Lazy Propagation Basic Code(Range query + Range Update):

Given an array of n elements, which are initially all 0. The array is indexed from 0 to n-1. After that you will be given q commands. They are -

1. 0 x y v - you have to add v to all numbers in the range of x to y (inclusive), where x and y are two indexes of the array.

2. 1 x y - output a line containing a single integer which is the sum of all the array elements between x and y (inclusive).

$n(1 \le n \le 10^5)$, $q(1 \le q \le 50000)$, $(1 \le v \le 1000)$, $(0 \le x \le y < n)$.

Sample code:

```
#define mx 100005
#define ll long long int
struct dt{
    ll sum,prop;
}tree[mx*4];

void update(ll node,ll b,ll e,ll x,ll y,ll v)
{
    if(b>y||e<x)return;
    if(b>=x&&e<=y){
        tree[node].sum+=(e-b+1)*v;
        tree[node].prop+=v;
        return;
    }
    ll left=node*2; ll right=node*2+1; ll mid=(b+e)/2;

    update(left,b,mid,x,y,v);
    update(right,mid+1,e,x,y,v);
    tree[node].sum=tree[left].sum+tree[right].sum+(e-b+1)*tree[node].prop;
}

ll query(ll node,ll b,ll e,ll x,ll y,ll carry)
{
    if(b>y||e<x) return 0LL;
    if(b>=x&&e<=y) return tree[node].sum+(e-b+1)*carry;

    ll left=node*2; ll right=node*2+1; ll mid=(b+e)/2;

    ll p1=query(left,b,mid,x,y,carry+tree[node].prop);
    ll p2=query(right,mid+1,e,x,y,carry+tree[node].prop);
    return p1+p2;
}
```

```
int main()
{
    ll t; scanf("%lld",&t);

    for(ll kase=1; kase<=t; kase++)
    {
        printf("Case %lld:\n",kase);
        ll n,q; scanf("%lld%lld\n",&n,&q);

        memset(tree,0,sizeof(tree));

        for(ll i=0; i<q; i++)
        {
            ll type; scanf("%lld",&type);

            if(type==0)
            {
                ll x,y,v; scanf("%lld%lld%lld",&x,&y,&v);

                update(1,0,n-1,x,y,v);
            }
            else
            {
                ll x,y; scanf("%lld%lld",&x,&y);

                ll ans=query(1,0,n-1,x,y,0LL);

                printf("%lld\n",ans);
            }
        }
    }

    return 0;
}
```

Another Sample Code:

```
#define mx 100005
#define ll long long int
struct dt{
    ll sum,prop;
}tree[mx*4];

void update(ll node,ll b,ll e,ll x,ll y,ll v)
{
    if(b==x&&e==y){
        tree[node].sum+=(e-b+1)*v;
        tree[node].prop+=v;
        return;
    }
    ll left=node*2; ll right=node*2+1; ll mid=(b+e)/2;

    if(y<=mid){
        update(left,b,mid,x,y,v);
    }
    else if(x>mid){
        update(right,mid+1,e,x,y,v);
    }
    else{
        update(left,b,mid,x,mid,v);
        update(right,mid+1,e,mid+1,y,v);
    }
    tree[node].sum=tree[left].sum+tree[right].sum+(e-b+1)*tree[node].prop;
}

ll query(ll node,ll b,ll e,ll x,ll y,ll carry)
{
    if(b==x&&e==y) return tree[node].sum+(e-b+1)*carry;

    ll left=node*2; ll right=node*2+1; ll mid=(b+e)/2;

    if(y<=mid){
        return query(left,b,mid,x,y,carry+tree[node].prop);
    }
    else if(x>mid){
        return query(right,mid+1,e,x,y,carry+tree[node].prop);
    }
    else{
       ll p1=query(left,b,mid,x,mid,carry+tree[node].prop);
       ll p2 = query(right,mid+1,e,mid+1,y,carry+tree[node].prop);
       return p1+p2;
    }
}
```

```
int main()
{
    ll t; scanf("%lld",&t);
    for(ll kase=1; kase<=t; kase++)
    {
        printf("Case %lld:\n",kase);
        ll n,q; scanf("%lld%lld\n",&n,&q);
        memset(tree,0,sizeof(tree));

        for(ll i=0; i<q; i++)
        {
            ll type; scanf("%lld",&type);
            if(type==0)
            {
                ll x,y,v; scanf("%lld%lld%lld",&x,&y,&v);
                update(1,0,n-1,x,y,v);
            }
            else
            {
                ll x,y; scanf("%lld%lld",&x,&y);
                ll ans=query(1,0,n-1,x,y,0LL);
                printf("%lld\n",ans);
            }
        }
    }
    return 0;
}
```