Segment Tree Applications

1. Given an array of n elements. You are given a query from L to R of value K. Find # of elements $\geq k \to [L, R]$

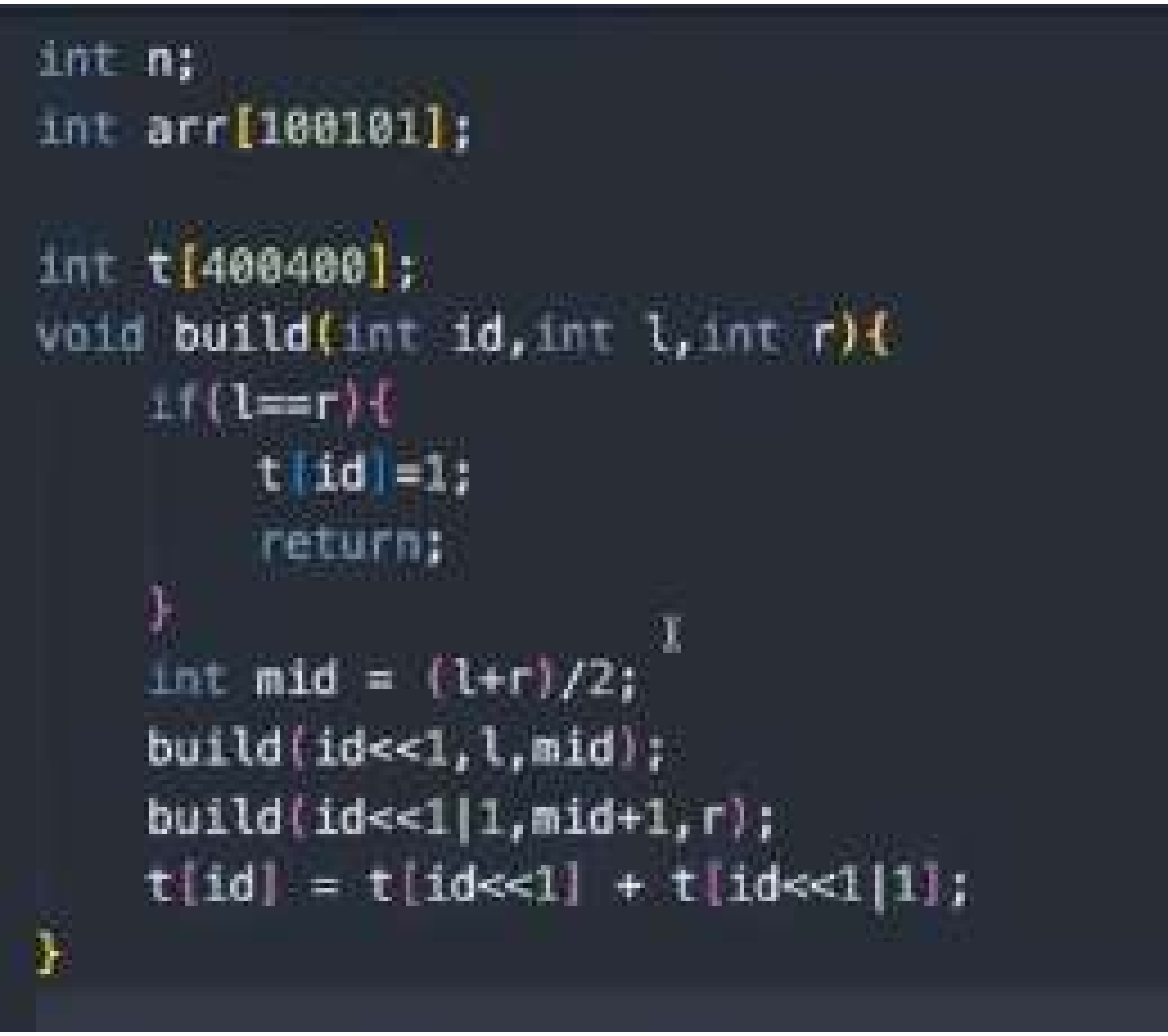
$N \leq 10^5$

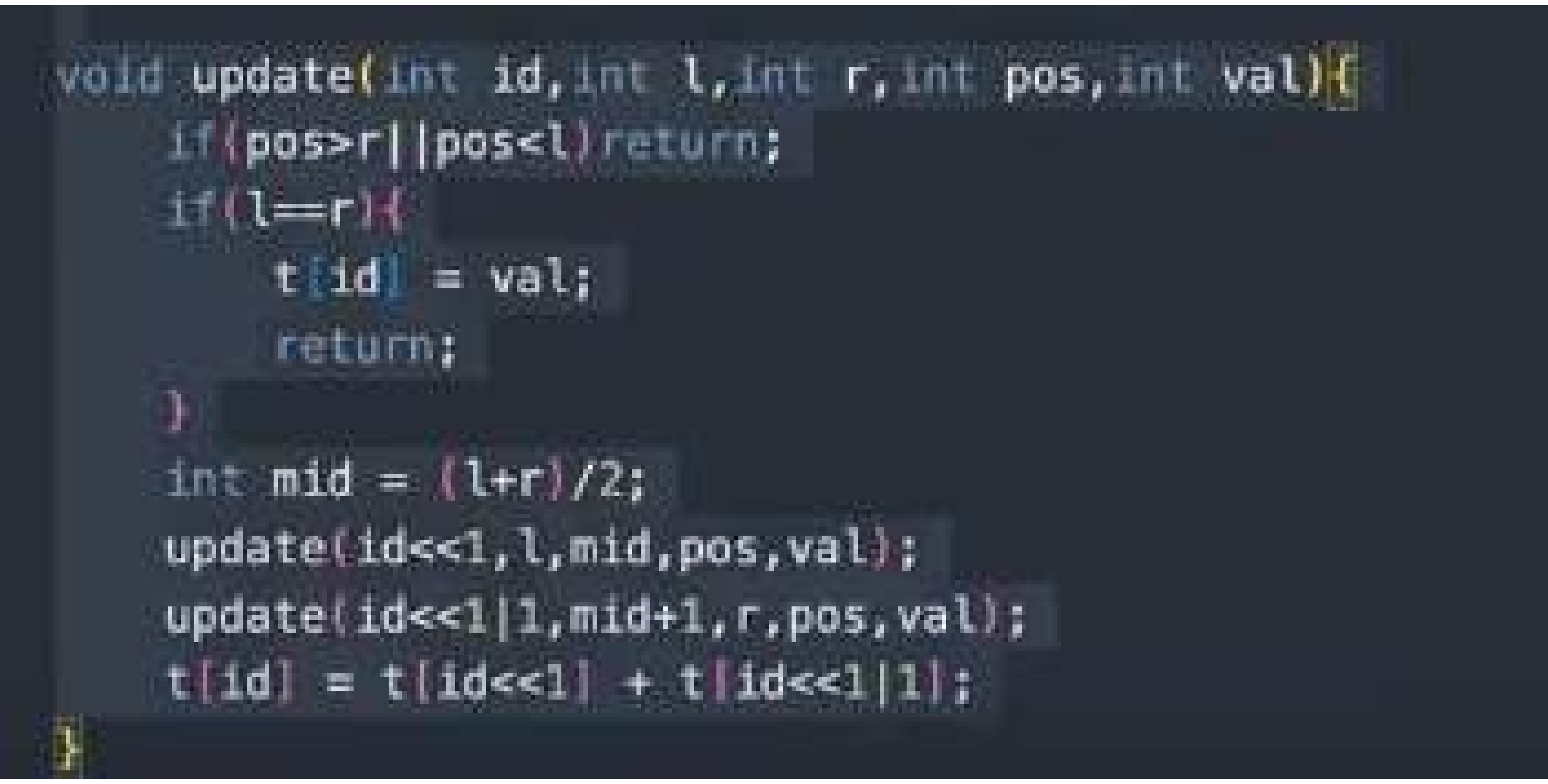
$Q \leq 10^5$

$arr[i], K \leq 10^5$

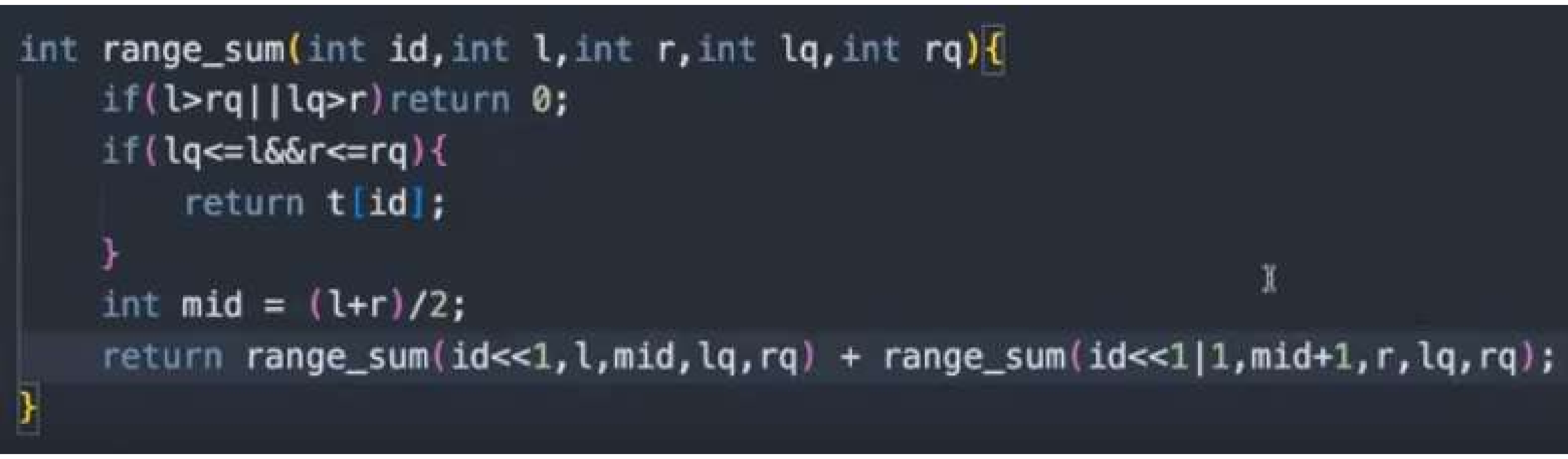
Approach :-

1. Sort array elements in descending order of value.
2. Sort queries in descending order of K.
3. Tree stores 1 at index i if $arr[i] >$ current K , otherwise 0
4. So, range sum give count of elements $> K$.

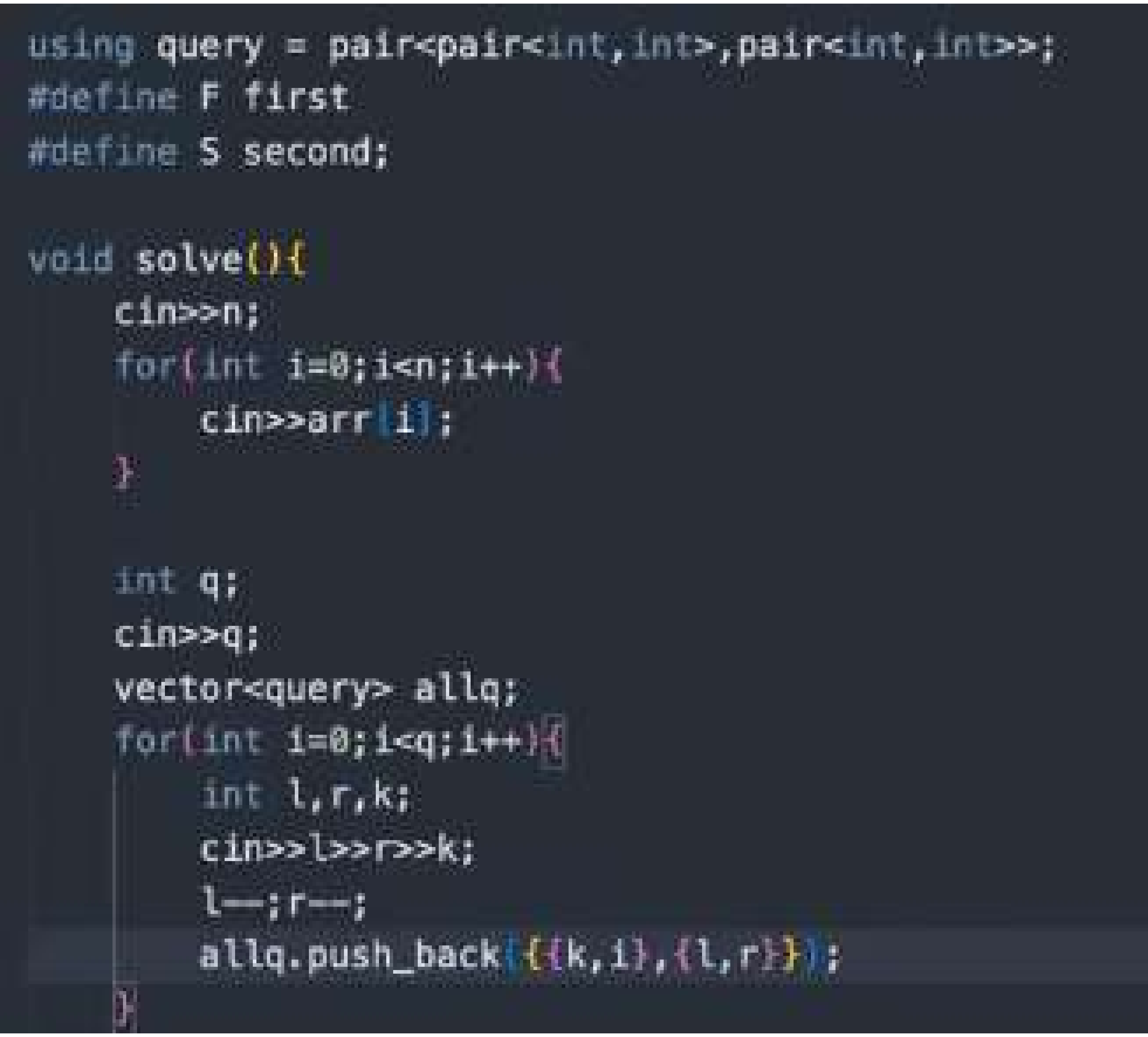

```
int n;
int arr[100101];

int t[400400];
void build(int id,int l,int r){
    if(l==r){
        t[id]=1;
        return;
    }
    int mid = (l+r)/2;
    build(id<<1,l,mid);
    build(id<<1|1,mid+1,r);
    t[id] = t[id<<1] + t[id<<1|1];
}
```



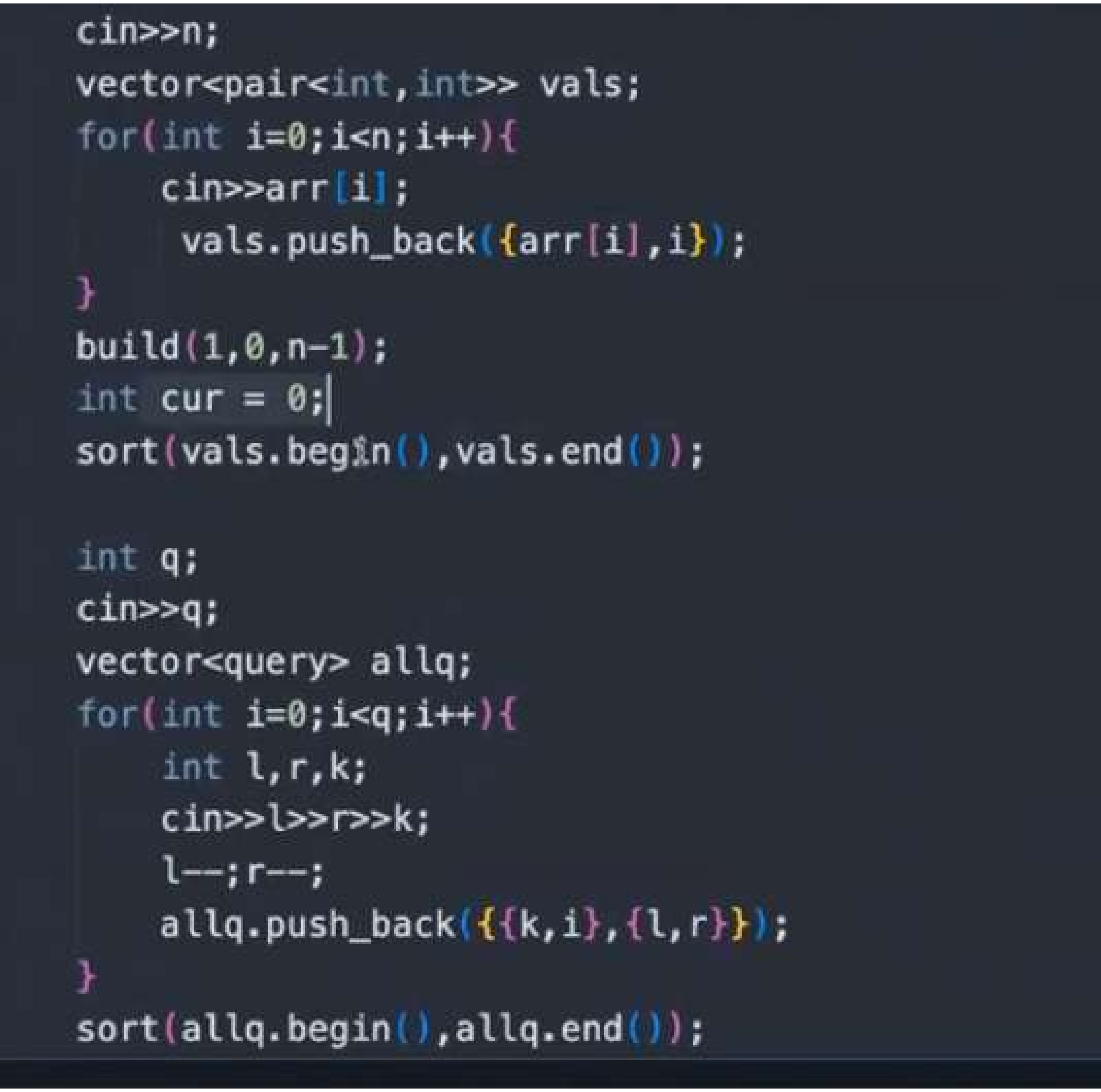
```
void update(int id,int l,int r,int pos,int val){
    if(pos>r||pos<l)return;
    if(l==r){
        t[id] = val;
        return;
    }
    int mid = (l+r)/2;
    update(id<<1,l,mid,pos,val);
    update(id<<1|1,mid+1,r,pos,val);
    t[id] = t[id<<1] + t[id<<1|1];
}
```



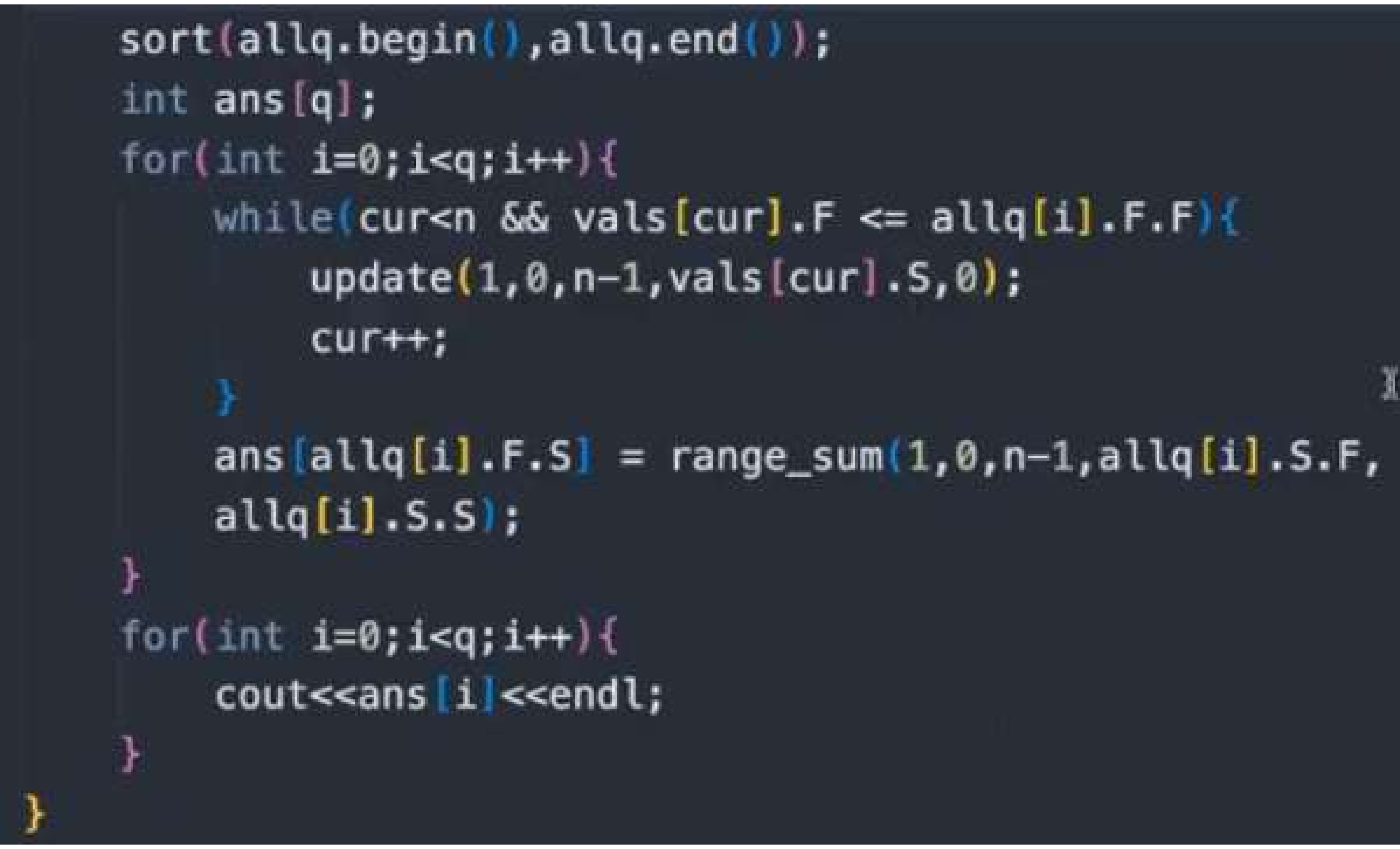
```cpp
int range_sum(int id,int l,int r,int lq,int rq){
    if(l>rq||lq>r)return 0;
    if(lq<=l&&r<=rq){
        return t[id];
    }
    int mid = (l+r)/2;
    return range_sum(id<<1,l,mid,lq,rq) + range_sum(id<<1|1,mid+1,r,lq,rq);
}
```



```cpp
using query = pair<pair<int,int>,pair<int,int>>;
#define F first
#define S second;

void solve(){
    cin>>n;
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }

    int q;
    cin>>q;
    vector<query> allq;
    for(int i=0;i<q;i++){
        int l,r,k;
        cin>>l>>r>>k;
        l--;r--;
        allq.push_back({{k,i},{l,r}});
    }
```



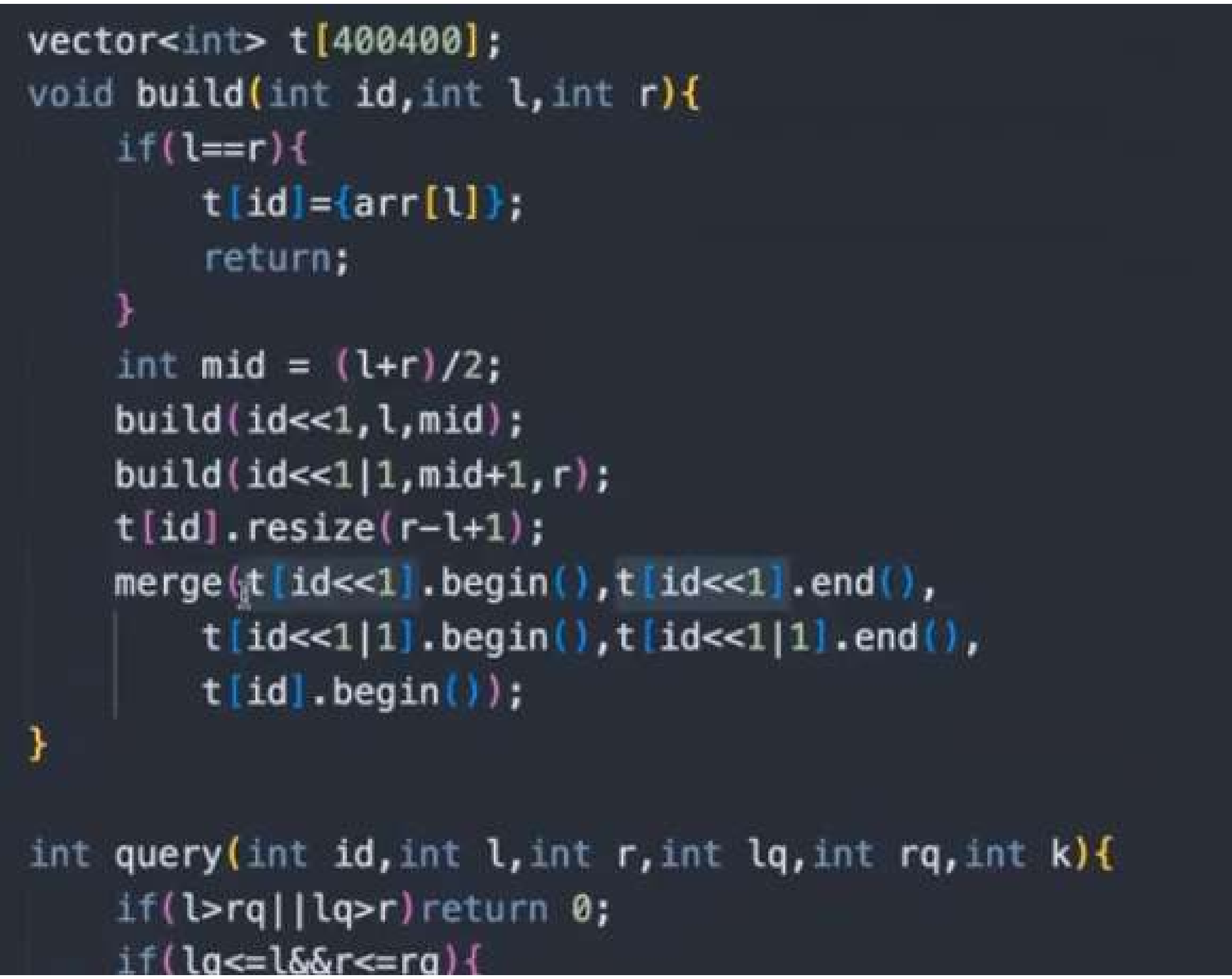
```
cin>>n;
vector<pair<int,int>> vals;
for(int i=0;i<n;i++){
    cin>>arr[i];
     vals.push_back({arr[i],i});
}
build(1,0,n-1);
int cur = 0;
sort(vals.begin(),vals.end());

int q;
cin>>q;
vector<query> allq;
for(int i=0;i<q;i++){
    int l,r,k;
    cin>>l>>r>>k;
    l--;r--;
    allq.push_back({{k,i},{l,r}});
}
sort(allq.begin(),allq.end());
```



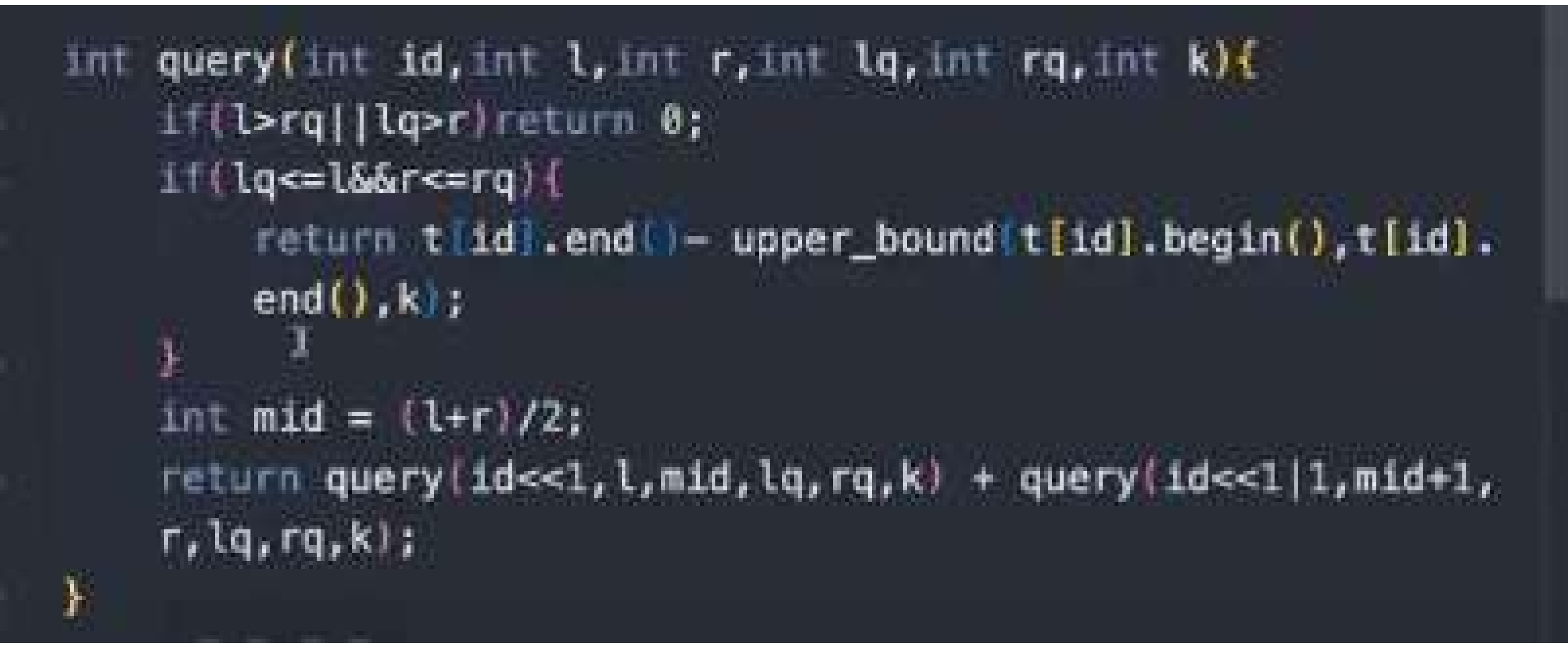
```
    sort(allq.begin(),allq.end());
    int ans[q];
    for(int i=0;i<q;i++){
        while(cur<n && vals[cur].F <= allq[i].F.F){
            update(1,0,n-1,vals[cur].S,0);
            cur++;
        }
        ans[allq[i].F.S] = range_sum(1,0,n-1,allq[i].S.F,
        allq[i].S.S);
    }
    for(int i=0;i<q;i++){
        cout<<ans[i]<<endl;
    }
}
```

Q. K - Query online

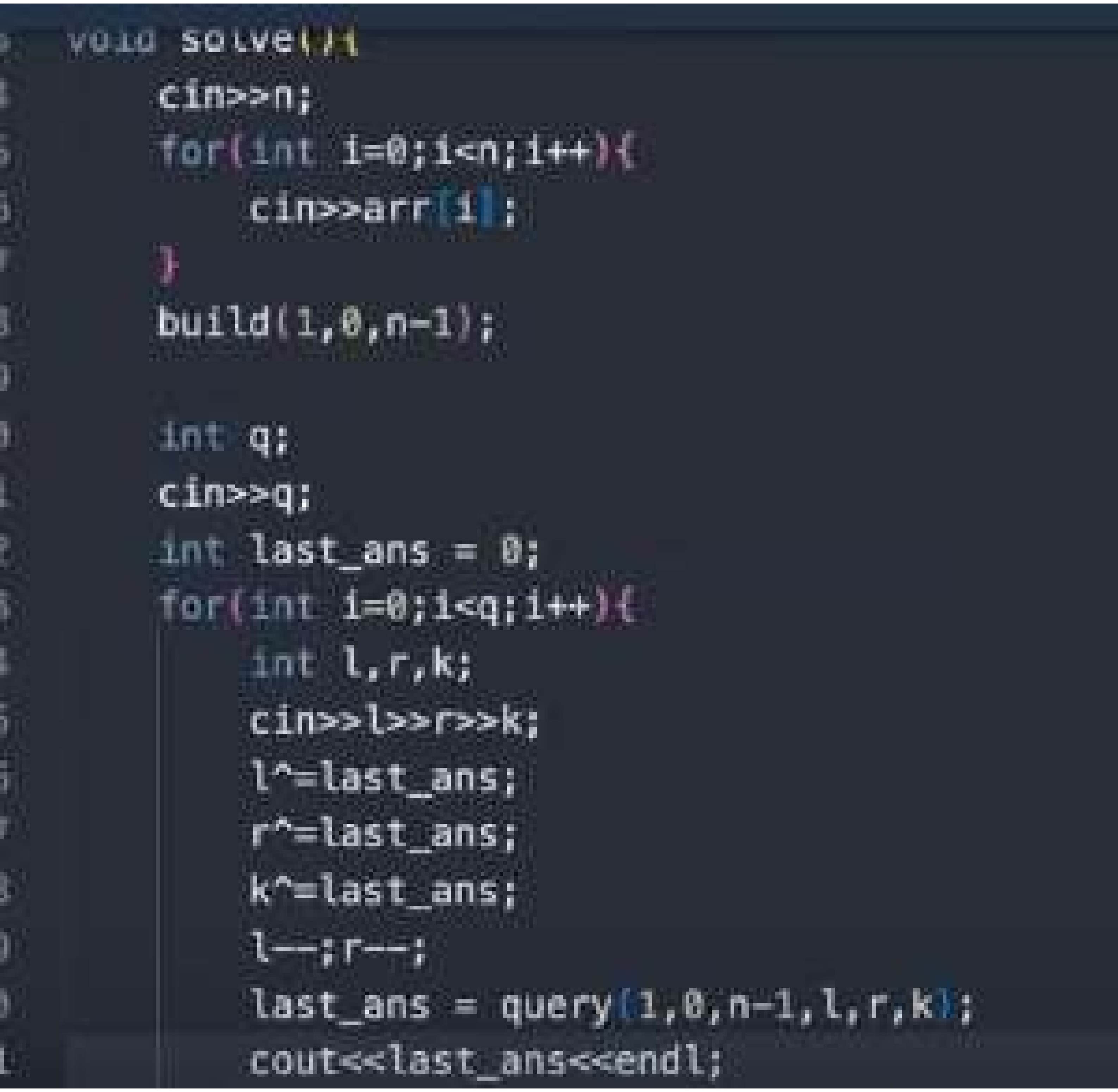
Build a segment Tree where each node stores a sorted list of its range, and answer each query by combining binary search with segment tree traversal to count elements greater than K



```
vector<int> t[400400];
void build(int id,int l,int r){
    if(l==r){
        t[id]={arr[l]};
        return;
    }
    int mid = (l+r)/2;
    build(id<<1,l,mid);
    build(id<<1|1,mid+1,r);
    t[id].resize(r-l+1);
    merge(t[id<<1].begin(),t[id<<1].end(),
        t[id<<1|1].begin(),t[id<<1|1].end(),
        t[id].begin());
}

int query(int id,int l,int r,int lq,int rq,int k){
    if(l>rq||lq>r)return 0;
    if(lq<=l&&r<=rq){
```



```
int query(int id,int l,int r,int lq,int rq,int k){
    if(l>rq||lq>r)return 0;
    if(lq<=l&&r<=rq){
        return t[id].end()- upper_bound(t[id].begin(),t[id].
        end(),k);
    }
    int mid = (l+r)/2;
    return query(id<<1,l,mid,lq,rq,k) + query(id<<1|1,mid+1,
    r,lq,rq,k);
}
```



```
void solve(){
    cin>>n;
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    build(1,0,n-1);

    int q;
    cin>>q;
    int last_ans = 0;
    for(int i=0;i<q;i++){
        int l,r,k;
        cin>>l>>r>>k;
        l^=last_ans;
        r^=last_ans;
        k^=last_ans;
        l--;r--;
        last_ans = query(1,0,n-1,l,r,k);
        cout<<last_ans<<endl;
```