**C++ Assignment [ 18-01-2018 ]**

|  |  |
| --- | --- |
| **Emp Name** | **Program And Output** |
| **1\_Rahul** | **/\*Program using Vector \*/**  #include<iostream>  #include<vector>  using namespace std;  void display(vector<int> &v)  { for(int itr=0; itr<v.size();itr++)  {  cout<<v[itr] <<" ";  }  cout<<"\n";  }  int main()  {  vector<int> vt; //create a vector of type int  int num;  cout<<"Initial size : "<<vt.size()<<endl;  //putting values into the vector  cout<<"Enter five integer values : ";  for(int itr=0;itr<5;itr++)  { cin>>num;  vt.push\_back(num);  }  cout<<"\nSize after adding 5 values :"<< vt.size()<<endl;  //Display the contents  cout<<"Current contents :";  display(vt);  //Add one more value  cout<<"Added one more value"<<endl;  vt.push\_back(6.6); //float value truncated to int  //Display the contents & size  cout<<"Now size is :"<< vt.size()<<endl;  cout<<"Now contents : ";  display(vt);  //Inserting elements  vector<int>::iterator itr =vt.begin(); // iterator  itr=itr+3; //itr pointer to 4th element  vt.insert(itr,1,9);  //Display the contents & size  cout<<"Size after inserting :"<< vt.size()<<endl;  cout<<"Contents after inserting : ";  display(vt);  //Removing 4th and 5th elements  vt.erase(vt.begin() +3, vt.begin()+5);  //Display the contents & size  cout<<"Size after deletion :"<< vt.size()<<endl;  cout<<"Contents after deletion : ";  display(vt);  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **Initial size : 0**  **Enter five integer values : 1 2 3 4 5**  **Size after adding 5 values :5**  **Current contents :1 2 3 4 5**  **Added one more value**  **Now size is :6**  **Now contents : 1 2 3 4 5 6**  **Size after inserting :7**  **Contents after inserting : 1 2 3 9 4 5 6**  **Size after deletion :5**  **Contents after deletion : 1 2 3 5 6** |
| **2\_Ashish\_Jain** | **/\* Program using List \*/**  #include <iostream>  #include <list>  using namespace std;  int main ()  {  std::list<int> first;  std::list<int> second (4,100);  std::list<int> third (second.begin(),second.end());  std::list<int> fourth (third);  int array[] = {16,2,77,29};  std::list<int> fifth (array, array + sizeof(array) / sizeof(int) );  std::cout << "The contents: ";  for (std::list<int>::iterator it = fifth.begin(); it != fifth.end(); it++)  std::cout << \*it << ' ';  std::cout << '\n';  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **The contents: 16 2 77 29** |
| **3\_Meena** | **/\* Program using Deque \*/**  #include <iostream>  #include <deque>  using namespace std;    void showdeque(deque <int> g)  {  deque <int> :: iterator iter;  for (iter = g.begin(); iter != g.end(); ++iter)  cout << '\t' << \*iter;  cout << '\n';  }  int main()  {  deque <int> que;  que.push\_back(10);  que.push\_front(20);  que.push\_back(30);  que.push\_front(15);  cout << "The deque que is : ";  showdeque(que);    cout << "\nque.size() : " << que.size();  cout << "\nque.max\_size() : " << que.max\_size();    cout << "\nque.at(2) : " << que.at(2);  cout << "\nque.front() : " << que.front();  cout << "\nque.back() : " << que.back();  cout << "\nque.pop\_front() : ";  que.pop\_front();  showdeque(que);  cout << "\nque.pop\_back() : ";  que.pop\_back();  showdeque(que);  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **The deque que is : 15 20 10 30**  **que.size() : 4**  **que.max\_size() : 4611686018427387903**  **que.at(2) : 10**  **que.front() : 15**  **que.back() : 30**  **que.pop\_front() : 20 10 30**  **que.pop\_back() : 20 10** |
| **4\_Divya\_Bolu** | **/\* Stack Operration \*/**  #include<iostream>  #include<conio.h>  #include<stdlib.h>  #define MAX\_SIZE 5  using namespace std;  int main() {  int item, choice, i;  int arr\_stack[MAX\_SIZE];  int top = 0;  int exit = 1;  cout << "\nSimple Stack Example - Array - C++";  do {  cout << "\n\nnStack Main Menu";  cout << "\n1.Push \n2.Pop \n3.Display \nOthers to exit";  cout << "\nEnter Your Choice : ";  cin>>choice;  switch (choice) {  case 1:  if (top == MAX\_SIZE)  cout << "\n## Stack is Full!";  else {  cout << "\nEnter The Value to be pushed : ";  cin>>item;  cout << "\n## Position : " << top << ", Pushed Value :" << item;  arr\_stack[top++] = item;  }  break;  case 2:  if (top == 0)  cout << "\n## Stack is Empty!";  else {  --top;  cout << "\n## Position : " << top << ", Popped Value :" << arr\_stack[top];  }  break;  case 3:  cout << "\n## Stack Size : " << top;  for (i = (top - 1); i >= 0; i--)  cout << "\n## Position : " << i << ", Value :" << arr\_stack[i];  break;  default:  exit = 0;  break;  }  } while (exit);  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**    **Stack Main Menu**  **1.Push**  **2.Pop**  **3.Display**  **Others to exit**  **Enter Your Choice : 1**  **Enter The Value to be pushed : 4**  **Position : 0, Pushed Value :4**  **Stack Main Menu**  **1.Push**  **2.Pop**  **3.Display**  **Others to exit**  **Enter Your Choice : 3**  **Stack Size : 1**  **Position : 0, Value :4**  **Stack Main Menu**  **1.Push**  **2.Pop**  **3.Display**  **Others to exit**  **Enter Your Choice : 2**  **Position : 0, Popped Value :4** |
| **5\_Pusplata** | **/\* C++ Program To Implement Queue using Linked List \*/**  #include<iostream>  #include<cstdlib>  using namespace std;  /\* Node Declaration \*/  struct node  {  int info;  struct node \*link;  }\*front, \*rear;    /\* Class Declaration \*/  class queue\_list  {  public:  void insert(int);  void display();  void del();  queue\_list()  {  front = NULL;  rear = NULL;  }  };  int main()  {  int choice, item;  queue\_list ql;  while (1)  {  cout<<"\n-------------"<<endl;  cout<<"Operations on Queue"<<endl;  cout<<"\n-------------"<<endl;  cout<<"1.Insert Element into the Queue"<<endl;  cout<<"2.Delete Element from the Queue"<<endl;  cout<<"3.Traverse the Queue"<<endl;  cout<<"4.Quit"<<endl;  cout<<"Enter your Choice: ";  cin>>choice;  switch(choice)  {  case 1:  cout<<"Enter value to be inserted into the queue: ";  cin>>item;  ql.insert(item);  break;  case 2:  ql.del();  break;  case 3:  ql.display();  break;  case 4:  exit(1);  break;  default:  cout<<"Wrong Choice"<<endl;  }  }  return 0;  }  void queue\_list::insert(int item)  {  node \*tmp;  tmp = new (struct node);  tmp->info = item;  tmp->link = NULL;  if (front == NULL)  front = tmp;  else  rear->link = tmp;  rear = tmp;  }    void queue\_list::del()  {  node \*tmp;  if (front == NULL)  cout<<"Queue Underflow"<<endl;  else  {  tmp = front;  cout<<"Element Deleted: "<<tmp->info<<endl;  front = front->link;  free(tmp);  }  }  void queue\_list::display()  {  node \*ptr;  ptr = front;  if (front == NULL)  cout<<"Queue is empty"<<endl;  else  {  cout<<"Queue elements :"<<endl;  while (ptr != NULL)  {  cout<<ptr->info<<" ";  ptr = ptr->link;  }  cout<<endl;  }  }  **-----------------------------------------------------------------------------------------**  **Output:**  **-------------**  **Operations on Queue**  **-------------**  **1.Insert Element into the Queue**  **2.Delete Element from the Queue**  **3.Traverse the Queue**  **4.Quit**  **Enter your Choice: 1**  **Enter value to be inserted into the queue: 4**  **-------------**  **Operations on Queue**  **-------------**  **1.Insert Element into the Queue**  **2.Delete Element from the Queue**  **3.Traverse the Queue**  **4.Quit**  **Enter your Choice: 1**  **Enter value to be inserted into the queue: 2**  **-------------**  **Operations on Queue**  **-------------**  **1.Insert Element into the Queue**  **2.Delete Element from the Queue**  **3.Traverse the Queue**  **4.Quit**  **Enter your Choice: 1**  **Enter value to be inserted into the queue: 5**  **-------------**  **Operations on Queue**  **-------------**  **1.Insert Element into the Queue**  **2.Delete Element from the Queue**  **3.Traverse the Queue**  **4.Quit**  **Enter your Choice: 3**  **Queue elements :**  **4 2 5**  **-------------**  **Operations on Queue**  **-------------**  **1.Insert Element into the Queue**  **2.Delete Element from the Queue**  **3.Traverse the Queue**  **4.Quit**  **Enter your Choice: 2**  **Element Deleted: 4**  **-------------**  **Operations on Queue**  **-------------**  **1.Insert Element into the Queue**  **2.Delete Element from the Queue**  **3.Traverse the Queue**  **4.Quit**  **Enter your Choice: 4** |
| **6\_Srinivas** | **/\* C++ Program to Implement Array in STL \*/**  #include <iostream>  #include <array>  #include <string>  #include <cstdlib>  using namespace std;  int main()  {  array<int, 5> arr;  array<int, 5>::iterator it;  int choice, item;  arr.fill(0);  int count = 0;  while (1)  {  cout<<"\n---------------------"<<endl;  cout<<"Array Implementation in Stl"<<endl;  cout<<"\n---------------------"<<endl;  cout<<"1.Insert Element into the Array"<<endl;  cout<<"2.Size of the array"<<endl;  cout<<"3.Front Element of Array"<<endl;  cout<<"4.Back Element of Array"<<endl;  cout<<"5.Display elements of the Array"<<endl;  cout<<"6.Exit"<<endl;  cout<<"Enter your Choice: ";  cin>>choice;  switch(choice)  {  case 1:  cout<<"Enter value to be inserted: ";  cin>>item;  arr.at(count) = item;  count++;  break;  case 2:  cout<<"Size of the Array: ";  cout<<arr.size()<<endl;  break;  case 3:  cout<<"Front Element of the Array: ";  cout<<arr.front()<<endl;  break;  case 4:  cout<<"Back Element of the Stack: ";  cout<<arr.back()<<endl;  break;  case 5:  for (it = arr.begin(); it != arr.end(); ++it )  cout <<" "<< \*it;  cout<<endl;  break;  case 6:  exit(1);  break;  default:  cout<<"Wrong Choice"<<endl;  }  }  return 0;  **}**  **-----------------------------------------------------------------------------------------**  **Output:**  **---------------------**  **Array Implementation in Stl**  **---------------------**  **1.Insert Element into the Array**  **2.Size of the array**  **3.Front Element of Array**  **4.Back Element of Array**  **5.Display elements of the Array**  **6.Exit**  **Enter your Choice: 1**  **Enter value to be inserted: 2**  **---------------------**  **Array Implementation in Stl**  **---------------------**  **1.Insert Element into the Array**  **2.Size of the array**  **3.Front Element of Array**  **4.Back Element of Array**  **5.Display elements of the Array**  **6.Exit**  **Enter your Choice: 1**  **Enter value to be inserted: 3**  **---------------------**  **Array Implementation in Stl**  **--------------------**  **1.Insert Element into the Array**  **2.Size of the array**  **3.Front Element of Array**  **4.Back Element of Array**  **5.Display elements of the Array**  **6.Exit**  **Enter your Choice: 1**  **Enter value to be inserted: 4**  **---------------------**  **Array Implementation in Stl**  **---------------------**  **1.Insert Element into the Array**  **2.Size of the array**  **3.Front Element of Array**  **4.Back Element of Array**  **5.Display elements of the Array**  **6.Exit**  **Enter your Choice: 1**  **Enter value to be inserted: 5**  **---------------------**  **Array Implementation in Stl**  **---------------------**  **1.Insert Element into the Array**  **2.Size of the array**  **3.Front Element of Array**  **4.Back Element of Array**  **5.Display elements of the Array**  **6.Exit**  **Enter your Choice: 5**  **2 3 4 5** |
| **7\_Dayanand** | **/\* C++ Program to Implement Set in STL \*/**  #include <iostream>  #include <set>  #include <string>  #include <cstdlib>  using namespace std;  int main()  {  set<int> st;  set<int>::iterator it;  int choice, item;  while (1)  {  cout<<"\n---------------------"<<endl;  cout<<"Set Implementation in Stl"<<endl;  cout<<"\n---------------------"<<endl;  cout<<"1.Insert Element into the Set"<<endl;  cout<<"2.Delete Element of the Set"<<endl;  cout<<"3.Size of the Set"<<endl;  cout<<"4.Find Element in a Set"<<endl;  cout<<"5.Dislplay by Iterator"<<endl;  cout<<"6.Exit"<<endl;  cout<<"Enter your Choice: ";  cin>>choice;  switch(choice)  {  case 1:  cout<<"Enter value to be inserted: ";  cin>>item;  st.insert(item);  break;  case 2:  cout<<"Enter the element to be deleted: ";  cin>>item;  st.erase(item);  break;  case 3:  cout<<"Size of the Set: ";  cout<<st.size()<<endl;  break;  case 4:  cout<<"Enter the element to be found: ";  cin>>item;  it = st.find(item);  if (it != st.end())  cout<<"Element "<<\*it<<" found in the set" <<endl;  else  cout<<"No Element Found"<<endl;  break;  case 5:  cout<<"Displaying Map by Iterator: ";  for (it = st.begin(); it != st.end(); it++)  {  cout << (\*it)<<" ";  }  cout<<endl;  break;  case 6:  exit(1);  break;  default:  cout<<"Wrong Choice"<<endl;  }  }  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **Set Implementation in Stl**  **---------------------**  **1.Insert Element into the Set**  **2.Delete Element of the Set**  **3.Size of the Set**  **4.Find Element in a Set**  **5.Dislplay by Iterator**  **6.Exit**  **Enter your Choice: 1**  **Enter value to be inserted: 4**  **---------------------**  **Set Implementation in Stl**  **---------------------**  **1.Insert Element into the Set**  **2.Delete Element of the Set**  **3.Size of the Set**  **4.Find Element in a Set**  **5.Dislplay by Iterator**  **6.Exit**  **Enter your Choice: 5**  **Displaying Map by Iterator: 4**  **---------------------**  **Set Implementation in Stl**  **---------------------**  **1.Insert Element into the Set**  **2.Delete Element of the Set**  **3.Size of the Set**  **4.Find Element in a Set**  **5.Dislplay by Iterator**  **6.Exit**  **Enter your Choice: 6** |
| **8\_Swetha\_H** | **/\* Program using Multiset \*/**  #include<iostream>  #include<set>  using namespace std;  int main()  {  multiset<int,less<int> >ms;  ms.insert(10);  ms.insert(20);  ms.insert(10);  cout<<"There are "<<ms.count(10);  multiset<int,less<int> >::iterator it;  it=ms.find(10);  if(it!=ms.end())  cout<<" number of 10 was found";  return 0;  }  -**----------------------------------------------------------------------------------------**  **Output:**  **There are 2 number of 10 was found** |
| **9\_Ashiwini** | **/\* C++ Program to Implement Set in STL \*/**  #include <iostream>  #include <map>  int main()  {  std::map <int, std::string> Country;  std::map <int, std::string>::const\_iterator i;  Country.insert(std::pair <int, std::string>(1, "USA"));  Country.insert(std::pair <int, std::string>(7, "Russia"));  Country.insert(std::pair <int, std::string>(33, "France"));  Country.insert(std::pair <int, std::string>(39, "Italy"));  Country.insert(std::pair <int, std::string>(49, "Germany"));  Country.insert(std::pair <int, std::string>(61, "Australia"));    std::cout << "ISD\tCountry " << std::endl;  std::cout << "---\t--------" << std::endl;  for (i = Country.begin(); i != Country.end(); i++)  {  std::cout << (\*i).first << "\t" << (\*i).second << std::endl;  }  return 0;  }  -**----------------------------------------------------------------------------------------**  **Output:**  **ISD Country**  **--- --------**  **1 USA**  **7 Russia**  **33 France**  **39 Italy**  **49 Germany**  **61 Australia** |
| **10\_Rathod** | **/\* Multimap program using STL [Standard Templete Library] \*/**  #include<iostream>  #include<map>  using namespace std;  typedef multimap<int, string> MULTIMAP;  typedef MULTIMAP::iterator ITERATOR;  int main()  {  MULTIMAP m\_map; /\* creation of multimap \*/  ITERATOR position; /\* ITERATOR to insert \*/  m\_map.insert(pair<int, string>(7, "Ram"));  m\_map.insert(pair<int , string>(3, "Sham"));  m\_map.insert(pair<int , string>(1, "Rama"));  m\_map.insert(pair<int , string>(1, "Shama"));  cout << "Multimap Output:\n" ;  for(position=m\_map.begin(); position != m\_map.end();position++)  cout << position->first << " " << position->second << "\n";  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **Multimap Output:**  **1 Rama**  **1 Shama**  **3 Sham**  **7 Ram** |
| **11\_Venketesh** | **/\* C++ Program to implement Vector with iterator using STL \*/**  #include <iostream>  #include <vector>  int main ()  {  std::vector<int> myvector;  for (int i=1; i<=5; i++) myvector.push\_back(i);  std::cout << "myvector contains:";  for (std::vector<int>::iterator it = myvector.begin() ; it != myvector.end(); ++it)  std::cout << ' ' << \*it;  std::cout << '\n';  return 0;  **}**  **-----------------------------------------------------------------------------------------**  **Output:**  **myvector contains: 1 2 3 4 5** |
| **12\_Ishaque** | **/\* C++ Program to implement List Iteration \*/**  #include<iostream>  #include<list>  #include<cstdlib>  using namespace std;  void display(list<int> &lst)  {  list<int> :: iterator p;  for(p=lst.begin(); p!=lst.end(); ++p)  cout<<\*p <<" , ";  cout<<"\n";  }  int main()  {  list<int> llist1;  list<int> llist2(5);  for(int i=0;i<5;i++)  llist1.push\_back(rand()/100);  list<int> :: iterator p;  for(p=llist2.begin(); p!=llist2.end(); ++p)  \*p=rand()/100;  cout<<"List1 :"<<endl;  display(llist1);  cout<<"List2 :"<<endl;  display(llist2);  //Add two elements at the ends of list1  llist1.push\_front(100);  llist1.push\_front(200);    //Remove an elements at the front of list2  llist2.pop\_front();  cout<<"Now List1 :"<<endl;  display(llist1);    cout<<"Now List2 :"<<endl;  display(llist2);  list<int> listA, listB;  listA=llist1;  listB=llist2;  //Merging two lists unsorted  llist1.merge(llist2);  cout<<"Merge unsorted List :"<<endl;  display(llist1);  //Sorting & merging  listA.sort();  listB.sort();  listA.merge(listB);  cout<<"Merge sorted List :"<<endl;  display(listA);  //Remove a list  listA.reverse();  cout<<"Reversed Merrged List :"<<endl;  display(listA);  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **List1 :**  **18042893 , 8469308 , 16816927 , 17146369 , 19577477 ,**  **List2 :**  **4242383 , 7198853 , 16497604 , 5965166 , 11896414 ,**  **Now List1 :**  **200 , 100 , 18042893 , 8469308 , 16816927 , 17146369 , 19577477 ,**  **Now List2 :**  **7198853 , 16497604 , 5965166 , 11896414 ,**  **Merge unsorted List :**  **200 , 100 , 7198853 , 16497604 , 5965166 , 11896414 , 18042893 , 8469308 , 16816927 , 17146369 , 19577477 ,**  **Merge sorted List :**  **100 , 200 , 5965166 , 7198853 , 8469308 , 11896414 , 16497604 , 16816927 , 17146369 , 18042893 , 19577477 ,**  **Reversed Merrged List :**  **19577477 , 18042893 , 17146369 , 16816927 , 16497604 , 11896414 , 8469308 , 7198853 , 5965166 , 200 , 100 ,** |
| **13\_Uday** | **/\* Vector search \*/**  #include <iostream>  #include <vector>  #include <algorithm>  using namespace std;  int main ()  {  vector <int> v;  v.push\_back (50);  v.push\_back (2991);  v.push\_back (23);  v.push\_back (9999);  vector <int>::iterator i = v.begin ();  while (i != v.end ()){  cout << \*i << endl;  ++ i;  }  i = find (v.begin (),v.end (), 2991);  if (i != v.end ())  {  int nPosition = distance (v.begin (), i);  cout << "Value "<< \*i;  cout << " found in the vector at position: " << nPosition << endl;  }  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **50**  **2991**  **23**  **9999**  **Value 2991 found in the vector at position: 1** |
| **14\_Anan** | **/\* Vector Sorting \*/**  **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **\* NOTE use g++ -std=c++0x vectorsort.cpp to compile \***  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**    #include <iostream>  #include <algorithm>  #include <vector>  #include <string>  using namespace std;  int main()  {  // Warning this type of initialization requires a C++11 Compiler  vector<int> intVec = {56, 32, -43, 23, 12, 93, 132, -154};  vector<string> stringVec = {"John", "Bob", "Joe", "Zack", "Randy"};  // Sorting the int vector  cout << "sorting integer data." << endl;  sort(intVec.begin(), intVec.end());  for (vector<int>::size\_type i = 0; i != intVec.size(); ++i)  cout << intVec[i] << " ";  cout << endl;  // Sorting the string vector  cout << "sorting string data"<< endl;  sort(stringVec.begin(), stringVec.end());  // Ranged Based loops. This requires a C++11 Compiler also  // If you don't have a C++11 Compiler you can use a standard  // for loop to print your vector.  for (string &s : stringVec)  cout << s << " ";  cout << endl;  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **sorting integer data.**  **-154 -43 12 23 32 56 93 132**  **sorting string data**  **Bob Joe John Randy Zack** |
| **15\_Divya\_P** | **/\* Program to find min and max using vector \*/**  #include<iostream>  #include<algorithm>  #include<vector>  using namespace std;  int main()  {  int values[] = { 100,50,14,29,18,101,67,59,1};  vector<int> v(values,values+9);  cout<< "Max Element is"<<\*max\_element(v.begin(), v.end()) << endl;  /\* prints 10 \*/    cout<< "Min Element is " << \*min\_element(v.begin(), v.end()) << endl;  /\* prints 1 \*/  }  **-----------------------------------------------------------------------------------------**  **Output:**  **Max Element is 101**  **Min Element is 1** |
| **16\_Arjun** | **/\* Program to find min and max using vector \*/**  #include<iostream>  #include<algorithm>  #include<vector>  using namespace std;  int main()  {  int values[] = { 11,56,42,99,18,1,60,25,8};  vector<int> v(values,values+9);  cout<<"max value is :"<< \*max\_element(v.begin(), v.end())<<endl;  cout<<"min value is :"<< \*min\_element(v.begin(), v.end())<<endl;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **max value is :99**  **min value is :1** |
| **17\_Shivaprasad** | **/\* C++ program to implement stack algorithm \*/** |
| **18\_Harnath** | **/\* Program 1: Object slicing \*/**  #include <iostream>  using namespace std;  class Base  {  public:  Base(int val)  {  val\_ = val;  }  void print()  {  cout<< "In Base::print() : val\_ " << val\_ <<endl;  }  private:  int val\_;  };  class Derived : public Base  {  public:  Derived(int val, int b):Base(val)  {  b\_ = b;  }  void print()  {  cout<< "In Derived::print() : b\_ " << b\_ <<endl;  }  private:  int b\_;  };  void disp (Base ob)  {  ob.print();  }  int main()  {  Base b(10);  Derived d(15, 25);  disp(b);  disp(d); // slicing will happen  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **In Base::print() : val\_ 10**  **In Base::print() : val\_ 15**  ------------------------------------------------------------------------------------------  **/\* Program 2: Queue Algorithm \*/**  **------------------------------------------------------------------------------------------**  #include <iostream>  #include<stdlib.h>  using namespace std;  class queuearr {  int queue1[5];  int rear, front;  public:  queuearr()  {  rear = -1;  front = -1;  }  void insert(int data)  {  if (rear > 4) {  cout << "queue over flow";  front = rear = -1;  return;  }  queue1[++rear] = data;  cout << "inserted " << data;  }  void delet()  {  if (front == rear) {  cout << "queue under flow";  return;  }  cout << "deleted " << queue1[++front];  }  void display()  {  if (rear == front) {  cout << " queue empty";  return;  }  for (int i = front + 1; i <= rear; i++)  cout << queue1[i] << " ";  }  };  int main()  {  int ch;  queuearr qu;  while (1) {  cout << "\n1.insert 2.delet 3.display 4.exit\nEnter ur choice: "; cin >> ch;  switch (ch) {  case 1:  cout << "enter the element: "; cin >> ch;  qu.insert(ch);  break;  case 2:  qu.delet();  break;  case 3:  qu.display();  break;  case 4:  exit(0);  }  }  }  **-----------------------------------------------------------------------------------------**  **Output:**  **1.insert 2.delet 3.display 4.exit**  **Enter ur choice: 1**  **enter the element: 4**  **inserted 4**  **1.insert 2.delet 3.display 4.exit**  **Enter ur choice: 3**  **4**  **1.insert 2.delet 3.display 4.exit**  **Enter ur choice: 4** |
| **19\_Ramya** | **/\* Sort the element using Deque\*/**  #include <iostream>  #include <vector>  #include <deque>  #include <list>  #include <set>  #include <map>  #include <string>  #include <algorithm>  #include <iterator>  #include <functional>  #include <numeric>  template <class T>  inline void PRINT\_ELEMENTS (const T& coll, const char\* optcstr="")  {  typename T::const\_iterator pos;  std::cout << optcstr;  for (pos=coll.begin(); pos!=coll.end(); ++pos) {  std::cout << \*pos << ' ';  }  std::cout << std::endl;  }  template <class T>  inline void INSERT\_ELEMENTS (T& coll, int first, int last)  {  for (int i=first; i<=last; ++i) {  coll.insert(coll.end(),i);  }  }  using namespace std;  int main()  {  deque<int> coll;  INSERT\_ELEMENTS(coll,1,9);  INSERT\_ELEMENTS(coll,1,9);  PRINT\_ELEMENTS(coll,"on entry: ");  sort (coll.begin(), coll.end());  PRINT\_ELEMENTS(coll,"sorted: ");  }  **-----------------------------------------------------------------------------------------**  **Output:**  **on entry: 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9**  **sorted: 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9** |
| **20\_Sandeep** | **/\* C++ Progra to implement Set using STL \*/**  #include <iostream> #include <set> #include <algorithm> #include <iterator> // ostream\_iterator using namespace std; int main() { double a[ 5 ] = { 2.1, 4.2, 9.5, 2.1, 3.7 }; set< double, less< double > > doubleSet( a, a + 4);; ostream\_iterator< double > output( cout, " " );  cout << "doubleSet contains: "; copy( doubleSet.begin(), doubleSet.end(), output );  cout << endl; return 0; }  **-----------------------------------------------------------------------------------------**  **Output:**  **doubleSet contains: 2.1 4.2 9.5** |
| **21\_Deepika** | **/\* C++ Progra to implement List with iterator using STL \*/**  #include <iostream>  #include <list>  using namespace std;  int main ()  {  int myints[] = {75,23,65,42,13,90};  list<int> mylist (myints,myints+6);  cout << "mylist contains:";  for (list<int>::iterator it=mylist.begin(); it != mylist.end(); ++it)  cout << ' ' << \*it;  cout << '\n';  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **mylist contain 75 23 65 42 13 90** |
| **22\_Saikrishna** | **/\* C++ Program to implement Map algorithm using STL \*/** |
| **23\_Harish** | **/\* Insertion of member function using Multimap in STL \*/**  #include <iostream>  #include <map>  int main ()  {  std::multimap<char,int> mymultimap;  mymultimap.insert (std::pair<char,int>('a',10));  mymultimap.insert (std::pair<char,int>('b',20));  mymultimap.insert (std::pair<char,int>('b',150));  // show content:  for (std::multimap<char,int>::iterator it=mymultimap.begin(); it!=mymultimap.end(); ++it)  std::cout << (\*it).first << " => " << (\*it).second << '\n';  return 0;  }  **-----------------------------------------------------------------------------------------**  **Output:**  **a => 10**  **b => 20**  **b => 150** |