Policy Based Data Structures

Question

Given q queries, of 3 types

1. Insert a number X into the set
2. Print the K’th element
3. Print the number of elements less than K

Policy Based Data Structures

- Ordered Set
- Based on Red Black Trees
- Included in

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
```

Ordered Set

```
template<class T> using oset = tree<T, null_type, less<T>,
rb_tree_tag, tree_order_statistics_node_update>;
```

- null_type: It is the mapped policy. It is null here to use it as a set. If we want to get map but not the set, as the second argument type must be used mapped type.
- less: It is the basis for comparison of two functions. Use less_equal for ordered multiset.
- rb_tree_tag: type of tree used.
- tree_order_statistics_node__update: It is included in tree_policy.hpp and contains various operations for updating the node variants of a tree-based container, so we can keep track of metadata like the number of nodes in a subtree.

Code

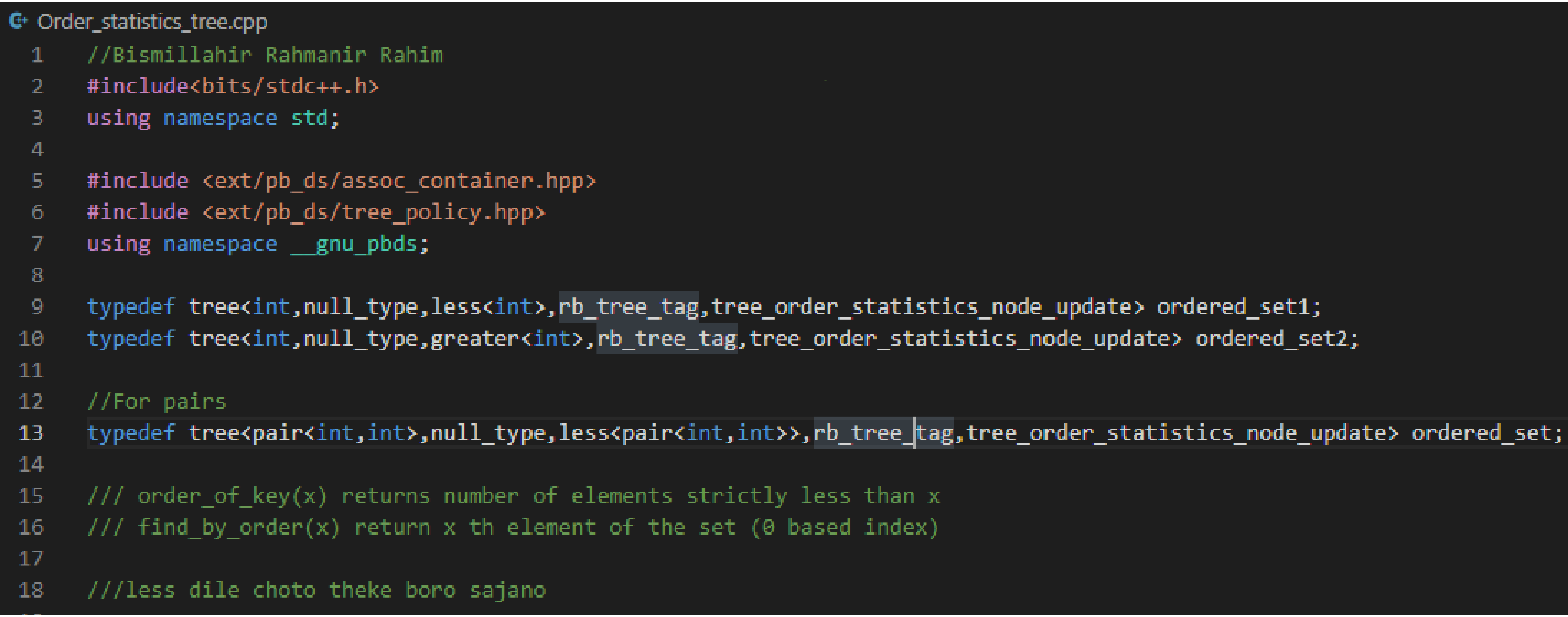
It has two key functions:

- order_of_key(k): Number of items strictly smaller than k
- find_by_order(k): K-th element in a set (counting from zero)

Ordered Set

```
template<class T> using oset = tree<T, null_type, less<T>,
rb_tree_tag, tree_order_statistics_node_update>;
```

- Normal set operations
- Random Access in O(Log N)
    - Find K’th largest element in the set
- Number of Items less than K in O(Log N)



```cpp
//Bismillahir Rahmanir Rahim
#include<bits/stdc++.h>
using namespace std;

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

typedef tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_update> ordered_set1;
typedef tree<int,null_type,greater<int>,rb_tree_tag,tree_order_statistics_node_update> ordered_set2;

//For pairs
typedef tree<pair<int,int>,null_type,less<pair<int,int>>,rb_tree_tag,tree_order_statistics_node_update> ordered_set;

/// order_of_key(x) returns number of elements strictly less than x
/// find_by_order(x) return x th element of the set (0 based index)

///less dile choto theke boro sajano
```