Program 1:

//Vector Sequence Container

#include<iostream>

#include<conio.h>

#include<vector>

using namespace std;

void display(vector <int> &); // display function prototype

int main()

{

vector <int> v; // integer vector created

cout<<"\n\nInitial size() = "<<v.size(); // gives no of elements

cout<<"\n\nInitial capacity() = "<<v.capacity(); // capacity returns no elements that vector can store b4 that vector needs to dynamically resize itself to accommodate more elements

v.push\_back(10); // pushing the element at back of vector

v.push\_back(20);

v.push\_back(30);

v.push\_back(40);

v.push\_back(50);

cout<<"\n\nAfter push\_back() size() = "<<v.size();

cout<<"\n\nAfter push\_back() capacity() = "<<v.capacity();

cout<<"\n\nDisplay vector elements after push\_back() :";

display(v);

cout<<"\n\nFrist element of vector = "<<v.front();

cout<<"\n\nLast element of vector = "<<v.back();

//Inserting elements in vector using iterator

vector<int>::iterator itr=v.begin(); //here itr is pointing to 0th element of v

itr = itr + 5; // itr made to point 4th element;

v.insert(itr,60); // insert 40 as 4th element of v

cout<<"\n\nDisplay vector elements after insertion :";

display(v);

//pop\_back() function to delete last element

v.pop\_back();

cout<<"\n\nDisplay vector elements after pop\_back() :";

display(v);

// erase(delete) vector elements

v.erase(v.begin()+2,v.begin()+4); // erase(2,4) = deletes 30 & 40 but not 50

cout<<"\n\nDisplay vector elements after erase() :";

display(v);

//resizing vector

v.resize(10);

cout<<"\n\nAfter resize() vector size = "<<v.size();

//using clear function

v.clear();

cout<<"\n\nAfter clear() function :";

display(v);

cout<<"\n\nIs vector empty = "<<v.empty();

getch();

return 0;

}

void display(vector <int> & v)

{

for(int i=0;i<v.size();i++)

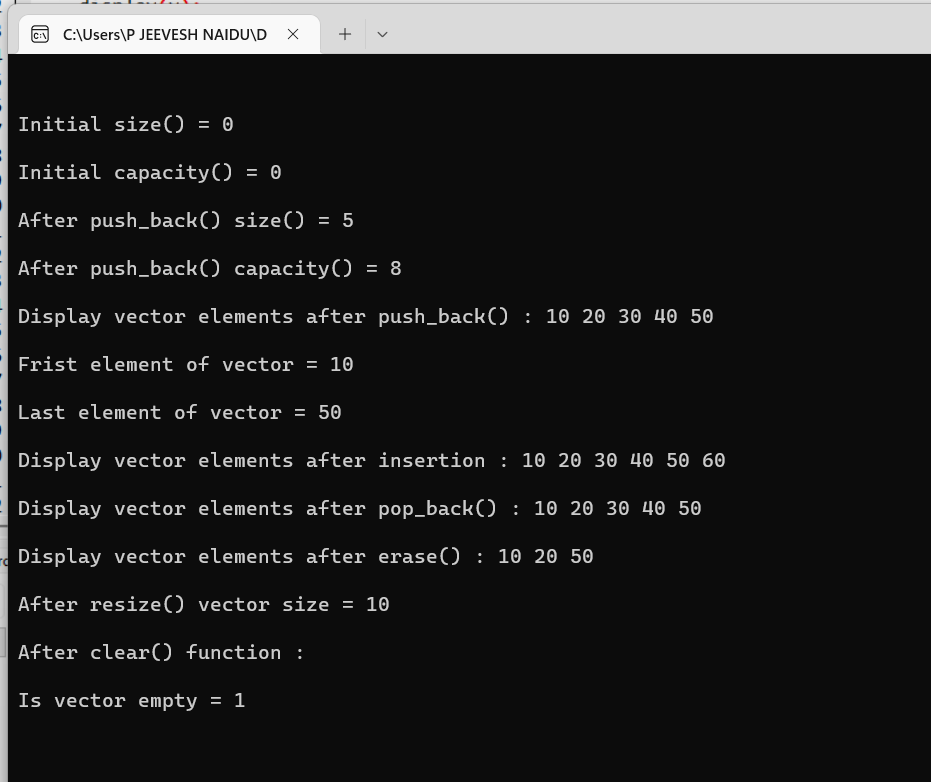
{

cout<<" "<<v.at(i); // at() prints vector element at each reference index

}

}

Output:



Program 2:

//List sequence Container

#include<iostream>

#include<conio.h>

#include<list> //linear linked list

using namespace std;

void display(list <int> &); // display function prototype

int main()

{

list <int> list1; //empty list1 of zero length

list <int> list2; //empty list2

list <int> list3;

cout<<"\n\nsize of list1 = "<<list1.size();

list1.push\_front(2);

list1.push\_front(1);

list1.push\_back(3);

list1.push\_back(4); //1234

cout<<"\n\nList1 elements after push\_front() and push\_back() :";

display(list1);

//Remove an element from front end

list1.pop\_front(); // same way pop\_back()

cout<<"\n\nAfter removing front element of list1 : ";

display(list1); //234

//insert an element(1) at the begining of list1

list1.insert(list1.begin(),1);

cout<<"\n\nAfter inserting an element at begining of list1 : ";

display(list1); //1234

// pushing elements in list2

list2.push\_front(5);

list2.push\_front(6);

list2.push\_back(9);

list2.push\_back(8);

list2.push\_back(7); // 65987

cout<<"\n\nList2 elements after push\_front() and push\_back() :";

display(list2);

//sorting list2 elements

list2.sort();

cout<<"\n\nList2 elements after sorting :";

display(list2); //56789

//splice(insert) the elements of list2 at the end of list1

list1.splice(list1.end(),list2); // similarly splice could be at begin() also

cout<<"\n\nList1 elements after splicing :";

display(list1); //123456789

//merging list1 contents into list3

list3.merge(list1);

cout<<"\n\nList3 elements after merging :";

display(list3);

//reverse a list

list3.reverse();

cout<<"\n\nList3 elements after reversing :";

display(list3);

list3.push\_back(9);

list3.push\_back(9);

cout<<"\n\nUpdated list3 elements :";

display(list3);

//Removing duplicates from list3 elements

list3.unique();

cout<<"\n\nAfter removing duplicates from list3 elements are :";

display(list3);

//remove all 9's from list3

list3.remove(9);

cout<<"\n\nAfter removing all 9's from list3 elements :";

display(list3);

//using swap() function

list2.swap(list3);

cout<<"\n\nAfter swapping list2 : ";

display(list2);

cout<<"\n\nAfter swapping list3 : ";

display(list3);

//Using assign() function

list3.assign(list2.begin(),list2.end());

cout<<"\n\nAfter assiging list2 elements to lsit3 :";

display(list3);

getch();

return 0;

}

void display(list <int> & v)

{

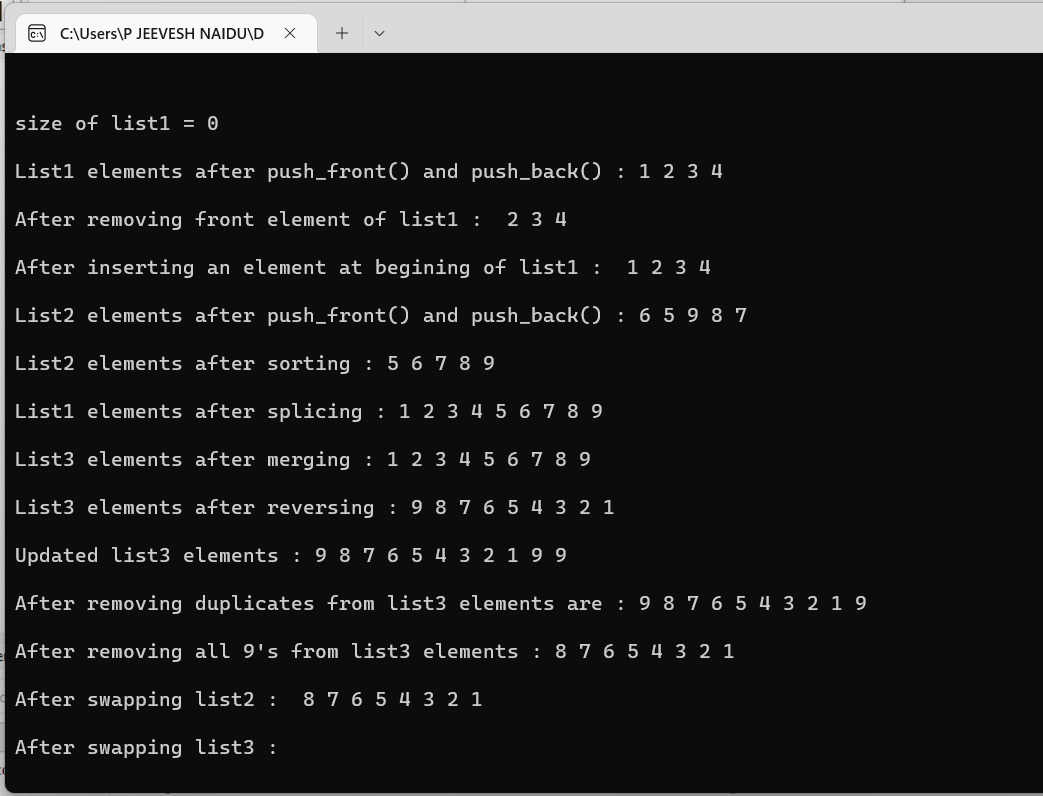
list<int> :: iterator p;

for(p = v.begin(); p!=v.end(); ++p)

cout<<" "<<\*p;

}

Output:



Program 3:

//Dequeu Sequence Container

#include<iostream>

#include<conio.h>

#include<deque>

using namespace std;

void display(deque <double> &); // display function prototype

int main()

{

deque <double> d;

//insert elements in d

d.push\_front(2.2);

d.push\_front(3.5);

d.push\_back(1.1); // 3.5 2.2 1.1

cout<<"\n\nDeque elements after insertion are as follows :";

display(d);

//pop\_front() to remove front element

d.pop\_front(); // d.pop\_back();

cout<<"\n\nDeque elements after pop\_front() are as follows :";

display(d);

//using [] subsricpt operator to modify elements

d[1]=3.3; // 1.1 gets overwritten by 3.3

cout<<"\n\nDeque elements after subscript insertion using [] are as follows :";

display(d);

getch();

return 0;

}

void display(deque <double> & d1)

{

for(int i=0;i<d1.size();i++)

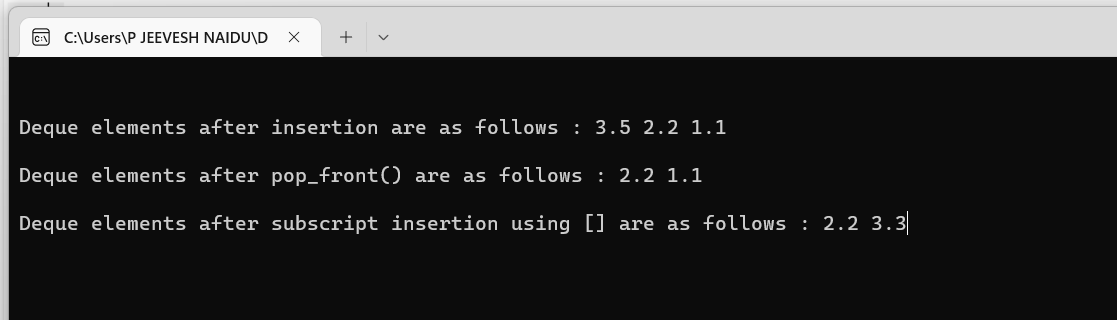
{

cout<<" "<<d1[i];

}

}

Output:



Program 4:

//Standard Library stack adapter class {can be implemented as vector,deque,list}

#include<iostream>

#include<conio.h>

#include<vector>

#include<list>

#include<deque>

#include<stack>

using namespace std;

template<class T> void pushelement(T & s); //pushelement() function prototype

template<class T> void popelement(T & s); //popelement() function prototype

int main()

{

stack <int> dequestack; // stack with default underlying deque

stack <int,vector<int> > vectorstack; //stack with underlying int vector

stack <int,list<int> > liststack; //stack with underlying int list

//push 10 elements on each of these stacks (i.e dequestack,vectorstack,liststack)

cout<<"\n\npushing elements onto dequestack : ";

pushelement(dequestack);

cout<<"\n\npushing elements onto vectorstack : ";

pushelement(vectorstack);

cout<<"\n\npushing elements onto liststack : ";

pushelement(liststack);

cout<<"\n\n------------------------------------------------";

cout<<"\n\npopping element from dequestack : ";

popelement(dequestack);

cout<<"\n\npopping element from vectorstack : ";

popelement(vectorstack);

cout<<"\n\npopping element from liststack : ";

popelement(liststack);

getch();

return 0;

}

//Function definition for pushelement()

template<class T>

void pushelement(T & s)

{

for(int i=0;i<10;i++)

{

s.push(i);

cout<<s.top()<<" ";

}

}

//Function definition for popelement()

template<class T>

void popelement(T & s)

{

while(!s.empty())

{

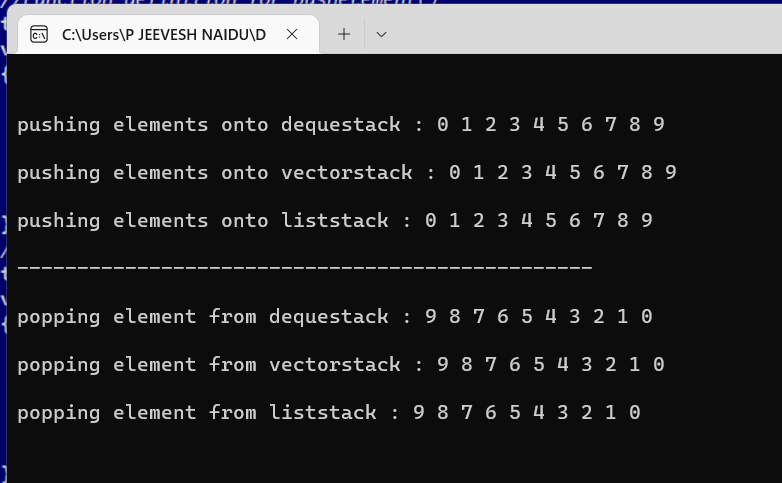
cout<<s.top()<<" ";

s.pop();

}

}

Output:



Program 5:

//Standard Library queue adapter class

#include<iostream>

#include<conio.h>

#include<queue>

using namespace std;

int main()

{

queue <double> q; // double type queue "q" created (initially empty)

//push elements onto queue

q.push(1.1); // 1.1 2.2 3.3

q.push(2.2);

q.push(3.3);

cout<<"\n\nPopping elements from queue : ";

while(!q.empty())

{

cout<<q.front()<<" ";

q.pop();

}

getch();

return 0;

}

Output:

