Assignment 4 : CSL 201
Topic : Skip-lists

Your objective in this assignment is to implement the skip-list data structure so that it supports each of the following operations in $O(\log n)$ expected time:

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

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

2. **Insert.** Inserts a key into the skip-list. 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 skip-list. 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 skip-list.

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

   To achieve this operation in $O(\log n)$ expected time, you will have to augment the skip-list data structure by adding a field in every node.

The following code snippet tosses a coin with the probability of heads equal to p where $0 < p < 1$:

```
#include <stdlib.h>
#include <math.h>

int toss_coin(float p) {
float x = ((1.0*rand())/RAND_MAX);
if(x <= p)
return 1;
else
return 0;
}
```

Choose $p = 1/2$ for your implementation of skip-lists.

1 Input Format

The program will maintain a single skip-list S, which will initially be empty. You can assume that all numbers in an insert, delete, or select operation on the skip-list are non-negative integers.

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

An insert instruction consists of the keyword "insert" followed by a single number n, which denotes the number of integers to be inserted. This is followed by a sequence of n positive integers. Note that we allow batch insertions, i.e., a single insert instruction can be used to insert any number of items into the skip-list.

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

```
insert \\ keyword
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 the skip-list:

```
delete \\ keyword
7 \\ number of integers to be deleted
7 32 41 15 16 19 22 \\ list of integers to be deleted
```

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

```
select 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 skip-list.

Note that all other instructions except "select" have no output. We are giving five test cases *in0*, *in1*, *in2*, *in3*, and *in4* 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_skip_list".

2 Example

Suppose your program is given the following input:

```
5
insert 5 1 3 5 7 9
insert 5 2 4 6 8 10
select 5
delete 3 2 5 8
select 4
```

The first number is equal to 5, and tells us that there are 5 instructions in the input. The first instruction inserts 5 numbers into the skip-list. After this, $S = (1\ 3\ 5\ 7\ 9)$. The second instruction inserts 5 more numbers into the skip-list. After this, $S = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10)$. The third instruction asks for the 5th smallest number in the skip-list, which is equal to 5. The program prints the result on a line by itself. The fourth instruction deletes trees three numbers $2, 5$, and 8 from the skip-list. After this instruction, the skip-list S contains the numbers (1 3 4 6 7 9 10). The final instruction asks for the 4th smallest number in the skip-list. The output of this instruction is the number 6 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:

```
5
6
```

Again, note that all instructions except "select" have no output.