Assignment 4 : CSL 201
Topic : Treaps

The following describes the outline of a template class implementing treaps:

```
template <class T>
class TreapNode {
friend class Treap<T>;
// your code here
}

template <class T>
class Treap {
private:
TreapNode<T> *root;
// your code here
}
```

Your objective is to add the following functions to the template class for treaps. Each function should run in $O(\log n)$ expected time, where n is the total number of nodes in the treap.

1. **Search.** Searches for a key in the treap. Returns a pointer to the key if it is present, otherwise returns **NULL**.

   ```
   T* search(T x) {};
   ```

2. **Insert.** Inserts a key into treap. If the key is already present, prints a "Key already present" message.

   ```
   void insert(T x) {};
   ```

3. **Delete.** Deletes the item with a particular key from the treap. If the key is not present, prints a "Key not present" message.

   ```
   void delete(T x) {};
   ```

4. **k-th Smallest.** Given an integer k, returns the kth smallest element in the treap.

   ```
   T kthSmallest(int k) {};
   ```

   Note that to implement this function in $O(\log n)$ expected time, you will have to augment the data structure by adding extra field in each treap node.

5. **Three Way Join.** Given a key k and two treaps T_1 and T_2 such that all keys in treap T_1 are $< k$ and all keys in treap T_2 are $> k$, returns a treap T containing all keys in $T_1 \bigcup \{k\} \bigcup T_2$.

```
Treap& threeWayJoin(T k, Treap& t1, Treap& t2) {};
```

6. **Split.** Given a key k and a treap T on which the function is called, makes two treaps T_1 and T_2 of the same order as T such that T_1 contains all keys in treap T which are $< k$, and T_2 contains all keys in treap T which are $> k$.

```
void split(T k, Treap& t1, Treap& t2) {};
```

1 Input Format

Your algorithm will maintain a sequence of treaps $T_0, T_1, T_2, \cdots$ and so on. Initially, all the treaps are empty. You can assume that there will never be more than 10000 treaps in the program.

The first line of the input will contain a single number n, which denotes the number of treap operations. Each line after that will contain a single instruction which can be an insert, a delete, a split, a join, or finding the k-th smallest element.

An insert instruction consists of the keyword "insert" followed by the number i of the treap T_i in which insertion is to be made. After this is a single number n, which denotes the number of integers to be inserted. This is followed by a sequence of n integers. Note that we allow batch insertions, i.e., a single insert instruction can be used to insert any number of items into a treap.

For example, to insert 10 numbers $45, 13, 11, 77, 2, 9, 1, 66, 5, 22$ into treap number 4, we will give the instruction:

```
insert 4 \\ insert operation on treap number 4
10 \\ number of integers to be inserted
45 13 11 77 2 9 1 66 5 22 \\ the list of integers to be inserted
```

A deletion instruction is similar to an insert instruction, except that we use the keyword "delete". For example, the following instruction deletes the set of numbers $7, 32, 41, 15, 16, 19, 22$ from treap 10:

```
delete 10 \\ delete operation on treap number 10
7 \\ number of integers to be deleted
7 32 41 15 16 19 22 \\ list of integers to be deleted
```

The instruction of a three-way join, consists of the keyword "join" followed by the number x of the first treap, the key k, and the number y of the second treap. This is followed by the number z of the treap which is obtained by joining T_x, k, and T_y. After the join, treaps x and y will be empty, and treap z will contain all the keys in $T_x \bigcup \{k\} \bigcup T_y$. You can assume that all keys in the first treap are less than k, and all keys in the second treap are greater than k.

For example, the instruction to join treaps 5 and 8 with intermediate key 77 into treap 11 is:

```
join 5 77 8 11
```

The instruction for a split, consists of the keyword "split" followed by the number x of the treap to be split, the splitting key k, and the numbers y and z of the target treaps. After the split, treap y consists of all keys in treap x that were less than k, treap z consists of all keys in treap x which were greater than k, and treap x itself becomes empty.

Thus, the instruction to split treap 5 about key 61 with target treaps 11 and 17 is:

```
split 5 61 11 17
```

The instruction for k-th smallest consists of the keyword "select" followed by the treap number and the rank of the key to be returned. To find the 11th smallest key in treap 7, we will write the following:

```
select 7 11
```

The output of "select" instruction is a single number on a line by itself, which is the value of the k-th smallest key in the corresponding treap.

Note that all other instructions except "select" have no output.

We are giving six input test files $in0, in1, in2, \cdots, in5$ in the tar archive "input-assgn4.tar.gz". The description of each test case along with the correct output is given in the file "test_cases".

2 Example

This example is illustrative and hence the treaps considered here consist of small number of keys.

Suppose your program is given the following input:

```
7
insert 0 5 1 3 5 7 9
insert 1 5 11 14 15 16 17
delete 1 2 14 16
join 0 10 1 2
split 2 7 3 4
select 3 2
select 4 4
```

The first number is equal to 7, and tells us that there are 7 instructions in the input. The first instruction inserts 5 numbers into treap 0. After this, $T_0 = (1\ 3\ 5\ 7\ 9)$. The second instruction inserts 5 numbers into treap 1. After this, $T_1 = (11\ 14\ 15\ 16\ 17)$. The third instruction deletes two numbers 14 and 16 from treap 1. After this, $T_1 = (11\ 15\ 17)$. The fourth instruction joins treaps 0 and 1 along with intermediate key 10 into treap 2. After this instruction treaps 0 and 1 are empty, and treap 2 contains the keys (1 3 5 7 9 10 11 15 17). The fifth instruction splits treap 2 about key 7 into treaps 3 and 4. After this instruction, treaps $0, 1$, and 2 are empty, and $T_3 = (1\ 3\ 5)$ and $T_4 = (9\ 10\ 11\ 15\ 17)$. The sixth instruction asks for the 2nd smallest key in treap 3. The output of this instruction is the number 3 on a line by itself. The seventh instruction asks for the 4th smallest key in treap 4. The output of this instruction is the number 15 on a line by itself.

The final output of the program consists of just two numbers, which are the outcome of the two "select" instructions:

```
3
15
```

Note that all instructions except "select" have no output.