Competitive Coding

Algorithm

```
#include <algorithm>
find(Start_location,
End_location, key) auto id =
find(arr,arr+n,key);
```

C++

```
bool compare(typr a, type
b) {return a>b;}
sort(arr,arr+n,compare);
//type can be
//string/integer/char/doub
```

C++

```
bool present = binary_search(arr,arr+n,key); auto lb=
lower_bound(arr,arr+n,key); //lower bound first occurence of
element >= the key auto ub= upper_bound(arr,arr+n,key); //upper
bound first occurence of element > the key index=lb-arr;
```

C++

String

String initialization

```
#include<string> string s0;
string s1("Hello World!");
string s2="hello World";
string s3(s2); string s4=s3;
char a[]=
{'a','b','c','d','\\n'};\
string s5(a);
```

C++

String function

```
s.empty() bool
s.append("str") || s+="str"
s.clear() || s.length()
s1.compare(s2) || s1<s2 s[0]
|| s.find("apple")
s.erase(st_index,length to
remove) for(int
i=0;i<s.length();i++)
cout<<s[i]<<":"; for(char
c:s0) cout<<c;
```

C++

String tokenization

```
#include<cstring> char arr[s.length()+1]; strcpy(arr,s.c_str());
for(auto it=strtok(arr," ");it!=NULL;it=strtok(NULL," "))
cout<<it;
```

C++

Vector

Vector Initialization

```
#include <vector>
vector<int> a; vector<int>
b(5,10); //array having 5
elements of value 0;
vector<int>
c(b.begin(),b.end());
vector<int> d{1,2,3,4,5,6};
```

C++

Vector treversal

```
for(int i=0;i<c.size();i++)
cout<<c[i]<<" "; for(int
i:d) cout<<i; for(auto
i=d.begin();d!=d.end();i++)
cout<<*i<<" "; //can use
vector<int> instead auto
```

C++

Some general function

```
d.push_back(10); d.pop_back();
d.insert(d.begin()+3,4,100); //
insert 100 4 times
d.erase(d.begin()+2,d.begin()+5
//remove from 2 to 5
```

C++

```
d.empty();// is empty bool
d.front(); //first element
d.back(); //last element
d.reserve(1000); // capacity
1000 space reserved v.size()
v.capacity() v.max_size()
```

C++

Append 1 vector in another

```
vector<int> v1{ 10, 20, 30, 40, 50 }; vector<int> v2{ 100, 200,
300, 400 }; //appending elements of vector v2 to vector v1
v1.insert(v1.end(), v2.begin(), v2.end());
```

C++

LIST

```
#include <list> list<string> s{"apple","mango","banana"};
push_back || push_front || pop_back || pop_front insert || erase
|| remove || reverse || sort
```

C++

Traversing

```
list <int> :: iterator it; auto
it = g.begin(); for(it =
g.begin(); it != g.end(); ++it)
cout << *it; for(string ss:s)
cout<<ss<<endl;
```

C++

```
//traverse in // reverse
order auto j=v.end(); j-
-; for(;*j>=i;j--)
```

C++

General Functions

```
s.remove("apple"); //remove
all instances of apple auto
it = s.begin(); it++; it++;
s.erase(it); // remove 3rd
element
```

C++

```
auto it = s.begin(); it++;
it++; s.insert(it,"the new
string"); //insert at 3rd
position
```

C++

Other Data Structure

```
stack<int> st; st.push(10);
while (!st.empty())
st.top(); st.pop();
```

C++

```
queue<int> st; st.push(10);
while (!st.empty())
st.front(); st.pop();
```

C++

```
priority_queue<int> pqx; // max heap
priority_queue<int,vector<int>,greater<int>> pq; pq.push(no);
while(!pq.empty()) cont<<pq.top(); pq.pop();
```

C++

```
priority_queue<Person,vector<Person>, comparatorPerson> Q; class
comparatorPerson { public : bool operator()(Person p1, Person p2)
{ // this will return true when second person // has greater
height. Suppose we have p1.height=5 // and p2.height=5.5 then the
object which // have max height will be at the top(or // max
priority) return p1.height < p2.height; } };
```

C++

```
map<int, int> m; m[key]=value //insert values // check if key
exists if(m.find(key)!=m.end()) element exists; map<int,
int>::iterator itr; for (itr = gquiz1.begin(); itr !=
gquiz1.end(); ++itr) cout<< itr->first; cout<< itr->second; num =
gquiz2.erase(4);//remove 4 from the map unordered_map<string, int>
umap; // rest is same import map
```

C++