Cut the Rope

Limits: 5s, 512 MB

Bokkor is a rope seller. He has a special tool for cutting rope. But recently this tool is behaving weirdly. When he tries to cut the rope, it puts the cut in a random position of the rope from the starting position. He doesn't have enough money to repair the machine. As his friend, you need to find the $\mathbf{k^{th}}$ smallest rope segment after cutting the rope few times with the machine.

Input

There will be several test cases. First line of the input contains the number **T** < 40 (Number of test cases).

Every test case start with a value **N** (Length of the rope) and **Q**(number of query). Next **Q** lines contain a character (**'C'** or **'F'**) and an integer **'X'**. If the query starts with **'C'**, then the machine cut the rope at position **X**. If it starts with **'F'** then find the Length of the $\mathbf{X^{th}}$ rope segment, after sorting the ropes by length. Queries will always be valid.

Constraints:
$\mathbf{1 < N < 10^{15}}$
$\mathbf{1 \le Q \le 10^{5}}$
$\mathbf{1 \le X \le 10^{15}}$

Output

For every test case, the first line contains **"Case X:"**. **X** is the case number.
For every query starts with **F**. you need to print the length of the desired rope in a single line.

Samples

Input	Output
1 10 5 C 4 F 1 F 2 C 9 F 1	

```cpp
/// https://toph.co/p/cut-the-rope
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;
using namespace __gnu_cxx;

#define ll long long
#define pll pair<ll,ll>

typedef tree<
pll,
null_type,
less<pll>,
rb_tree_tag,
tree_order_statistics_node_update>
ordered_set;

ordered_set os;
set<pll>ss;

int main(){
    int tt; scanf("%d",&tt);
    for(int ks=1; ks<=tt; ks++){
        printf("Case %d:\n",ks);
        ll n,q; scanf("%lld %lld",&n,&q);

        os.clear();
        ss.clear();

        int id = 0;
        os.insert({0,0});
        os.insert({n,++id});

        ss.insert({n,1});

        while(q--){
            char ch[2]; ll x;
            scanf("%s %lld",&ch,&x);
            if(ch[0]=='C'){
                auto it = ss.upper_bound({x,-1});
                pll cur = (*it);
                ll st = cur.second, ed = cur.first;
                ll len = ed-st+1;

                int cnt = os.order_of_key({len,-1});
                auto jt = os.find_by_order(cnt);

                os.erase(*jt);
                os.insert({x-st+1, ++id});
                os.insert({ed-x, ++id});

                ss.erase(it);
                ss.insert({x,st});
                ss.insert({ed,x+1});

            }else{
                ll ans = (*os.find_by_order(x)).first;
                printf("%lld\n",ans);
            }
        }
    }
    return 0;
}
```