function.
- If the first integer is 3, print the details of the customers present in the waiting area, in the non-increasing order of their preference using the function $sort\_customers()$.
- If the first integer is 4, print n lines where the i^{th} line contains the token number and preference of the customer, separated by a space, seated at the i^{th} seat. If the i^{th} seat is vacant, print -1.
- If the first integer is -1, then terminate the program.

Sample Input and Output

Input

```
5
0 101 20
0 102 13
0 103 25
0 104 24
1
2 102 28
4
0 105 28
1
3
-1
```

Output

```
0
1
2
3
103 25
101 20
102 28
-1
104 24
-1
2
102 28
101 20
104 24
105 28
```