Vector

Function	**Explanation**	**Syntax**	**Complexity**
1) push_back	insert an element at the end of the vector	v.push_back(5)	O(1)
2) pop_back	remove the last element of the vector	v.pop_back()	O(1)
3) front	get the first element of the vector	v.front()	O(1)
4) back	get the last element of the vector	v.back()	O(1)
5) empty	check if the vector is empty	v.empty()	O(1)
6) clear	remove all the elements of the vector	v.clear()	O(n)
7) size	get the number of elements of the vector	v.size()	O(1)
8) assign	assign new values to the vector	v.assign(10,2)	O(n)
9) resize	change the size of the vector	v.resize(5) or v.resize(5,2)	O(n)
10) begin	iterator to the first element	v.begin()	O(1)
11) end	iterator to the element after the last	v.end()	O(1)

12) –end	iterator to the last element	–v.end()	O(1)
13) reverse	reverse the elements of the vector	reverse(v.begin(),v.end())	O(n)
14) sort	sort the elements of the the vector in ascending order	sort(v.begin(),v.end())	O(n log n)
15) min_element	iterator to the minimum element	min_element(v.begin(),v.end())	O(n)
16) max_element	iterator to the maximum element	max_element(v.begin(),v.end())	O(n)

String

Function	**Explanation**	**Syntax**	**Complexity**
push_back	insert an element at the end	s.push_back(‘p’)	O(1)
pop_back	remove the last element	s.pop_back()	O(1)
front	get the first character	s.front()	O(1)
back	get the last character	s.back()	O(1)
empty	check if the string is empty	s.empty()	O(1)
clear	remove all characters from the string	s.clear()	O(n)

substr	get a substring from the string	s.substr(1,3) or s.substr(3)	O(m)

Map

Function	**Explanation**	**Syntax**	**Complexity**
1) begin	iterator to the first element	mp.begin()	O(1)
2) end	iterator to the element after the last	mp.end()	O(1)
3) –end	iterator to the last element	–mp.end()	O(1)
4) insert	insert an element into the map	mp.insert({25,rahim}) or mp[25]=rahim	O(log n)
5) erase	remove an element which is binded with that key	mp.erase(25)	O(log n)
6) find	find an element on the basis of key	mp.find(25)	O(log n)
7) size	get the number of elements of the map	mp.size()	O(1)
8) empty	check if the map is empty	mp.empty()	O(1)
9) clear	remove all the elements from the map	mp.clear()	O(n)
10) lower_bound	get an iterator to the first element which is greater or equal to x	mp.lower_bound(x)	O(log n)

11) upper_bound	get an iterator to the first element which is strictly greater than x	mp.upper_bound(x)	O(log n)

Set & Multiset

Function	Explanation	Syntax	Complexity
1) begin	iterator to the first element	s.begin()	O(1)
2) end	iterator to the element after the last	s.end()	O(1)
3) –end	iterator to the last element	–s.end()	O(1)
4) insert	insert an element into the set	s.insert(5)	O(log n)
5) erase	remove an element from the set	s.erase(5)	O(log n)
6) find	find an element from the set	s.find(5)	O(log n)
7) size	get the number of elements of the set	s.size()	O(1)
8) empty	check if the set is empty	s.empty()	O(1)
9) clear	remove all the elements of the set	s.clear()	O(n)
10) lower_bound	get an iterator to the first element which is greater or equal to x	s.lower_bound(x)	O(log n)
11) upper_bound	get an iterator to the first element which is strictly greater than x	s.upper_bound(x)	O(log n)

Priority Queue

Function	**Explanation**	**Syntax**	**Complexity**
1) push	insert an element	pq.push(5)	O(log n)
2) pop	remove the top element	pq.pop()	O(log n)
3) top	get the top element	pq.top()	O(1)
4) size	get the number of elements of the priority queue	pq.size()	O(1)
5) empty	check if the priority queue is empty	pq.empty()	O(1)