题意

初始所有点都为1
三种操作:
D x 将x从1变成0
Q x 包含x的最长连续1区间长度
R　将最后一次D操作的x变回1

分析

线段树节点维护三个信息:
1. 区间最长全1前缀长度 treel
2. 区间最长全1后缀长度 treer
3. 区间最长连续1长度　trees
当前区间为x,左孩子lc,右孩子rc
update:
treel[x] = treel[lc]
如果 treel[lc] 满了 treel[x] +=treer[lc]
同理
treer[x] = treer[rc]
如果 treer[rc] 满了 treer[x] +=treel[rc]
trees[x] = max(trees[lc]， trees[rc] ,treer[lc]+treel[rc])
query:
如果当前区间为空或满或为叶子节点 返回trees[o]
如果x在左区间，如果 x在最长后缀范围内 返回treer[lc]+treel[rc],否则查询左区间
同理如果x在右区间，如果x在最长前缀范围内 返回treer[lc]+treel[rc]，否则查询右区间
删除顺序:
用栈维护

代码

```
#include<iostream>
#include<cmath>
#include<cstring>
#include<cstdio>
#include<stack>
#define For(i,a,b) for(int i=(a); i<=(b) ; i++)
```

```cpp
#define _For(i,a,b) for(int i=(a); i>=(b) ; i--)
#define Memset(a,b); memset((a),(b),sizeof((a)));
#define Cin(a); scanf("%d",&(a));
#define Cinc(a); scanf(" %c",&(a));
#define Cins(a); scanf("%s",(a));
#define Cout(a,b);  printf("%d",(a));printf(b);
#define Coutc(a,b);  printf("%c",(a));printf(b);
#define Couts(a,b);  printf("%s",(a));printf(b);
using namespace std;
typedef  long long LL;
typedef  unsigned long long ULL;
typedef  long double LDB;
inline LL readint() {LL x;cin>>x;return x;}
int treel[200005],treer[200005],trees[200005];
int n,m;
int D = -1;
char cmd;
int x;
int ans;
void update(int o,int l,int r,int x,int d)
{
    if(l == r && l == x){
        treel[o] = treer[o] = trees[o] = d;
        return;
    }
    if(l == r) return;
    int M = (l+r)>>1;
    if(M>=x) update(o<<1,l,M,x,d);
    else update(o<<1|1,M+1,r,x,d);
    treel[o] = treel[o<<1];
    treer[o] = treer[o<<1|1];
    trees[o] = max(trees[o<<1],trees[o<<1|1]);
    trees[o] = max(trees[o],treer[o<<1]+treel[o<<1|1]);
    if(trees[o<<1] == M -l+1){
        treel[o]+=treel[o<<1|1];
    }
    if(trees[o<<1|1] == r-(M+1)+1){
        treer[o]+=treer[o<<1];
    }
}
void build(int o,int l,int r)
{
    treel[o] = treer[o] = trees[o] = r-l+1;
    if(l == r) return ;
    int M = (l+r)>>1;
    build(o<<1,l,M);
    build(o<<1|1,M+1,r);
    return ;
}
int query(int o,int l,int r,int x)
{
    if(l == r|| trees[o] == 0|| trees[o] == r-l+1)
    {
        return trees[o];
    }
    int M = (l+r)>>1;
    if(M>=x)
    {

         if(x>= (M-treer[o<<1]+1))
         {
            return (treer[o<<1]+treel[o<<1|1]);

         }
            return query(o<<1,l,M,x);

    }
    else
    {
        if(x<=M+treel[o<<1|1])
        {
            return (treel[o<<1|1]+treer[o<<1]);
        }
            return query(o<<1|1,M+1,r,x);
    }
}
stack<int>s;
int main()
{
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        while(!s.empty())s.pop();
        build(1,1,n);
        while(m--)
        {
            Cinc(cmd);
            if(cmd == 'D')
            {
                Cin(x);
                s.push(x);
                update(1,1,n,x,0);
            }
            else if(cmd == 'Q')
            {
                Cin(x);
                printf("%d\n",query(1,1,n,x));
            }
            else if(cmd == 'R')
            {
                if(s.empty()) continue;
                D = s.top();
                s.pop();
                update(1,1,n,D,1);
            }
        }
    }
}
```