

July 2025 CSE 208  
Offline on RBT and AVL  
Submission: Sunday, 15/02/2026, 11:55 pm

**Problem 1:** A super-computer named TEUB has many running processes. Each process has a priority  $x$ . Two active processes in TEUB cannot have equal priorities. When a process finishes its task, it gets terminated. Sometimes, the programmers of TEUB want to know how many running processes have priorities less than  $y$ . The programmers hired you to help them using the Red-Black tree.

The input has four types of commands.

- Initiation of a program
- Termination of a program
- Searching for a program
- Find the programs with lower priority

### Input

The first line of input shows the total number of commands ( $N$ ).

Each of the following  $N$  commands has two integers  $e_i$  and  $x_i$ .

$e_i$	Meaning
0	Terminate the program with priority $x_i$ .
1	Initiate a program with priority $x_i$ .
2	Search the program with priority $x_i$ .
3	Find the number of programs with priority less than $x_i$ .

### Output

The first line of input shows the number of output lines.

For each command, you have to print three integers  $e_i$ ,  $x_i$  and  $r_i$ .

$r_i$  signifies the result of the corresponding command.

$e_i$	$r_i$
0	1 if successful termination.

	0 if there is no program with priority $x_i$
1	1 if successful initiation 0 if there is already a program with priority $x_i$
2	1 if found 0 if not found
3	The number of programs with priority $< x_i$ .

### Sample I/O

Sample Input	Sample Output	Explanation
11	11	Line count
1 1	1 1 1	Successful initiation 1
1 2	1 2 1	Successful initiation 2
1 3	1 3 1	Successful initiation 3
1 1	1 1 0	Same priority (1) exists
0 1	0 1 1	Successful termination 1
0 4	0 4 0	No priority (4) exists
2 3	2 3 1	Priority 3 found
2 5	2 5 0	Priority 5 not found
1 1	1 1 1	Successful initiation 1
3 3	3 3 2	2 programs having priority $< 3$
3 6	3 6 3	3 programs having priority $< 6$

### Constraints

$$1 \leq N \leq 10^5$$

$$1 \leq x_i \leq 10^6$$

$$0 \leq e_i \leq 3$$

Each command must be executed in logarithmic time.

### More instructions

- The program must accept input from a file and output to a file.
- Write the Red-Black Tree code so that it can be reused for other purposes, regardless of the data type.

**Problem 2:** You have to implement an AVL tree, which is a self-balancing binary search tree.

This implementation provides the following features:

- Insertion: You can insert a new node with a given key and value into the AVL tree. The tree will automatically balance itself after the insertion.
- Deletion: You can delete an existing node with a given key from the AVL tree. The tree will automatically balance itself after the deletion.
- Traversal: You can traverse the AVL tree in different orders, such as pre-order, in-order, post-order, or level-order. The tree will print the keys and values of each node during the traversal.

## Input

The first line of input shows the total number of commands (N).

Each of the following N commands has two integers  $e_i$  and  $x_i$ .

$e_i$	Meaning
0	Delete an existing node with a given key $x_i$ from the AVL Tree.
1	Insert a new node with a given key $x_i$ into the AVL Tree.
2	Traverse the AVL Tree in different orders, such as pre-order (when $x_i = 1$ ), in-order (when $x_i = 2$ ), post-order (when $x_i = 3$ ), or level-order (when $x_i = 4$ ).

## Output

The first line of input shows the number of output lines.

For each command, you have to print three integers:  $e_i$ ,  $x_i$  and  $r_i$ , except for the traversal command.

$r_i$  signifies the result of the corresponding command.

$e_i$	$r_i$
0	1 if successful removal. 0 if there is a node with key $x_i$ .
1	1 if successful insertion. 0 if there is already a program with priority $x_i$ .

For  $e_i = 2$ , Traverse and Print the tree

### Sample I/O

Sample Input	Sample Output	Explanation
16	16	Line count
1 9	1 9 1	Successful insertion 9
1 5	1 5 1	Successful insertion 5
1 10	1 10 1	Successful insertion 10
1 0	1 0 1	Successful insertion 0
1 6	1 6 1	Successful insertion 6
1 11	1 11 1	Successful insertion 11
1 -1	1 -1 1	Successful insertion -1
1 2	1 2 1	Successful insertion 2
1 1	1 1 1	Successful insertion 1
1 1	1 1 0	Same value exists
2 1	9 1 0 -1 5 2 6 10 11	Pre-order traversal
2 2	9 1 10 0 5 11 -1 2 6	Level-order traversal
0 10	0 10 1	Successful removal 10
0 10	0 10 0	Value not exist
2 1	1 0 -1 9 5 2 6 11	Pre-order traversal
2 2	1 0 9 -1 5 11 2 6	Level-order traversal

### Constraints

$$1 \leq N \leq 10^5$$

$$1 \leq x_i \leq 10^6$$

$$0 \leq e_i \leq 2$$

### More instructions

- The program must accept input from a file and output to a file.
- Write the AVL tree code so that it can be reused for other purposes, regardless of the data type.

### Submission

- Include only source files
- Do not include executable binaries, input/output files
- Place your files in a folder named 2305XXX\_1 for RBT and 2305XXX\_2 for AVL
- Place your folders in a folder named 2305XXX
- Zip the folder
- Submit to Moodle after renaming it to 2305XXX.zip