//线段树 单点替换、单点增减、区间求和、区间最值

#define lson l , m , rt << 1

#define rson m + 1 , r , rt << 1 | 1

const int maxn = 222222;

int MAX[maxn<<2]; int MIN[maxn<<2]; int SUM[maxn<<2];

void PushUP(int rt){

MAX[rt] = max(MAX[rt<<1] , MAX[rt<<1|1]);

MIN[rt] = min(MIN[rt<<1] , MIN[rt<<1|1]);

SUM[rt] = SUM[rt<<1] + SUM[rt<<1|1];}

void build(int l,int r,int rt) {

if (l == r){

scanf("%d",&MAX[rt]);

MIN[rt] = MAX[rt];

SUM[rt] = MAX[rt];

//printf("mi = %d\n",MIN[rt]);

//printf("ma = %d\n",MAX[rt]);

return ;}

int m = (l + r) >> 1;

build(lson);

build(rson);

PushUP(rt); }

void update(int p,int tihuan,int l,int r,int rt){

if (l == r) {

MAX[rt] = tihuan;

MIN[rt] = tihuan;

SUM[rt] = tihuan;

return ;}

int m = (l + r) >> 1;

if (p <= m) update(p , tihuan ,lson);

else update(p , tihuan , rson);

PushUP(rt);}

void update1(int p,int add,int l,int r,int rt){

if (l == r) {

SUM[rt] = SUM[rt] + add;

return }

int m = (l + r) >> 1;

if (p <= m) update1(p , add ,lson);

else update1(p , add , rson);

PushUP(rt);}

int query(int L,int R,int l,int r,int rt){

if (L <= l && r <= R)

return MAX[rt];

int m = (l + r) >> 1;

int ret = -1;

if (L <= m) ret = max(ret , query(L , R , lson));

if (R > m) ret = max(ret , query(L , R , rson));

return ret;

int query1(int L,int R,int l,int r,int rt){

if (L <= l && r <= R)

return MIN[rt];

int m = (l + r) >> 1;

int ret = 99999;

if (L <= m) ret = min(ret , query1(L , R , lson));

if (R > m) ret = min(ret , query1(L , R , rson));

return ret;}

int queryhe(int L,int R,int l,int r,int rt){

if (L <= l && r <= R)

return SUM[rt];

int m = (l + r) >> 1;

int ret = 0;

if (L <= m) ret += queryhe(L , R , lson);

if (R > m) ret += queryhe(L , R , rson);

return ret}

int main(){

int n , m;

while (~scanf("%d %d",&n,&m)){

build(1 , n , 1);

while (m --)

char op[2];

int a , b;

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

if (op[0] == 'Q') //区间求最大

printf("%d\n",query(a , b , 1 , n , 1));

else if(op[0]=='U') //单点替换

update(a , b , 1 , n , 1);

else if(op[0]=='M')//区间求最小

printf("%d\n",query1(a , b , 1 , n , 1));

else if(op[0]=='Q')//区间求和

printf("%d\n",queryhe(a , b , 1 , n , 1));

else if(op[0]=='S')//单点增加

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

update1(a , b , 1 , n , 1);

else if(op[0]=='E')//单点减少

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

update1(a , -b , 1 , n , 1);