Day 1 - Standard Template Libraries (STL) in C++

1. **Sequence Containers**
    1. **vector**
    2. **array**
    3. **list**
    4. **forward_list**
2. **Associative Containers**
    1. **set / multiset**
    2. **map / multimap**
3. **Container Adaptors**
    1. **stack**
    2. **queue**
    3. **deque**
    4. **priority_queue**

1. vector

```
#include <vector>
vector<Type> v;

v.push_back(val);       // O(1)
v.pop_back();           // O(1)
v.insert(pos,val)       // O(n+m), insert before pos(iterator)

v.erase(pos);           // O(n)
v.erase(first,last);    // O(n), delete [first,last)
v.clear();              // O(n)

v[0];                   // O(1)
v.at(2);                // O(1)
v.front(), v.back();    // O(1)

v.begin(), v.end();     // O(1), return iterator

v.empty();              // O(1)
v.size();               // O(1)
```

2. set / multiset / unordered_set

```
#include <set>
set<Type> s;            // Binary Search Tree
multiset<Type> s;       // Binary Search Tree
unordered_set<Type> s; // Hash Table

s.insert(val);          // O(log n) / O(log n) / O(1)
                        // return pair<iterator,bool>, bool -> exist

s.erase(val);           // O(log n) / O(log n) / O(1)
s.erase(iterator)       // O(1)     / O(1)     / O(1)
s.find(val);            // O(log n) / O(log n) / O(1)
                        // return iterator, or s.end() if not found

s.clear();              // O(n)     / O(n)     / O(n)
s.empty();              // O(1)     / O(1)     / O(1)
s.size();               // O(1)     / O(1)     / O(1)

s.begin();              // O(1)     / O(1)     / O(1)
s.end();                // O(1)     / O(1)     / O(1)

set<Type>::iterator it;// iterator all the elements
for (it=s.begin();it!=s.end();++it)
     cout<<*it;
```

3. map / multimap / unordered_map

```
#include <map>
map<Type,Type> m;           // Binary Search Tree
multimap<Type,Type> m;      // Binary Search Tree
unordered_map<Type,Type> m; // Hash Table

m[val1];                // O(log n) / O(log n) / O(1)
                        // read and write

m.size();               // O(1)     / O(1)     / O(1)
m.empty();              // O(1)     / O(1)     / O(1)
m.clear();              // O(n)     / O(n)     / O(n)

m.find(Val1);           // O(log n) / O(log n) / O(1)
                        // return iterator, m.end() if not found
m.erase(iterator);      // O(1)     / O(1)     / O(1)
m.erase(val);           // O(log n) / O(log n) / O(1)
M.erase(first,last);    // O(m)     / O(m)     / O(m)
                        // delete [first,last) (iterator)

s.begin();              // O(1)     / O(1)     / O(1)
s.end();                // O(1)     / O(1)     / O(1)

map<Type,Type>::iterator it;    // iterator all the elements
for (it=m.begin();it!=m.end();++it)
     cout<<it->first<<it->second
```

4. stack

```
#include <stack>
stack<Type> s;

s.push(val);                // O(1)
s.top();                    // O(1)
s.pop();                    // O(1)

s.size();                   // O(1)
s.empty();                  // O(1)
```

5. queue

```
#include <queue>
queue<Type> q;

q.push(val);                // O(1)
q.pop();                    // O(1)

q.front();                  // O(1)
q.back();                   // O(1)

q.size();                   // O(1)
q.empty();                  // O(1)
```

6. deque

```
#include <deque>
deque<Type> d;

d.push_back(val);           // O(1)
d.push_front(val);          // O(1)
d.pop_back();               // O(1)
d.pop_front();              // O(1)

d.front();                  // O(1)
d.back();                   // O(1)
d[loc]                      // O(1)
                            // read and write

d.size();                   // O(1)
d.empty();                  // O(1)
d.clear();                  // O(n)

d.begin();                  // O(1)
d.end();                    // O(1)
```

7. priority_queue

```
#include <queue>        // Heap
priority_queue<int> q;                                  // largest on top
priority_queue<int,vector<int>,greater<int> > q;   // smallest on top
// useful hints for IO:
priority_queue<pair<int,int>,vector<pair<int,int> >,
                              greater<pair<int,int> > > q;
priority_queue<pair<int,pair<int,int> >,
               vector<pair<int,pair<int,int> > >,
               greater<pair<int,pair<int,int> > > > q;
// need space between ">", or compiler considers it >> (right-shift)

q.push(2);              // O(log n)
q.top();                // O(1)
q.pop();                // O(log n)

q.size();               // O(1)
q.empty();              // O(1)
```