Maps in cpp

1. maps store values in sorted order as per keys
2. we cannot use it+1 ; we have to use ++it
3. for size of map ⇒ m.size()
4. keys are unique ⇒ if existing key is modified, it stores new value

```
map<int, string> m;
```

Find

```
auto m[1] = "abc";
m[2] = "cdc";
auto it = m.find(3);
if (it == m.end()) cout << "No value";
else cout << (*it).first << " " << (*it).second;
```

Erase

```
m.erase(3);   → but (3, "aa") may not be present

auto it = m.find(7);
if(it != m.end()) m.erase(it);
```

Q. Print frequency of given strings in lexicographical order.

8
abc
def
abc
ghj
jkl
ghj
ghj
abc

```
map<string, int> m;
int n; cin >> n;
while (n--)
{
string s; cin >> s;
 m[s]++;
}
for (auto pr : m)
{
cout << pr.first << ":" << pr.second << endl;
}
```

Teacher's Signature

Unordered Maps

1. Has better time complexity O(1) rather than O(log n) in map
2. m.find(_) & m.erase(_) work exactly same
3. Uses hash table for pair

Q Given N strings and Q queries. In each query you are given a string. Print frequency of that string.

A
```
int main()
{
unordered_map<string, int> m;
int n; cin>>n;
while (n--)
{
for int i=0; i<n; i string s; cin>>s;
m[s]++;
}
int q;
cin>>q;
while (q--)
{
string s;
cin>>s;
cout<<m[s]<<endl;
}
}
```

I/ 8 abc def abc ghj jkl ghj ghj abc
2 abc ghj

O/
2
3

Teacher's Signature

Sets in C++

1. similar to corresponding map (time complexity O(log n))

```
set<string> s;
s.insert("abc");
s.insert("zsdf");
s.insert("bcd");
```

To insert

To Print

```
for(auto it = s.begin(); it != s.end(); ++it)
    cout << (*it) << endl;
```

To find

```
auto it = s.find("bcd");
if(it != s.end())
    cout << (*it);
```

To delete

```
auto it = s.find("abc");
if(it != s.end())
    s.erase(it);
```

/ `s.erase("abc");`

Unordered Sets → time complexities O(1)

Q Given N strings and Q queries. Print yes if string present else no

A

```
unordered_set<string> s;
int n; cin >> n;
for(int j=0; i<n; i++)
{
string str; cin >> str; s.insert(str);
}
int q; cin >> q;
while(q--)
{
string str; cin >> str;
if(s.find(str) == s.end()) cout << "no" << endl;
else cout << "Yes" << endl;
}
```

Teacher's Signature

Scanned with OKEN Scanner

Pairs & Vectors

Pair

```
pair<int,int> p[3];
p[0] = {1,2};
p[1] = {2,3};
p[2] = {3,4};
```

```
cin >> p[0].first;
cin >> p[0].second;
```

SWAP ⇒ swap(p[0], p[1]);

Vectors

to initialise with specific value ⇒ vector<int> v(10,3)

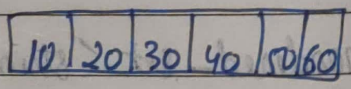
do pop back — v.pop_back();

Copy vector — vector<int> v2 = v;

If you want to pass vector to function and do changes in it, go give reference.

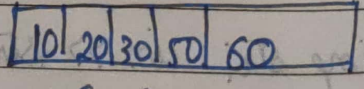
(vector<int> &v)

do erase element



v.erase(v.begin()+3, v.begin()+4)

1



Sort / R Sort

sort(v.begin(), v.end());

rsort(v.rbegin(), v.rend());

Deque

→ Implements linked list

→ it is better to insert in deques than vector (O(1) is better than O(n))

auto it = dq.begin(); it++; dq.insert(it, 20); {10, 5, 20, 13} → now it will point to 20

insert from end/front

deque<int> d;
d.push_back(1);
d.push_front(2);
d.push_front(3);

O(1)

3 | 2 | 1

remove from front/back

d.pop_front(); 2 | 1
d.pop_back(); 3 | 2

element at i^{th} position

int c = d.at(i);
dq.front() → first element
dq.back() → last element

delete element

similar to vector (give pointer to erase)

Algorithms

① reverse a string ⇒ reverse(s.begin(), s.end());

② to rotate anti-clockwise ⇒ rotate(v.begin(), v.begin()+1, v.end());

1367 → 3671

③ greatest common divisor ⇒ __gcd(a, b);

④ to get min/max element of vector (if we want from other order then i)

int mini = *min_element(v.begin(), v.end());

⑤ to get sum of array ⇒ sum = accumulate(v.begin(), v.end(), 0); (0 → initial sum)

⑥ to count ct = count(v.begin(), v.end(), 6); (6 → what you wanna count)

⑦ to find auto it = v.find(v.begin(), v.end(), 5); (it → iterator, 5 → what find)

if not found it == v.end();

⑧ to reverse string/vector ⇒ reverse(s.begin(), s.end());

Teacher's Signature

Scanned with OKEN Scanner

Vector of Pair

```
vector<pair<int, int>> v;
int n; cin >> n;
for (int i=0; i<n; i++)
{
int x, y; cin >> x >> y;
v.push_back({x, y});
}
for (int i=0; i<n; i++)
{
cout << v[i].first << " " << v[i].second << endl;
}
```

for size n

to print