C国的死对头A国这段时间正在进行军事演习，所以C国间谍头子Derek和他手下Tidy又开始忙乎了。A国在海岸线沿直线布置了N个工兵营地,Derek和Tidy的任务就是要监视这些工兵营地的活动情况。由于采取了某种先进的监测手段，所以每个工兵营地的人数C国都掌握的一清二楚,每个工兵营地的人数都有可能发生变动，可能增加或减少若干人手,但这些都逃不过C国的监视。   
中央情报局要研究敌人究竟演习什么战术,所以Tidy要随时向Derek汇报某一段连续的工兵营地一共有多少人,例如Derek问:“Tidy,马上汇报第3个营地到第10个营地共有多少人!”Tidy就要马上开始计算这一段的总人数并汇报。但敌兵营地的人数经常变动，而Derek每次询问的段都不一样，所以Tidy不得不每次都一个一个营地的去数，很快就精疲力尽了，Derek对Tidy的计算速度越来越不满:"你个死肥仔，算得这么慢，我炒你鱿鱼!”Tidy想：“你自己来算算看，这可真是一项累人的工作!我恨不得你炒我鱿鱼呢!”无奈之下，Tidy只好打电话向计算机专家Windbreaker求救,Windbreaker说：“死肥仔，叫你平时做多点acm题和看多点算法书，现在尝到苦果了吧!”Tidy说："我知错了。。。"但Windbreaker已经挂掉电话了。Tidy很苦恼，这么算他真的会崩溃的，聪明的读者，你能写个程序帮他完成这项工作吗？不过如果你的程序效率不够高的话，Tidy还是会受到Derek的责骂的.

Input

第一行一个整数T，表示有T组数据。   
每组数据第一行一个正整数N（N<=50000）,表示敌人有N个工兵营地，接下来有N个正整数,第i个正整数ai代表第i个工兵营地里开始时有ai个人（1<=ai<=50）。   
接下来每行有一条命令，命令有4种形式：   
(1) Add i j,i和j为正整数,表示第i个营地增加j个人（j不超过30）   
(2)Sub i j ,i和j为正整数,表示第i个营地减少j个人（j不超过30）;   
(3)Query i j ,i和j为正整数,i<=j，表示询问第i到第j个营地的总人数;   
(4)End 表示结束，这条命令在每组数据最后出现;   
每组数据最多有40000条命令

Output

对第i组数据,首先输出“Case i:”和回车,   
对于每个Query询问，输出一个整数并回车,表示询问的段中的总人数,这个数保持在int以内。

Sample Input

1

10

1 2 3 4 5 6 7 8 9 10

Query 1 3

Add 3 6

Query 2 7

Sub 10 2

Add 6 3

Query 3 10

End

Sample Output

Case 1:

6

33

59

树状数组模板题

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int c[50011],n;

int lowbit(int x)//计算lowbit

{

return x&(-x);

}

void add(int i,int val)//将第i个元素增加val

{

while(i<=n)//在不断更新i的父节点

{

c[i]+=val;

i+=lowbit(i);//i的父节点为p，则p=i+lowbit(i)

}

}

int sum(int i)//求前i项和

{

int s=0;

while(i>0)

{

s+=c[i];

i-=lowbit(i);

}

return s;

}

int main()

{

int i ,j=0,a,b,v,t,num;

char str[]="Add",str1[]="Sub",str2[]="End",sub1[6];

scanf("%d",&t);

while(t--)

{

scanf("%d",&n);

printf("Case %d:\n",++j);

memset(c,0,sizeof(c));

for(i=1;i<=n;++i)

{

scanf("%d",&v);

add(i,v);

}

while(1)

{

scanf("%s",sub1);

if(strcmp(str2,sub1)==0)//End

break;

if(strcmp(sub1,str)==0) //Add

{

scanf("%d %d",&a,&b);

add(a,b);

}

else if(strcmp(sub1,str1)==0)//Sub

{

scanf("%d %d",&a,&b);

add(a,-b);

}

else//query求和

{

scanf("%d %d",&a,&b);

printf("%d\n",sum(b)-sum(a-1));//因为如果询问区间2-4的话

//sum[4]-sum[2]=a[3]+a[4]却不包含2

}

}

}

return 0;

}