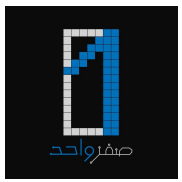
Copmpetitive Programming

Session 4

01

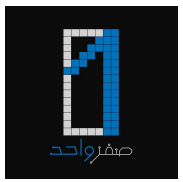


March 2017

Recap

What do we know so far?

- Stack
- Queue
- Vector
- Linked List

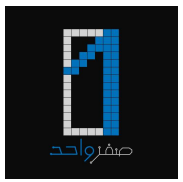


C++11

Here are the most commonly used features in C++11

- Type inference
- Initializer list
- Range-based for loop

They sound scary, but mostly simple to use and will make your life much easier.

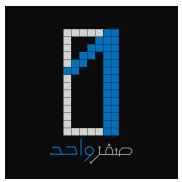


Problem 1

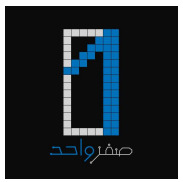
Sort numbers, evens non-decreasing then odds non-increasing

input
4
2 5 6 3
output
2 6 5 3

input
5
1 3 5 4 4
output
4 4 5 3 1

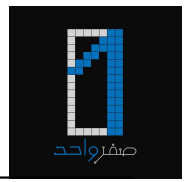


Using compare functions



Sort with compare function

```
vector<int>v;

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

// function to sort numbers in non-decreasing order
bool cmp(int first, int second){
  if(first <= second){
    // order is ok, don't change it
    return true;
  }else{
    //order is wrong, change it
    return false;
  }
```

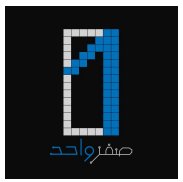


Problem 2

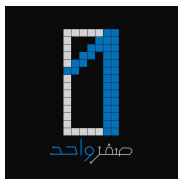
Create a phonebook.
Add name and phone number, Get phone number of a person.

input
2 (number of addtions)
amr 1234
thabet 567890
2 (number of queries)
thabet
amr
output
567890
1234

input
3
a 123
b 456
c 789
3
a
b
d
output
123
456
doesn't exist

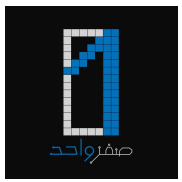


Map to the rescue

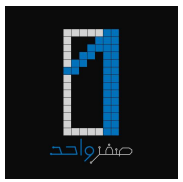


Map syntax

```
#include <map>
map<string, string>mp;

mp["amr"]="123"; // inserts {"amr", "123"} into mp
cout<<mp["amr"]; // outputs "123"
```

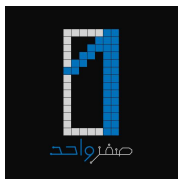


Internals of Map



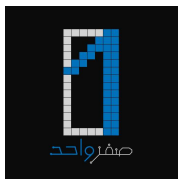
More map syntax

```
#include <map>
map<int, int>mp;

if(mp.find(1)!=mp.end()) cout<<"one found"<<endl;
cout<<mp[1]<<endl; // {1,0} will be added to mp, and
    will output 0
if(mp.find(1)!=mp.end()) cout<<"one found"<<endl;
```



Even more map syntax

```
#include <map>
map<int, int>mp;

// print all elements of a map
for(auto i:mp){ // i is pair<int,int>
  cout<<i.first<<" "<<i.second<<endl;
}
```



Problem 3

Output unique numbers only.

input

7

3 1 2 3 4 5 5

output

1 2 3 4 5

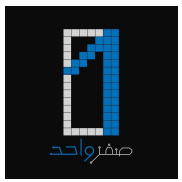
input

4

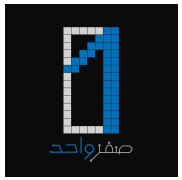
1 1 1 1

output

1

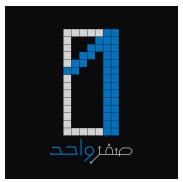


Set to the rescue

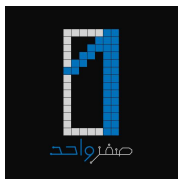


Set syntax

```
#include <set>
set<int>st;

st.insert(2); // st has {2}
st.insert(2); // st still has {2}
st.insert(1); // st has {1, 2}
```

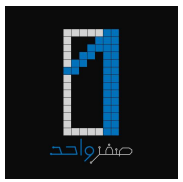


Internals of set



Final notes

- Midterm Vacation.
- Self study prioirity queue



Thank you