**Standard Template Library (C++)**

**Topic List:**

* **Vector**
* **2D Vector**
* **String**
* **Pair**
* **Map**
* **Multimap**
* **Set**
* **Multiset**
* **Stack**
* **Queue**
* **Deque**
* **Priority Queue**
* **Template**

***Md. Shajibul Islam.***

***East West University (CSE-19)***

**Source: *Geeks For Geeks, CPS Academy.***

**1D Vector:**

1. **#include<bits/stdc++.h>**
2. **using namespace std;**
3. **int main ()**
4. **{**
5. **vector<int> v;**
7. **v.push\_back( 1 );**
8. **v.push\_back( 2 );**
9. **v.push\_back( 3 );**
11. **cout << v[0] << " " << v[1] << " " << v[2] << endl; /// 1 2 3**
13. **v[1] = 3;**
14. **cout << v[0] << " " << v[1] << " " << v[2] << endl; /// 1 3 3**
16. **cout << v.size() << endl; /// 3**
17. **for ( int i = 0; i < v.size(); i++ ) cout << v[i] << " "; /// 1 3 3**
18. **cout << endl;**
20. **vector <int> v1 = { 2, 3, 4 };**
22. **cout << v1.size() << endl; /// 3**
23. **for ( int i = 0; i < v1.size(); i++ ) cout << v1[i] << " "; /// 2 3 4**
24. **cout << endl;**
26. **v.clear();**
27. **cout << v.size() << endl; /// 0**
28. **cout << v.empty() << endl; /// 1**
29. **cout << v1.empty() << endl; /// 0**
31. **v1.resize(5);**
32. **cout << v1.size() << endl; /// 5**
33. **for ( int i = 0; i < v1.size(); i++ ) cout << v1[i] << " "; /// 2 3 4 0 0**
34. **cout << endl;**
36. **vector<int> a(5);**
38. **cout << a.size() << endl; /// 5**
39. **for ( int i = 0; i < a.size(); i++ ) cout << a[i] << " "; /// 0 0 0 0 0**
40. **cout << endl;**
42. **a = v1;**
44. **for ( auto u : a ) cout << u << " "; /// 2 3 4 0 0**
45. **cout << endl;**
47. **vector<int>::iterator it;**
48. **for ( it = a.begin(); it != a.end(); it++ ) cout << \*it << " "; /// 2 3 4 0 0**
49. **cout << endl;**
51. **a = { 3, 4, 5, 1, 2 };**
53. **sort ( a.begin(), a.end() ); ///O(n\*log2(n))**
55. **for ( auto u : a ) cout << u << " "; /// 1 2 3 4 5**
56. **cout << endl;**
58. **sort ( a.rbegin(), a.rend() );**
60. **for ( auto u : a ) cout << u << " "; /// 5 4 3 2 1**
61. **cout << endl;**

64. **a = { 3, 4, 5, 1, 2 };**
65. **sort ( a.begin(), a.end(), greater<int>() );**
67. **for ( auto u : a ) cout << u << " "; /// 5 4 3 2 1**
68. **cout << endl;**
70. **a = { 3, 4, 5, 1, 2 };**
72. **reverse( a.begin(), a.end() );**
74. **for ( auto u : a ) cout << u << " "; /// 2 1 5 4 3**
75. **cout << endl;**
77. **cout << a.back() << endl; /// 3**
78. **a.pop\_back(); /// O(1) complexity.**
79. **cout << a.back() << endl; /// 4**

82. **a = { 3, 4, 5, 1, 2 };**
83. **cout << \*a.begin() << endl; /// 3**
85. **a.erase( a.begin() ); /// O(n) complexity.**
86. **for ( auto u : a ) cout << u << " "; /// 4 5 1 2**
87. **cout << endl;**
89. **a.erase( a.begin()+2 );**
90. **for ( auto u : a ) cout << u << " "; /// 4 5 2**
91. **cout << endl;**
93. **a = { 1, 1, 2, 2, 2, 3, 3 };**
94. **unique( a.begin(), a.end() );**
96. **for ( auto u : a ) cout << u << " "; /// 1 2 3 2 2 3 3**
97. **cout << endl;**

100. **a = { 1, 1, 2, 2, 2, 3, 3 };**
101. **int n = unique( a.begin(), a.end() ) - a.begin();**
103. **cout << n << endl; /// 3**
104. **for ( int i = 0; i < n; i++ ) cout << a[i] << " "; /// 1 2 3**
105. **cout << endl;**
107. **a = { 2, 3, 1, 5 };**
108. **cout << max\_element( a.begin(), a.end() ) - a.begin() << endl; /// 3**
109. **cout << \*max\_element( a.begin(), a.end() ) << endl; /// 5**
111. **return 0;**
112. **}**
113. **// C++ program to illustrate the**
114. **// element accesser in vector**
115. **#include <bits/stdc++.h>**
116. **using namespace std;**
118. **int main()**
119. **{**
120. **vector<int> g1;**
122. **for (int i = 1; i <= 10; i++)**
123. **g1.push\_back(i \* 10);//10,20,30,40……100**
125. **cout << "\nReference operator [g] : g1[2] = " << g1[2];//30**
127. **cout << "\nat : g1.at(4) = " << g1.at(4);//50**
129. **cout << "\nfront() : g1.front() = " << g1.front();//10**
131. **cout << "\nback() : g1.back() = " << g1.back();//100**
133. **// pointer to the first element**
134. **int\* pos = g1.data();**
136. **cout << "\nThe first element is " << \*pos;//10**
137. **return 0;**
138. **}**

**Output:**

**Reference operator [g] : g1[2] = 30**

**at : g1.at(4) = 50**

**front() : g1.front() = 10**

**back() : g1.back() = 100**

**The first element is 10**

1. **// C++ program to illustrate the**
2. **// Modifiers in vector**
3. **#include <bits/stdc++.h>**
4. **#include <vector>**
5. **using namespace std;**
7. **int main()**
8. **{**
9. **// Assign vector**
10. **vector<int> v;**
12. **// fill the array with 10 five times**
13. **v.assign(5, 10);**
15. **cout << "The vector elements are: ";**
16. **for (int i = 0; i < v.size(); i++)**
17. **cout << v[i] << " ";**
19. **// inserts 15 to the last position**
20. **v.push\_back(15);**
21. **int n = v.size();**
22. **cout << "\nThe last element is: " << v[n - 1];**
24. **// removes last element**
25. **v.pop\_back();**
27. **// prints the vector**
28. **cout << "\nThe vector elements are: ";**
29. **for (int i = 0; i < v.size(); i++)**
30. **cout << v[i] << " ";**
32. **// inserts 5 at the beginning**
33. **v.insert(v.begin(), 5);**
35. **cout << "\nThe first element is: " << v[0];**
37. **// removes the first element**
38. **v.erase(v.begin());**
40. **cout << "\nThe first element is: " << v[0];**
42. **// inserts at the beginning**
43. **v.emplace(v.begin(), 5);**
44. **cout << "\nThe first element is: " << v[0];**
46. **// Inserts 20 at the end**
47. **v.emplace\_back(20);**
48. **n = v.size();**
49. **cout << "\nThe last element is: " << v[n - 1];**
51. **// erases the vector**
52. **v.clear();**
53. **cout << "\nVector size after erase(): " << v.size();**
55. **// two vector to perform swap**
56. **vector<int> v1, v2;**
57. **v1.push\_back(1);**
58. **v1.push\_back(2);**
59. **v2.push\_back(3);**
60. **v2.push\_back(4);**
62. **cout << "\n\nVector 1: ";**
63. **for (int i = 0; i < v1.size(); i++)**
64. **cout << v1[i] << " ";**
66. **cout << "\nVector 2: ";**
67. **for (int i = 0; i < v2.size(); i++)**
68. **cout << v2[i] << " ";**
70. **// Swaps v1 and v2**
71. **v1.swap(v2);**
73. **cout << "\nAfter Swap \nVector 1: ";**
74. **for (int i = 0; i < v1.size(); i++)**
75. **cout << v1[i] << " ";**
77. **cout << "\nVector 2: ";**
78. **for (int i = 0; i < v2.size(); i++)**
79. **cout << v2[i] << " ";**
80. **}**

**Output:**

1. **The vector elements are: 10 10 10 10 10**
2. **The last element is: 15**
3. **The vector elements are: 10 10 10 10 10**
4. **The first element is: 5**
5. **The first element is: 10**
6. **The first element is: 5**
7. **The last element is: 20**
8. **Vector size after erase(): 0**
9. **Vector 1: 1 2**
10. **Vector 2: 3 4**
11. **After Swap**
12. **Vector 1: 3 4**
13. **Vector 2: 1 2**

**All Vector Functions :**

* [**vector::begin() and vector::end()**](https://www.geeksforgeeks.org/vectorbegin-vectorend-c-stl/)
* [**vector rbegin() and rend()**](https://www.geeksforgeeks.org/vector-rbegin-and-rend-function-in-c-stl/)
* [**vector::cbegin() and vector::cend()**](https://www.geeksforgeeks.org/vector-cbegin-vector-cend-c-stl/)
* [**vector::crend() and vector::crbegin()**](https://www.geeksforgeeks.org/vectorcrend-vectorcrbegin-examples/)
* [**vector::assign()**](https://www.geeksforgeeks.org/vector-assign-in-c-stl/)
* [**vector::at()**](https://www.geeksforgeeks.org/vectorat-vectorswap-c-stl/)
* [**vector::back()**](https://www.geeksforgeeks.org/vectorfront-vectorback-c-stl/)
* [**vector::capacity()**](https://www.geeksforgeeks.org/vector-capacity-function-in-c-stl/)
* [**vector::clear()**](https://www.geeksforgeeks.org/vectorclear-vectorerase-c-stl/)
* [**vector::push\_back()**](https://www.geeksforgeeks.org/vectorpush_back-vectorpop_back-c-stl/)
* [**vector::pop\_back()**](https://www.geeksforgeeks.org/vectorpush_back-vectorpop_back-c-stl/)
* [**vector::empty()**](https://www.geeksforgeeks.org/vectorempty-vectorsize-c-stl/)
* [**vector::erase()**](https://www.geeksforgeeks.org/vectorclear-vectorerase-c-stl/)
* [**vector::size()**](https://www.geeksforgeeks.org/vectorempty-vectorsize-c-stl/)
* [**vector::swap()**](https://www.geeksforgeeks.org/vectorat-vectorswap-c-stl/)
* [**vector::reserve()**](https://www.geeksforgeeks.org/using-stdvectorreserve-whenever-possible/)
* [**vector::resize()**](https://www.geeksforgeeks.org/vector-resize-c-stl/)
* [**vector::shrink\_to\_fit()**](https://www.geeksforgeeks.org/vector-shrink_to_fit-function-in-c-stl/)
* [**vector::operator=**](https://www.geeksforgeeks.org/vectoroperator-vectoroperator-c-stl/)
* [**vector::operator[]**](https://www.geeksforgeeks.org/vectoroperator-vectoroperator-c-stl/)
* [**vector::front()**](https://www.geeksforgeeks.org/vectorfront-vectorback-c-stl/)
* [**vector::data()**](https://www.geeksforgeeks.org/vector-data-function-in-c-stl/)
* [**vector::emplace\_back()**](https://www.geeksforgeeks.org/vectoremplace_back-c-stl/)
* [**vector::emplace()**](https://www.geeksforgeeks.org/vector-emplace-function-in-c-stl/)
* [**vector::max\_size()**](https://www.geeksforgeeks.org/vector-max_size-function-in-c-stl/)
* [**vector::insert()**](https://www.geeksforgeeks.org/vector-insert-function-in-c-stl/)

**2D Vector:**

1. **#include<bits/stdc++.h>**
2. **using namespace std;**
3. **#define FAST             ios\_base::sync\_with\_stdio(0); cin.tie(0); cout.tie(0)**
4. **int main()**
5. **{**
6. **FAST;**
7. **//input output a 2D vector with sort.**
9. **int row,col;cin>>row>>col;**
10. **vector<vector<int>>v(row,vector<int>(col));**
12. **//input part..**
13. **//row = 3, col = 3**
14. ***/\****
15. ***3 3***
16. ***3 2 1***
17. ***6 5 4***
18. ***9 8 7***
19. ***\*/***
20. **for(int i=0;i<v.size();i++)//v.size()==n**
21. **{**
22. **for(int j=0;j<v[i].size();j++)//v[i].size()==m**
23. **{**
24. **cin>>v[i][j];**
25. **}**
26. **sort(v[i].begin(),v[i].end());**
27. **}**
29. **//output part..**
30. ***/\****
31. ***1 2 3***
32. ***4 5 6***
33. ***7 8 9***
34. ***\*/***
36. **cout<<"The Sorted vector is: "<<endl;**
37. **for(int i=0;i<v.size();i++)//v.size()==n**
38. **{**
39. **for(int j=0;j<v[i].size();j++)//v[i].size()==m**
40. **{**
41. **cout<<v[i][j]<<" ";**
42. **}**
43. **cout<<endl;**
44. **}**

47. **}**
48. **// C++ code to demonstrate sorting of**
49. **// 2D vector on basis of no. of columns**
50. **// in ascending order**
51. **#include<iostream>**
52. **#include<vector> // for 2D vector**
53. **#include<algorithm> // for sort()**
54. **using namespace std;**
56. **// Driver function to sort the 2D vector**
57. **// on basis of a no. of columns in**
58. **// ascending order**
59. **bool sizecom(const vector<int>& v1, const vector<int>& v2)**
60. **{**
61. **return v1.size() < v2.size();**
62. **}**
64. **int main()**
65. **{**
66. **// Initializing 2D vector "vect" with**
67. **// values**
68. **vector< vector<int> > vect{{1, 2},**
69. **{3, 4, 5},**
70. **{6}};**
72. **// Displaying the 2D vector before sorting**
73. **cout << "The Matrix before sorting is:\n";**
74. **for (int i=0; i<vect.size(); i++)**
75. **{**
76. **//loop till the size of particular**
77. **//row**
78. **for (int j=0; j<vect[i].size() ;j++)**
79. **cout << vect[i][j] << " ";**
80. **cout << endl;**
81. **}**
83. **//Use of "sort()" for sorting on**
84. **//basis of no. of columns in**
85. **//ascending order.**
86. **sort(vect.begin(), vect.end(), sizecom);**
88. **// Displaying the 2D vector after sorting**
89. **cout << "The Matrix after sorting is:\n";**
90. **for (int i=0; i<vect.size(); i++)**
91. **{**
92. **//loop till the size of particular**
93. **//row**
94. **for (int j=0; j<vect[i].size() ;j++)**
95. **cout << vect[i][j] << " ";**
96. **cout << endl;**
97. **}**
99. **return 0;**
101. **}**

**Output:**

**The Matrix before sorting is:**

**1 2**

**3 4 5**

**6**

**The Matrix after sorting is:**

**6**

**1 2**

**3 4 5**

1. **// C++ code to demonstrate sorting of a**
2. **// 2D vector on basis of a column**
3. **#include<iostream>**
4. **#include<vector> // for 2D vector**
5. **#include<algorithm> // for sort()**
6. **using namespace std;**
8. **// Driver function to sort the 2D vector**
9. **// on basis of a particular column**
10. **bool sortcol( const vector<int>& v1,**
11. **const vector<int>& v2 ) {**
12. **return v1[1] < v2[1];**
13. **}**
15. **int main()**
16. **{**
17. **// Initializing 2D vector "vect" with**
18. **// values**
19. **vector< vector<int> > vect{{3, 5, 1},**
20. **{4, 8, 6},**
21. **{7, 2, 9}};**
23. **// Number of rows;**
24. **int m = vect.size();**
26. **// Number of columns (Assuming all rows**
27. **// are of same size). We can have different**
28. **// sizes though (like Java).**
29. **int n = vect[0].size();**
31. **// Displaying the 2D vector before sorting**
32. **cout << "The Matrix before sorting is:\n";**
33. **for (int i=0; i<m; i++)**
34. **{**
35. **for (int j=0; j<n ;j++)**
36. **cout << vect[i][j] << " ";**
37. **cout << endl;**
38. **}**
40. **// Use of "sort()" for sorting on basis**
41. **// of 2nd column**
42. **sort(vect.begin(), vect.end(),sortcol);**
44. **// Displaying the 2D vector after sorting**
45. **cout << "The Matrix after sorting is:\n";**
46. **for (int i=0; i<m; i++)**
47. **{**
48. **for (int j=0; j<n ;j++)**
49. **cout << vect[i][j] << " ";**
50. **cout << endl;**
51. **}**
52. **return 0;**
53. **}**
54. **Output:**
55. **The Matrix before sorting is:**
56. **3 5 1**
57. **4 8 6**
58. **7 2 9**
59. **The Matrix after sorting is:**
60. **7 2 9**
61. **3 5 1**
62. **4 8 6**

**Here, in v[1] = 2nd row 🡪 2<5<8.**

**So, 3rd row then 1st row then 2nd row.**

1. **// C++ program to demonstrate sorting in vector**
2. **// of pair according to 2nd element of pair**
3. **#include<bits/stdc++.h>**
4. **using namespace std;**
6. **// Driver function to sort the vector elements**
7. **// by second element of pairs**
8. **bool sortbysec(const pair<int,int> &a,**
9. **const pair<int,int> &b)**
10. **{**
11. **return (a.second < b.second);**
12. **}**
14. **int main()**
15. **{**
16. **// declaring vector of pairs**
17. **vector< pair <int, int> > vect;**
19. **// Initialising 1st and 2nd element of pairs**
20. **// with array values**
21. **int arr[] = {10, 20, 5, 40 };**
22. **int arr1[] = {30, 60, 20, 50};**
23. **int n = sizeof(arr)/sizeof(arr[0]);**
25. **// Entering values in vector of pairs**
26. **for (int i=0; i<n; i++)**
27. **vect.push\_back( make\_pair(arr[i],arr1[i]) );**
29. **// Printing the original vector(before sort())**
30. **cout << "The vector before sort operation is:\n" ;**
31. **for (int i=0; i<n; i++)**
32. **{**
33. **// "first" and "second" are used to access**
34. **// 1st and 2nd element of pair respectively**
35. **cout << vect[i].first << " "**
36. **<< vect[i].second << endl;**
38. **}**
40. **// Using sort() function to sort by 2nd element**
41. **// of pair**
42. **sort(vect.begin(), vect.end(), sortbysec);**
44. **// Printing the sorted vector(after using sort())**
45. **cout << "The vector after sort operation is:\n" ;**
46. **for (int i=0; i<n; i++)**
47. **{**
48. **// "first" and "second" are used to access**
49. **// 1st and 2nd element of pair respectively**
50. **cout << vect[i].first << " "**
51. **<< vect[i].second << endl;**
52. **}**
53. **return 0;**
54. **}**

**Output:**

1. **The vector before applying sort operation is:**
2. **10 30**
3. **20 60**
4. **5 20**
5. **40 50**
6. **The vector after applying sort operation is:**
7. **5 20**
8. **10 30**
9. **40 50**
10. **20 60**

**String:**

1. **#include<bits/stdc++.h>**
2. **using namespace std;**
4. **int main ()**
5. **{**
6. **/// Declare string**
7. **string s;**
9. **/// Assign string**
10. **s = "abcdf";**
12. **/// Printing size of string**
13. **cout << s.size() << endl; /// 5**
15. **/// Printing string**
16. **cout << s << endl; /// abcdf**
18. **/// Pushing char back to a string**
19. **s += 'b';**
20. **s += 'c';**
21. **cout << s << endl; /// abcdfbc**
23. **/// Taking input string**
24. **cin >> s;**
25. **cout << s << endl;**
27. **s = "asdfgg";**
29. **/// Checking is a string empty or not**
30. **string s1;**
31. **cout << s.empty() << endl; /// 0**
32. **cout << s1.empty() << endl; /// 1**
34. **/// Assigning an string in another string variable**
35. **s1 = s;**
36. **s.clear();**
38. **cout << s.empty() << endl; /// 1**
39. **cout << s1.empty() << endl; /// 0**
41. **/// assigning 'k' in 0-th index**
42. **s = "asdfg";**
43. **s[0] = 'k';**
44. **cout << s << endl;  ///kasdfg;**
46. **s = "abc";**
47. **s1 = "def";**
49. **/// String concatenation**
50. **string tmp = s + s1;**
51. **cout << tmp << endl; /// abcdef**
53. **/// String iterator**
54. **string::iterator it;**
55. **for ( it = s.begin(); it != s.end(); it++ ) cout << \*it; /// abc**
56. **cout << endl;**
58. **/// For each loop**
59. **for ( auto c : s ) cout << c; /// abc**
60. **cout << endl;**

63. **s = "asd";**
64. **tmp = s;**
66. **/// Comparing two strings**
67. **if ( tmp == s ) cout << "Yes Match\n";**
68. **else "No Match\n";**
70. **/// String reverse and checking is a string is palindrome or not**
71. **s = "asddsa";**
72. **tmp = s;**
73. **reverse( tmp.begin(), tmp.end() );**
75. **if ( tmp == s ) cout << "Yes Palindrome" << endl;**
76. **else cout << "Not Palindrome" << endl;**
78. **/// String sorting in non-decreasing order**
79. **s = "gfds";**
80. **sort ( s.begin(), s.end() );**
81. **cout << s << endl; /// dfgs**
83. **/// String sorting in non-increasing order**
84. **sort ( s.rbegin(), s.rend() );**
85. **cout << s << endl; /// sgfd**
87. **/// Getting all unique elements of a string. Be care full, string should be sorted.**
88. **s = "aaadddsss";**
89. **int n = unique( s.begin(), s.end() ) - s.begin();**
90. **for ( int i = 0; i < n; i++ ) cout << s[i];/// ads**
91. **cout << endl;**
93. **/// Getting maximum element of string**
94. **cout << \*max\_element( s.begin(), s.end() ) << endl; /// s**
95. **/// Getting minimum element of string**
96. **cout << \*min\_element( s.begin(), s.end() ) << endl; /// a**
98. **/// When we want to take input with space**
99. **/// input : Muhammad Shahriar Alam**
101. **char c;**
102. **cin >> c;**
103. **getline( cin, s );**
104. **s = c + s;**
106. **cout << s << endl; /// Muhammad Shahriar Alam**
108. **/// If we need to sort some string on lexicographical order :**
110. **vector<string> v;**
111. **v.push\_back( "Muhammad" );**
112. **v.push\_back( "Nova" );**
113. **v.push\_back( "Maslenia Mubarrat" );**
114. **v.push\_back( "CPS Academy" );**
115. **v.push\_back( "Rashedul Alam Anik" );**
116. **v.push\_back( "Farhan sadik Sakib" );**
117. **v.push\_back( "Gazi Mohaimin Iqbal" );**
119. **sort ( v.begin(), v.end() );**
120. **for ( auto u : v ) cout << u << endl;**
122. ***/\*\****
124. ***Out put :***
126. ***CPS Academy***
127. ***Farhan sadik Sakib***
128. ***Gazi Mohaimin Iqbal***
129. ***Maslenia Mubarrat***
130. ***Muhammad***
131. ***Nova***
132. ***Rashedul Alam Anik***
134. ***\*/***
136. **s = "asdf";**
138. **s.pop\_back(); /// removes last char of string**
139. **cout << s.back() << endl; /// print last char of string**
141. **v.clear();**
143. **v = { "Shahriar", "Shahriar", "Momo", "Momo", "Sharif", "Sharif" };**
144. **int Sz = unique ( v.begin(), v.end() ) - v.begin();**

147. **cout << Sz << endl; /// Number of unique strings in vector v;**
148. **for ( int i = 0; i < Sz; i++ ) cout << v[i] << endl; /// Prints all unique strings in vector v**
150. **/// Converting int to string**
151. **int a = 123;**
152. **s = to\_string (a);**
153. **cout << s << endl; /// 123**
154. **s[0] = '3';**
155. **cout << s << endl; /// 323**
157. **/// Converting string to integer**
159. **s = "123";**
160. **a = stoi ( s );**
161. **cout << a << endl; /// 123**
162. **a++;**
163. **cout << a << endl; /// 124;**
165. **/// Deleting a substring from string**
167. **s = "ShaKAKAhriar";**
169. **s.erase ( s.begin()+3, s.begin()+7 ); /// erase substring "KAKA" from string s**
170. **cout << s << endl;**

173. **/// Copying a substring of a string to a string**
174. **tmp = "Gagha Alam Gadha";**
175. **s = "Shahriar ";**
177. **copy ( tmp.begin()+6, tmp.begin()+10, back\_inserter ( s ) ); /// copying "Alam substring to string s back.**
178. **cout << s << endl; /// Shahriar Alam**
180. **/// Erasing all occurrence of a specific char from string.**
182. **s = "aaassdddaaasdd";**
183. **s.erase ( remove ( s.begin(), s.end(), 'a' ), s.end() ); /// removes all 'a' from s**
184. **cout << s << endl;**
186. **/// Checking is a string is substring of another string in O(n\*m)**
187. **s = "ashshasdakks";**
189. **if ( s.find( "asd" ) != -1 ) cout << "Substring found";**
190. **else cout << "Not found";**
192. **//transform all in lowercase or uppercase**
193. **//transform(s1.begin(),s1.end(),s1.begin(),::tolower);**
194. **//transform(s1.begin(),s1.end(),s1.begin(),::toupper);**
196. **return 0;**
197. **}**
198. **// CPP program to illustrate substr()**
199. **#include <string.h>**
200. **#include <iostream>**
201. **using namespace std;**
203. **int main()**
204. **{**
205. **// Take any string**
206. **string s = "dog:cat";**
208. **// Find position of ':' using find()**
209. **int pos = s.find(":");**
211. **// Copy substring after pos**
212. **string sub = s.substr(pos + 1);**
214. **// prints the result**
215. **cout << "String is: " << sub;**
217. **return 0;**
218. **}**

**Output:**

1. **String is: cat**
2. **// CPP program to illustrate substr()**
3. **#include <string.h>**
4. **#include <iostream>**
5. **using namespace std;**
7. **int main()**
8. **{**
9. **// Take any string**
10. **string s1 = "Geeks";**
12. **// Copy three characters of s1 (starting**
13. **// from position 1)**
14. **string r = s1.substr(1, 3);**
16. **// prints the result**
17. **cout << "String is: " << r;**
19. **return 0;**
20. **}**

**Output:**

**String is: eek**

**Pair:**

1. **#include<bits/stdc++.h>**
2. **using namespace std;**

5. **bool cmp ( const pair<int,int> &p1, const pair<int, int> &p2  )**
6. **{**
7. **if ( p1.first > p2.first ) return 1;**
8. **if ( p1.first == p2.first ) return ( p1.second < p2.second );**
9. **return 0;**
10. **}**
12. **int main()**
13. **{**
15. **/// Declear a pair o integers**
17. **pair<int,int> p;**
19. **p = make\_pair ( 2, 3 );**
20. **cout << p.first << " " << p.second << endl; /// 2 3**
22. **p.first++;**
23. **cout << p.first << " " << p.second << endl; /// 3 3**
25. **pair<int, int> p1 = { 2, 3 };**
26. **pair<int, int> p2 = { 1, 6 };**
28. **/// Getting minimum of 2 pairs**
29. **p = min ( p1, p2 );**
30. **cout << p.first << " " << p.second << endl; /// 1 6**
32. **/// Getting maximum of 2 pairs**
33. **p = max ( p1, p2 );**
34. **cout << p.first << " " << p.second << endl; /// 2 3**
36. **/// Sorting pair of integers**
38. **vector<pair<int,int>> v;**
39. **v.push\_back ( { 1, 5 } );**
40. **v.push\_back ( { 2, 5 } );**
41. **v.push\_back ( { 7, 1 } );**
42. **v.push\_back ( { 3, 6 } );**
43. **v.push\_back ( { 3, 6 } );**
44. **v.push\_back ( { 7, 1 } );**
46. **sort ( v.begin(), v.end() );**
47. **for ( auto u : v ) cout << u.first << " " << u.second << endl;**
48. **cout << endl;**
49. ***/\*\****
50. ***1 5***
51. ***2 5***
52. ***3 6***
53. ***3 6***
54. ***7 1***
55. ***7 1***
57. ***\*/***
59. **/// Making unique pair of integers**
61. **int Sz = unique ( v.begin(), v.end() ) - v.begin();**
62. **cout << Sz << endl;**
63. **for ( int i = 0; i < Sz; i++ ) cout << v[i].first << " " << v[i].second << endl;**
64. **cout << endl;**
66. ***/\*\****
67. ***4***
68. ***1 5***
69. ***2 5***
70. ***3 6***
71. ***7 1***
73. ***\*/***
75. **/// sorting using comparator**
76. **v = { {2, 3}, {4, 5}, {1, 5}, {1, 6}, {6, 7}, {6, 8} };**
78. **sort ( v.begin(), v.end(), cmp );**
79. **for ( auto u : v ) cout << u.first << " " << u.second << endl;**
80. **cout << endl;**
82. ***/\*\****
84. ***6 7***
85. ***6 8***
86. ***4 5***
87. ***2 3***
88. ***1 5***
89. ***1 6***
91. ***\*/***

94. **v = { {2, 3}, {4, 5}, {1, 5}, {1, 6}, {6, 7}, {6, 8} };**
96. **for ( int i = 0; i < v.size(); i++ ) v[i].first \*= -1;**
97. **sort ( v.begin(), v.end() );**
98. **for ( auto u : v ) cout << (u.first\*-1) << " " << u.second << endl;**
99. **cout << endl;**
101. ***/\*\****
103. ***6 7***
104. ***6 8***
105. ***4 5***
106. ***2 3***
107. ***1 5***
108. ***1 6***
110. ***\*/***

113. **return 0;**
114. **}**

**//swap two pair..**

1. **#include <iostream>**
2. **#include<utility>**
4. **using namespace std;**
6. **int main()**
7. **{**
8. **pair<char, int>pair1 = make\_pair('A', 1);**
9. **pair<char, int>pair2 = make\_pair('B', 2);**
11. **cout << "Before swapping:\n " ;**
12. **cout << "Contents of pair1 = "**
13. **<< pair1.first << " " << pair1.second ;**
14. **cout << "Contents of pair2 = "**
15. **<< pair2.first << " " << pair2.second ;**
16. **pair1.swap(pair2);**
18. **cout << "\nAfter swapping:\n ";**
19. **cout << "Contents of pair1 = "**
20. **<< pair1.first << " " << pair1.second ;**
21. **cout << "Contents of pair2 = "**
22. **<< pair2.first << " " << pair2.second ;**
24. **return 0;**
25. **}**

**Output:**

1. **Before swapping:**
2. **Contents of pair1 = (A, 1)**
3. **Contents of pair2 = (B, 2)**
4. **After swapping:**
5. **Contents of pair1 = (B, 2)**
6. **Contents of pair2 = (A, 1)**

**//CPP program to illustrate pair in STL**

1. **#include <iostream>**
2. **#include <utility>**
3. **#include <string>**
4. **using namespace std;**
6. **int main()**
7. **{**
8. **pair <string, int> g1;**
9. **pair <string, int> g2("Quiz", 3);**
10. **pair <string, int> g3(g2);**
11. **pair <int, int> g4(5, 10);**
13. **g1 = make\_pair(string("Geeks"), 1);**
14. **g2.first = ".com";**
15. **g2.second = 2;**
17. **cout << "This is pair g" << g1.second << " with "**
18. **<< "value " << g1.first << "." << endl << endl;**
20. **cout << "This is pair g" << g3.second**
21. **<< " with value " << g3.first**
22. **<< "This pair was initialized as a copy of "**
23. **<< "pair g2" << endl << endl;**
25. **cout << "This is pair g" << g2.second**
26. **<< " with value " << g2.first**
27. **<< "\nThe values of this pair were"**
28. **<< " changed after initialization."**
29. **<< endl << endl;**
31. **cout << "This is pair g4 with values "**
32. **<< g4.first << " and " << g4.second**
33. **<< " made for showing addition. \nThe "**
34. **<< "sum of the values in this pair is "**
35. **<< g4.first+g4.second**
36. **<< "." << endl << endl;**
38. **cout << "We can concatenate the values of"**
39. **<< " the pairs g1, g2 and g3 : "**
40. **<< g1.first + g3.first + g2.first**
41. **<< endl << endl;**
43. **cout << "We can also swap pairs "**
44. **<< "(but type of pairs should be same) : "**
45. **<< endl;**
46. **cout << "Before swapping, " << "g1 has "**
47. **<< g1.first**
48. **<< " and g2 has " << g2.first << endl;**
49. **swap(g1, g2);**
50. **cout << "After swapping, "**
51. **<< "g1 has " << g1.first << " and g2 has "**
52. **<< g2.first;**
54. **return 0;**
55. **}**

**Output:**

**This is pair g1 with value Geeks.**

**This is pair g3 with value Quiz**

**This pair was initialized as a copy of pair g2**

**This is pair g2 with value .com**

**The values of this pair were changed**

**after initialization.**

**This is pair g4 with values 5 and 10 made**

**for showing addition.**

**The sum of the values in this pair is 15.**

**We can concatenate the values of the pairs g1,**

**g2 and g3 : GeeksQuiz.com**

**We can also swap pairs (but type of pairs should be same) :**

**Before swapping, g1 has Geeks and g2 has .com**

**After swapping, g1 has .com and g2 has Geeks**

**Map:**

1. **#include<bits/stdc++.h>**
2. **using namespace std;**
4. **int main()**
5. **{**
6. **ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);**
8. **//User Input , Declare and Initializations**
10. **map < string, int > id; // we can use <string,string> or <int,string>**
11. **//<long long,int> or <pair<int,int>> , or <int,vector> etc.**
13. **int size\_of\_map;**
14. **cin >> size\_of\_map;**
16. **for(int i = 0; i < size\_of\_map; i++)**
17. **{**
18. **string s;//Index or "Key" of map (student name)**
19. **int input\_id;//"Value" of the Key (student id)**
21. **cin>>s>>input\_id;**
23. **id[s] = input\_id;**
24. **}**
26. **cout<<"Output Using For Each Loop: "<< endl << endl;**
28. **for(auto i : id)cout << i.first<<" = "<< i.second << endl;**
30. **cout<<endl;**
32. **cout<<"Output Using For Loop: "<< endl << endl;**
34. **map < string, int > :: iterator it;**
36. **for(auto it = id.begin(); it != id.end(); it++)**
37. **cout << it->first << " = " << it->second << endl;**
39. **cout << endl;**
41. ***/\* Input:***
42. ***5***
43. ***Shajib 5***
44. ***Arif 2***
45. ***Hasib 3***
46. ***Shohan 4***
47. ***Shajib 1***
48. ***Output Using For Each Loop:***
50. ***Arif = 2***
51. ***Hasib = 3***
52. ***Shajib = 5***
53. ***Shohan = 4***
55. ***Output Using For Loop:***
57. ***Arif = 2***
58. ***Hasib = 3***
59. ***Shajib = 5***
60. ***Shohan = 4 \*/***
62. **//Always Give Sorted Output**
64. **}**
65. **#include<bits/stdc++.h>**
66. **using namespace std;**
68. **int main()**
69. **{**
70. **ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);**
72. **//count frequency of a BIG Integer ;**
74. **vector <long long> v = {7986432987624398912,7986432987624398912,32687243}; //not possible in array**
75. **map <long long , int> cnt;**
77. **//for(auto i : v)cnt[i]++;**
78. **for(int i=0; i < v.size(); i++)cnt[v[i]]++; // key(element of v) and value(index of map)**
80. **for(auto i : cnt)cout << i.first << " = " << i.second << endl;**
82. **// OUTPUT (sorted)**
84. **//32687243 = 2**
85. **//7986432987624398912 = 1**
87. **//map as a vector parameter**
89. **cout<< "Total Value = " << v.size() <<endl; // 3 (in the vector)**
90. **cout<< "Unique Value = " << cnt.size() <<endl; //2 (automatically remove repeated value)**
92. **cout<<endl<<endl;**
94. **vector<map<string,int>>vec;**
96. **map<string,int>mp;**
98. **mp["Shajib"] = 1;**
99. **mp["Shohan"] = 2;**
101. **vec.push\_back(mp);//value 0 index in vector.**
103. **cout<<"vec[0][\"Shajib\"] : " << vec[0]["Shajib"] <<endl;//output 1**
104. **cout<<"vec[0][\"Shohan\"] : " << vec[0]["Shohan"] <<endl;//output 2**
105. **}**
106. **#include <iostream>**
107. **#include <iterator>**
108. **#include <map>**
110. **using namespace std;**
112. **int main()**
113. **{**
115. **// empty map container**
116. **map<int, int> gquiz1;**
118. **// insert elements in random order**
119. **gquiz1.insert(pair<int, int>(1, 40));**
120. **gquiz1.insert(pair<int, int>(2, 30));**
121. **gquiz1.insert(pair<int, int>(3, 60));**
122. **gquiz1.insert(pair<int, int>(4, 20));**
123. **gquiz1.insert(pair<int, int>(5, 50));**
124. **gquiz1.insert(pair<int, int>(6, 50));**
125. **gquiz1.insert(pair<int, int>(7, 10));**
127. **// printing map gquiz1**
128. **map<int, int>::iterator itr;**
129. **cout << "\nThe map gquiz1 is : \n";**
130. **cout << "\tKEY\tELEMENT\n";**
131. **for (itr = gquiz1.begin(); itr != gquiz1.end(); ++itr) {**
132. **cout << '\t' << itr->first**
133. **<< '\t' << itr->second << '\n';**
134. **}**
135. **cout << endl;**
137. **// assigning the elements from gquiz1 to gquiz2**
138. **map<int, int> gquiz2(gquiz1.begin(), gquiz1.end());**
140. **// print all elements of the map gquiz2**
141. **cout << "\nThe map gquiz2 after"**
142. **<< " assign from gquiz1 is : \n";**
143. **cout << "\tKEY\tELEMENT\n";**
144. **for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr) {**
145. **cout << '\t' << itr->first**
146. **<< '\t' << itr->second << '\n';**
147. **}**
148. **cout << endl;**
150. **// remove all elements up to**
151. **// element with key=3 in gquiz2**
152. **cout << "\ngquiz2 after removal of"**
153. **" elements less than key=3 : \n";**
154. **cout << "\tKEY\tELEMENT\n";**
155. **gquiz2.erase(gquiz2.begin(), gquiz2.find(3));**
156. **for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr) {**
157. **cout << '\t' << itr->first**
158. **<< '\t' << itr->second << '\n';**
159. **}**
161. **// remove all elements with key = 4**
162. **int num;**
163. **num = gquiz2.erase(4);**
164. **cout << "\ngquiz2.erase(4) : ";**
165. **cout << num << " removed \n";**
166. **cout << "\tKEY\tELEMENT\n";**
167. **for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr) {**
168. **cout << '\t' << itr->first**
169. **<< '\t' << itr->second << '\n';**
170. **}**
172. **cout << endl;**
174. **// lower bound and upper bound for map gquiz1 key = 5**
175. **cout << "gquiz1.lower\_bound(5) : "**
176. **<< "\tKEY = ";**
177. **cout << gquiz1.lower\_bound(5)->first << '\t';**
178. **cout << "\tELEMENT = "**
179. **<< gquiz1.lower\_bound(5)->second << endl;**
180. **cout << "gquiz1.upper\_bound(5) : "**
181. **<< "\tKEY = ";**
182. **cout << gquiz1.upper\_bound(5)->first << '\t';**
183. **cout << "\tELEMENT = "**
184. **<< gquiz1.upper\_bound(5)->second << endl;**
186. **return 0;**
187. **}**

**Output:**

**The map gquiz1 is :**

**KEY ELEMENT**

**1 40**

**2 30**

**3 60**

**4 20**

**5 50**

**6 50**

**7 10**

**The map gquiz2 after assign from gquiz1 is :**

**KEY ELEMENT**

**1 40**

**2 30**

**3 60**

**4 20**

**5 50**

**6 50**

**7 10**

**gquiz2 after removal of elements less than key=3 :**

**KEY ELEMENT**

**3 60**

**4 20**

**5 50**

**6 50**

**7 10**

**gquiz2.erase(4) : 1 removed**

**KEY ELEMENT**

**3 60**

**5 50**

**6 50**

**7 10**

**gquiz1.lower\_bound(5) : KEY = 5 ELEMENT = 50**

**gquiz1.upper\_bound(5) : KEY = 6 ELEMENT = 50**

1. **// CPP program to illustrate**
2. **// Implementation of swap() function**
3. **#include <bits/stdc++.h>**
4. **using namespace std;**
6. **int main()**
7. **{**
8. **// Take any two maps**
9. **map<int, char> map1;**
10. **map<char, int> map2;**
12. **map1[1] = 'a';**
13. **map1[2] = 'b';**
14. **map1[3] = 'c';**
15. **map1[4] = 'd';**
17. **map2['w'] = 1;**
18. **map2['y'] = 2;**
19. **map2['z'] = 3;**
21. **// Print the associated element**
22. **cout << "Element at map1[2] = "**
23. **<< map1.at(2) << endl;**
25. **cout << "Element at map2['w'] = "**
26. **<< map2.at('w') << endl;**
28. **return 0;**
29. **}**

**Output:**

1. **Element at map1[2] = b**
2. **Element at map2['w'] = 1**

**map::swap()  
swap() function is used to exchange the contents of two maps but the maps must be of same type, although sizes may differ.**

**Syntax:**

1. **map1.swap(map2)**
2. **OR**
3. **swap(map1, map2)**
4. **Parameters:**
5. **map1 is the first map object.**
6. **map2 is the second map object.**
7. **// C++ program makes a map to store**
8. **// elements in descending order.**
9. **#include<bits/stdc++.h>**
10. **using namespace std;**
12. **int main()**
13. **{**
14. **// Here if greater<int> is used to make**
15. **// sure that elements are stored in**
16. **// descending order of keys.**
17. **map<int, string, greater <int> > mymap;//using greater**
19. **// Inserting the elements one by one**
20. **mymap.insert(make\_pair(10, "queen"));**
21. **mymap.insert(make\_pair(20, "rose"));**
22. **mymap.insert(make\_pair(5," lion"));**
24. **// begin() returns to the first value of map.**
25. **map<int,string> :: iterator it;**
26. **for (it=mymap.begin() ; it!=mymap.end() ; it++)**
27. **cout << "(" << (\*it).first << ", "**
28. **<< (\*it).second << ")" << endl;**
30. **return 0;**
31. **}**

**Output:**

1. **(20, rose)**
2. **(10, queen)**
3. **(5, lion)**

**Multimap:**

1. **#include <iostream>**
2. **#include <map>**
3. **#include <iterator>**
5. **using namespace std;**
7. **int main()**
8. **{**
9. **multimap <int, int> gquiz1; // empty multimap container**
11. **// insert elements in random order**
12. **gquiz1.insert(pair <int, int> (1, 40));**
13. **gquiz1.insert(pair <int, int> (2, 30));**
14. **gquiz1.insert(pair <int, int> (3, 60));**
15. **gquiz1.insert(pair <int, int> (6, 50));**
16. **gquiz1.insert(pair <int, int> (6, 10));**
18. **// printing multimap gquiz1**
19. **multimap <int, int> :: iterator itr;**
20. **cout << "\nThe multimap gquiz1 is : \n";**
21. **cout << "\tKEY\tELEMENT\n";**
22. **for (itr = gquiz1.begin(); itr != gquiz1.end(); ++itr)**
23. **{**
24. **cout << '\t' << itr->first**
25. **<< '\t' << itr->second << '\n';**
26. **}**
27. **cout << endl;**
29. **//adding elements randomly,**
30. **// to check the sorted keys property**
31. **gquiz1.insert(pair <int, int> (4, 50));**
32. **gquiz1.insert(pair <int, int> (5, 10));**
34. **// printing multimap gquiz1 again**
36. **cout << "\nThe multimap gquiz1 after**
37. **adding extra elements is : \n";**
38. **cout << "\tKEY\tELEMENT\n";**
39. **for (itr = gquiz1.begin(); itr != gquiz1.end(); ++itr)**
40. **{**
41. **cout << '\t' << itr->first**
42. **<< '\t' << itr->second << '\n';**
43. **}**
44. **cout << endl;**


48. **// assigning the elements from gquiz1 to gquiz2**
49. **multimap <int, int> gquiz2(gquiz1.begin(),**
50. **gquiz1.end());**
52. **// print all elements of the multimap gquiz2**
53. **cout << "\nThe multimap gquiz2 after**
54. **assign from gquiz1 is : \n";**
55. **cout << "\tKEY\tELEMENT\n";**
56. **for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr)**
57. **{**
58. **cout << '\t' << itr->first**
59. **<< '\t' << itr->second << '\n';**
60. **}**
61. **cout << endl;**
63. **// remove all elements up to**
64. **// element with value 30 in gquiz2**
65. **cout << "\ngquiz2 after removal of**
66. **elements less than key=3 : \n";**
67. **cout << "\tKEY\tELEMENT\n";**
68. **gquiz2.erase(gquiz2.begin(), gquiz2.find(3));**
69. **for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr)**
70. **{**
71. **cout << '\t' << itr->first**
72. **<< '\t' << itr->second << '\n';**
73. **}**
75. **// remove all elements with key = 4**
76. **int num;**
77. **num = gquiz2.erase(4);**
78. **cout << "\ngquiz2.erase(4) : ";**
79. **cout << num << " removed \n" ;**
80. **cout << "\tKEY\tELEMENT\n";**
81. **for (itr = gquiz2.begin(); itr != gquiz2.end(); ++itr)**
82. **{**
83. **cout << '\t' << itr->first**
84. **<< '\t' << itr->second << '\n';**
85. **}**
87. **cout << endl;**
89. **//lower bound and upper bound for multimap gquiz1 key = 5**
90. **cout << "gquiz1.lower\_bound(5) : " << "\tKEY = ";**
91. **cout << gquiz1.lower\_bound(5)->first << '\t';**
92. **cout << "\tELEMENT = " << gquiz1.lower\_bound(5)->second << endl;**
93. **cout << "gquiz1.upper\_bound(5) : " << "\tKEY = ";**
94. **cout << gquiz1.upper\_bound(5)->first << '\t';**
95. **cout << "\tELEMENT = " << gquiz1.upper\_bound(5)->second << endl;**
97. **return 0;**
98. **}**

**Output:**

1. **The multimap gquiz1 is :**
2. **KEY ELEMENT**
3. **1 40**
4. **2 30**
5. **3 60**
6. **4 20**
7. **5 50**
8. **6 50**
9. **6 10**
10. **The multimap gquiz2 after assign from gquiz1 is :**
11. **KEY ELEMENT**
12. **1 40**
13. **2 30**
14. **3 60**
15. **4 20**
16. **5 50**
17. **6 50**
18. **6 10**
19. **gquiz2 after removal of elements less than key=3 :**
20. **KEY ELEMENT**
21. **3 60**
22. **4 20**
23. **5 50**
24. **6 50**
25. **6 10**
26. **gquiz2.erase(4) : 1 removed**
27. **KEY ELEMENT**
28. **3 60**
29. **5 50**
30. **6 50**
31. **6 10**
32. **gquiz1.lower\_bound(5) : KEY = 5 ELEMENT = 50**
33. **gquiz1.upper\_bound(5) : KEY = 6 ELEMENT = 50**
34. **// C++ program makes a multimap to store**
35. **// elements in descending order.**
36. **#include<bits/stdc++.h>**
37. **using namespace std;**
39. **int main()**
40. **{**
41. **// Here if greater<int> is used to make**
42. **// sure that elements are stored in**
43. **// descending order of keys.**
44. **multimap<int, string, greater <int> > mymap;**
46. **// Inserting the elements one by one**
47. **mymap.insert(make\_pair(10, "queen"));**
48. **mymap.insert(make\_pair(20, "rose"));**
49. **mymap.insert(make\_pair(5," lion"));**
50. **mymap.insert(make\_pair(20, "van")); // Duplicates allowed**
51. **mymap.insert(make\_pair(20, "watch"));**
52. **mymap.insert(make\_pair(5,"joker"));**

55. **// begin() returns to the first value of multimap.**
56. **multimap<int,string> :: iterator it;**
57. **for (it=mymap.begin() ; it!=mymap.end() ; it++)**
58. **cout << "(" << (\*it).first << ", "**
59. **<< (\*it).second << ")" << endl;**
61. **return 0;**
62. **}**

**Output:**

1. **(20, rose)**
2. **(20, van)**
3. **(20, watch)**
4. **(10, queen)**
5. **(5, lion)**
6. **(5, joker)**

**Set:**

1. **#include<bits/stdc++.h>**
2. **using namespace std;**

5. **int main()**
6. **{**
7. **/// Declaration of set of integers**
8. **set<int> s = { 1, 1, 2, 1, 3 };**
10. **/// Printing size and elements**
11. **cout << s.size() << endl;**
12. **for ( auto u : s ) cout << u << " "; /// 1 2 3**
13. **cout << endl;**
15. **set<int>:: iterator it;**
16. **for ( it = s.begin(); it != s.end(); it++ ) cout << \*it << " "; /// 1 2 3**
17. **cout << endl;**
19. **/// clearing and checking is empty set**
20. **s.clear();**
21. **cout << s.empty() << endl; /// 1**
23. **/// inserting in set**
24. **s.insert ( 1 );**
25. **s.insert ( 1 );**
26. **s.insert ( 1 );**
27. **s.insert ( 2 );**
28. **s.insert ( 1 );**
29. **s.insert ( 3 );**
31. **cout << s.size() << endl; /// 3**
32. **for ( auto u : s ) cout << u << " "; /// 1 2 3**
33. **cout << endl;**
35. **/// checking is specific element is in a set**
36. **cout << s.count ( 2 ) << endl; /// 1**
37. **cout << s.count ( 4 ) << endl; /// 0**

40. **/// Front element in set**
41. **cout << \*s.begin() << endl; /// 1**
43. **/// Last element in set**
44. **cout << \*(--s.end()) << endl; /// 3**
45. **cout << \*s.rbegin() << endl; /// 3**

48. **/// Erase an element;**
50. **s = { 1, 2, 3, 4, 5, 6 };**
52. **s.erase ( 2 );**
53. **cout << s.size() << endl; /// 5**
54. **for ( auto u : s ) cout << u << " "; /// 1 3 4 5 6**
55. **cout << endl;**
57. **s.erase ( 10 );**
58. **cout << s.size() << endl; /// 5**
59. **for ( auto u : s ) cout << u << " "; /// 1 3 4 5 6**
60. **cout << endl;**

63. **/// Erasing front element in set**
65. **s = { 1, 2, 3, 4, 5, 6 };**
67. **s.erase ( s.begin() );**
68. **cout << s.size() << endl; /// 5**
69. **for ( auto u : s ) cout << u << " "; /// 2 3 4 5 6**
70. **cout << endl;**
72. **/// Erasing back element in set**
74. **s = { 1, 2, 3, 4, 5, 6 };**
76. **s.erase ( --s.end() );**
77. **cout << s.size() << endl; /// 5**
78. **for ( auto u : s ) cout << u << " "; /// 1 2 3 4 5**
79. **cout << endl;**

82. **/// set of pair**
83. **set<pair<int, int>> s1;**
85. **s1.insert ( { 1, 2 } );**
86. **s1.insert ( { 1, 2 } );**
87. **s1.insert ( { 4, 2 } );**
88. **s1.insert ( { 4, 3 } );**
89. **s1.insert ( { 2, 2 } );**
90. **s1.insert ( { 2, 1 } );**
92. ***/\*\****
94. ***Output :***
96. ***5***
97. ***1 2***
98. ***2 1***
99. ***2 2***
100. ***4 2***
101. ***4 3***
103. ***\*/***
105. **cout << s1.size() << endl;**
106. **for ( auto u : s1 ) cout << u.first << " " << u.second << endl;**
108. **/// Set of string**
110. **set<string> s2;**
112. **s2.insert ( "momo" );**
113. **s2.insert ( "momo" );**
114. **s2.insert ( "prety" );**
115. **s2.insert ( "prety" );**
116. **s2.insert ( "shahriar" );**
117. **s2.insert ( "nobel" );**
118. **s2.insert ( "sharif" );**
119. **s2.insert ( "proma" );**
121. **cout << s2.size() << endl;**
122. **for ( auto u : s2 ) cout << u << endl;**
124. ***/\*\****
126. ***Output :***
128. ***6***
129. ***momo***
130. ***nobel***
131. ***prety***
132. ***proma***
133. ***shahriar***
134. ***sharif***
136. ***\*/***
138. **/// set in discanding order**
139. **set<int, greater<int>> s3 = {3, 4, 1, 2};**
140. **for ( auto u : s3 ) cout << u << " "; /// 4 3 2 1**
141. **cout << endl;**

144. **set<string, greater<string>> s4;**
146. **s4.insert ( "momo" );**
147. **s4.insert ( "momo" );**
148. **s4.insert ( "prety" );**
149. **s4.insert ( "prety" );**
150. **s4.insert ( "shahriar" );**
151. **s4.insert ( "nobel" );**
152. **s4.insert ( "sharif" );**
153. **s4.insert ( "proma" );**
155. **cout << s4.size() << endl;**
156. **for ( auto u : s4 ) cout << u << endl;**
158. ***/\*\****
159. ***Output :***
161. ***6***
162. ***sharif***
163. ***shahriar***
164. ***proma***
165. ***prety***
166. ***nobel***
167. ***momo***
169. ***\*/***
171. **set<pair<int, int>, greater<pair<int,int>>> s5;**
173. **s5.insert ( { 1, 2 } );**
174. **s5.insert ( { 1, 2 } );**
175. **s5.insert ( { 4, 2 } );**
176. **s5.insert ( { 4, 3 } );**
177. **s5.insert ( { 2, 2 } );**
178. **s5.insert ( { 2, 1 } );**

181. **cout << s5.size() << endl;**
182. **for ( auto u : s5 ) cout << u.first << " " << u.second << endl;**
184. ***/\*\****
186. ***Output :***
187. ***5***
188. ***4 3***
189. ***4 2***
190. ***2 2***
191. ***2 1***
192. ***1 2***
194. ***\*/***

197. **return 0;**
198. **}**
199. **#include<bits/stdc++.h>**
200. **using namespace std;**
202. **int main()**
203. **{**
204. **ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);**
206. **vector<int>v = {2,2,3,3,1,1};**
208. **set<int>s(v.begin(),v.end());//1 2 3**

211. **cout<<s.count(5)<<endl;//1**
213. **cout<<\*s.begin()+2<<endl;//3,Here,\*s.begin()+2=1+2=3**
215. **cout<<\*(--s.end()) << " "<<\*s.end()<<endl;//same (3 3)**
217. **}**
218. **// CPP program to demonstrate the**
219. **// set::find() function and find the position of an element**
220. **#include <bits/stdc++.h>**
221. **using namespace std;**
222. **int main()**
223. **{**
225. **// Initialize set**
226. **set<int> s;**
228. **s.insert(1);**
229. **s.insert(4);**
230. **s.insert(2);**
231. **s.insert(5);**
232. **s.insert(3);**
234. **// iterator pointing to**
235. **// position where 2 is**
236. **auto pos = s.find(3);**
238. **// prints the set elements**
239. **cout << "The set elements after 3 are: ";**
240. **for (auto it = pos; it != s.end(); it++)**
241. **cout << \*it << " ";**
243. **return 0;**
244. **}**

**The set is : 1 2 3 4 5 . 3 is standing in index(pos) 2. Loop(pos to end).**

**Output:**

1. **The set elements after 3 are: 3 4 5**

**Multiset:**

1. **// Set is sorted and unique .**
2. **//Multiset is sorted but no unique .**
4. **#include<bits/stdc++.h>**
5. **using namespace std;**
7. **int main()**
8. **{**
9. **ios\_base::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);**
11. **multiset<int>ms;**
12. **ms.insert(5);**
13. **ms.insert(1);**
14. **ms.insert(2);**
15. **ms.insert(2);**
16. **ms.insert(1);**
18. **// output will be all value with sorted.**
19. **for(auto i : ms)cout<<i<<" "; //1 1 2 2 5**
20. **cout<<endl;**
22. **cout<<ms.size()<<endl;//5**
24. **cout<<ms.count(1)<<endl;//2**
26. **ms.erase(1);//erase all 1**
28. **for(auto i : ms)cout<<i<<" ";//2 2 5**
29. **cout<<endl;**
31. **ms.erase(ms.find(2));//(erase just one element.)**
32. **for(auto i : ms)cout<<i<<" ";//2 5**
33. **cout<<endl;**
35. **}**

**Stack: Last In First Out (LIFO).**

1. **#include<bits/stdc++.h>**
2. **using namespace std;**
4. **int main()**
5. **{**
6. **stack < int > st; // declare a stack.**
8. **//pushing element in a stack.**
10. **st.push(1); // first in will be last.**
11. **st.push(2);**
12. **st.push(3);**
13. **st.push(4); //last in will be first (top element).**
15. **cout << st.top() << endl; // 4 (last in->first(top)).**
17. **st.pop(); // top element (4) will be pop (remove).**
19. **cout << st.top() << endl; //3 (after pop 4 -> top will be 3.)**
21. **//How to Print all element in a Stack.**
22. **while( !st.empty() )**
23. **{**
24. **cout << st.top() << " " ;//3 2 1 (Original Stack -> 1 2 3 4) (Output-> 4(popped) 3 2 1).**
25. **st.pop();**
26. **}**
27. **cout << endl ;**
29. **cout<<st.size()<<endl;//0 (empty).**
30. **cout<<st.empty()<<endl;//1 (true).**
31. **}**

**Queue: First In First Out (FIFO).**

1. **#include<bits/stdc++.h>**
2. **using namespace std;**
4. **int main()**
5. **{**
6. **queue < int > q; // Declare a queue.**
8. **//pushing element**
9. **q.push(1);**
10. **q.push(2);**
11. **q.push(3);**
12. **q.push(4);**
13. **//queue = 1 2 3 4**
15. **cout<<q.size()<<endl; //size = 4(total elements).**
17. **cout << q.front() << endl; //1 top(First or Front element).**
18. **cout << q.back() << endl; // 4 (last element)**
20. **q.pop(); //remove front(first)element.**
21. **cout << q.front() << endl; //2 (after popped 1 , 2 is front element).**
23. **//Output all emements.**
25. **// Using iterator.**
26. **for(auto i = q.front(); i != q.back() + 1; i++) cout<<i<<" ";**
27. **//2 3 4 (must use q.back() + 1 or it'll not print last element).**
28. **cout<<endl;**
30. **//another way**
31. **while( !q.empty() )**
32. **{**
33. **cout << q.front() << " " ; // 2 3 4 (1 popped before).**
34. **q.pop();**
35. **}**
36. **}**

**Deque:**

1. **#include<bits/stdc++.h>**
2. **using namespace std;**
4. **int main()**
5. **{**
6. **deque < int > dq;//declare deque.**
8. **dq.push\_front(0); // 0**
9. **dq.push\_front(1); // 1 0**
10. **dq.push\_front(2); // 2 1 0**
11. **dq.push\_back(4);  // 2 1 0 4**
12. **dq.push\_back(5);  // 2 1 0 4 5**
14. **cout << dq.front() << endl; //2 (first element).**
16. **cout << dq.back() << endl; // 5 (last element).**
18. **dq.pop\_front(); // after pop front -> 1 0 4 5 (2 popped).**
19. **dq.pop\_back(); // after pop back -> 1 0 4 (5 popped).**
21. **cout << dq.size() << endl; // size = 3 (total element).**
23. **cout << dq.at(0) <<endl; // 1 (element in index 0).**
25. **//Output Using Iterator**
27. **deque < int > :: iterator it;**
28. **for(auto it = dq.begin(); it != dq.end(); it++) cout << \*it << " "; // 1 0 4 (ascending order).**
29. **cout<<endl;**
31. **for(auto it = dq.rbegin(); it != dq.rend(); it++) cout << \*it << " "; // 4 0 1 (descending order).**
32. **cout<<endl;**
34. **// Without Iterator.**
35. **while(!dq.empty())**
36. **{**
37. **cout << dq.front() << " ";**
38. **dq.pop\_front() ;**
39. **}**
40. **cout<<endl;**
42. **dq.clear(); // clear a deque.**
43. **cout << dq.size() << endl; // 0 (after clear the deque).**
44. **}**

**Priority Queue:**

1. **#include<bits/stdc++.h>**
2. **using namespace std;**
4. **int main()**
5. **{**
6. **priority\_queue < int > pq; // declare**
8. **//pushing element (Big Integer will get priority)**
10. **pq.push(1); // 1**
11. **pq.push(2); // 2 1**
12. **pq.push(5); // 5 2 1**
13. **pq.push(3); // 5 3 2 1**
14. **pq.push(4); // 5 4 3 2 1 (like as sort in descending order).**
16. **cout << pq.top() << endl; // 5 (big Integer) (Output will be 1 for normal queue and 4 for stack).**
18. **pq.pop(); // top element 5 will be removed.**
20. **cout << pq.top() << endl; // 4 (after remove 5 , top will be 4(2nd big element)).**
22. **// Output full priority\_queue**
24. **while(!pq.empty())**
25. **{**
26. **cout << pq.top() << " "; // 4 3 2 1**
27. **pq.pop();**
28. **}**
29. **cout<<endl;**
31. **cout << pq.size() << endl; // size = 0 (all element is popped already).**
33. **//If we want the priority\_queue will be Ascending order then we need to use comparator.**
35. **priority\_queue < int , vector < int > , greater < int > > new\_pq;**
36. **new\_pq.push(1); // 1**
37. **new\_pq.push(2); // 1 2**
38. **new\_pq.push(1); // 1 1 2**
39. **new\_pq.push(3); // 1 1 2 3**
40. **new\_pq.push(2); // 1 1 2 2 3 (like as sort in Ascending order).**
42. **cout << new\_pq.top() << endl; // 1 (small element).**
44. **//for Pair**
45. **priority\_queue < pair <int,int> , vector < pair <int,int>> , greater < pair <int,int> > > p\_pq;**
47. **p\_pq.push( { 4 , 5 } );**
48. **p\_pq.push( { 3 , 4 } );**
49. **p\_pq.push( { 1 , 2 } );**
50. **p\_pq.push( { 3 , 3 } );**
51. **p\_pq.push( { 0 , 1 } ); // for greater small pair (0,1) will get priority.**
53. **cout << p\_pq.top().first << " " << p\_pq.top().second << endl; // 0 1 (small pair).**
55. **while(!p\_pq.empty())**
56. **{**
57. **cout << p\_pq.top().first << " " << p\_pq.top().second <<endl;**
58. **p\_pq.pop();**
59. **}**
61. **//Output**
62. ***/\* 0 1 (small pair)***
63. ***1 2***
64. ***3 3***
65. ***3 4***
66. ***4 5 \*/***
67. **}**

**Template:**

**//Md. Shajibul Islam..**

**//East West University, CSE Department'19**

**#include<bits/stdc++.h>**

**using namespace std;**

**typedef long long ll;**

**typedef vector<int> vi;**

**typedef vector<ll> vl;**

**typedef vector<vi> vvi;**

**typedef vector<vl> vvl;**

**typedef pair<int,int> pii;**

**typedef pair<double, double> pdd;**

**typedef pair<ll, ll> pll;**

**typedef vector<pii> vii;**

**typedef vector<pll> vll;**

**typedef vector<int>::iterator vit;**

**typedef set<int>::iterator sit;**

**#define FAST ios\_base::sync\_with\_stdio(0); cin.tie(0); cout.tie(0)**

**#define endl '\n'**

**#define loop(i,n) for(int i=0;i<n;i++)**

**#define CASE\_PRINT cout<<"Case "<<C<<": "**

**#define CASE\_PRINT2 cout<<"Case "<<C<<":"<<endl**

**#define ll long long**

**#define ld long double**

**#define Pi 2\*acos(0.0) // acos(-1.0)**

**#define PB push\_back**

**#define F first**

**#define S second**

**#define MP make\_pair**

**#define all(a) (a).begin(),(a).end()**

**#define mid(l,r) ((r+l)/2)**

**#define left(node) (node\*2)**

**#define right(node) (node\*2+1)**

**#define mx\_int\_prime 999999937**

**#define mem(a,b) memset(a, b, sizeof(a) )**

**#define gcd(a,b) \_\_gcd(a,b)**

**#define sqr(a) ((a) \* (a))**

**template < typename F, typename S >**

**ostream& operator << ( ostream& os, const pair< F, S > & p ) {**

**return os << "(" << p.first << ", " << p.second << ")";**

**}**

**template < typename T >**

**ostream &operator << ( ostream & os, const vector< T > &v ) {**

**os << "{";**

**for(auto it = v.begin(); it != v.end(); ++it) {**

**if( it != v.begin() ) os << ", ";**

**os << \*it;**

**}**

**return os << "}";**

**}**

**template < typename T >**

**ostream &operator << ( ostream & os, const set< T > &v ) {**

**os << "[";**

**for(auto it = v.begin(); it != v.end(); ++it) {**

**if( it != v.begin() ) os << ", ";**

**os << \*it;**

**}**

**return os << "]";**

**}**

**template < typename T >**

**ostream &operator << ( ostream & os, const multiset< T > &v ) {**

**os << "[";**

**for(auto it = v.begin(); it != v.end(); ++it) {**

**if( it != v.begin() ) os << ", ";**

**os << \*it;**

**}**

**return os << "]";**

**}**

**template < typename F, typename S >**

**ostream &operator << ( ostream & os, const map< F, S > &v ) {**

**os << "[";**

**for(auto it = v.begin(); it != v.end(); ++it) {**

**if( it != v.begin() ) os << ", ";**

**os << it -> first << " = " << it -> second ;**

**}**

**return os << "]";**

**}**

**#define dbg(args...) do {cerr << #args << " : "; faltu(args); } while(0)**

**void faltu () {**

**cerr << endl;**

**}**

**template <typename T>**

**void faltu( T a[], int n ) {**

**for(int i = 0; i < n; ++i) cerr << a[i] << ' ';**

**cerr << endl;**

**}**

**template <typename T, typename ... hello>**

**void faltu( T arg, const hello &... rest) {**

**cerr << arg << ' ';**

**faltu(rest...);**

**}**

**const double PI = acos(-1.0);**

**const double eps = 1e-9;**

**const int inf = 2000000000;**

**const ll infLL = 9000000000000000000;**

**#define MOD 1000000007**

**int dx[] = {0, 0, +1, -1, +1, +1, -1, -1};**

**int dy[] = {+1, -1, 0, 0, +1, -1, +1, -1};**

**const int mx = 1e5+123;**

**int main()**

**{**

**//freopen("input.txt","r",stdin);**

**//freopen("output.txt", "w", stdout);**

**FAST;**

**int TC,C=1;cin>>TC;**

**while(TC--)**

**{**

**//C++;**

**}**

**}**