Q1. Inserts an element. And returns an iterator that points to the first of the newly inserted elements.

#include <iostream>

#include <iterator>

#include <deque>

using namespace std;

int main()

{

deque <int> d = {10, 20, 30};

auto it = d.insert(d.begin()+1, 100);

cout<<\*it<<endl;

}

Q2. Returns a reverse iterator which points to the last element of the deque (i.e., its reverse beginning).

#include <iostream>

#include <deque>

#include <iterator>

using namespace std;

int main()

{

deque <int> d = {10, 20, 30, 40, 50};

deque <int>::reverse\_iterator rit = d.rbegin();

for(int i = 0; i < d.size(); i++, rit++)

{

cout<<\*rit<<" ";

}

}

Q3. Returns a reverse iterator which points to the position before the beginning of the deque (which is considered its reverse end).

#include <iostream>

#include <deque>

#include <iterator>

using namespace std;

int main()

{

deque <int> d = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

deque <int>::reverse\_iterator rit = d.rend();

cout<<\*rit;

}

Q4. Returns a constant iterator pointing to the first element of the container, that is, the

iterator cannot be used to modify, only traverse the deque.

#include <iostream>

#include <deque>

#include <iterator>

using namespace std;

int main()

{

deque <int> f ={10, 20, 30, 40, 50};

deque <int>::const\_iterator it = f.begin();

while(it != f.end())

{

cout<<\*it<<" ";

it++;

}

}

/\*Q5. Returns the maximum number of elements that a deque container can hold.\*/

#include <iostream>

#include <deque>

using namespace std;

int main()

{

deque <int> f = {1, 2, 3, 4, 5};

cout<<f.max\_size();

}

Q6. Assign values to the same or different deque container.

#include <iostream>

#include <deque>

using namespace std;

int main()

{

deque <int> f1 = {10, 20, 30, 40, 50};

deque <int> f2(f1);

for(int x : f2) cout<<x<<" ";

cout<<endl;

deque <int> f3;

f3.assign(f1.begin(), f1.end());

for(int x : f3) cout<<x<<" ";

cout<<endl;

}

Q7. Return the first element and last element of the deque container.

#include <iostream>

#include <deque>

using namespace std;

int main()

{

deque <int> l = {3, 2, 9};

cout<<l.front()<<endl;

cout<<l.back();

}

Q8. Remove elements from a container from the specified position or range in deque.

#include <iostream>

#include <deque>

#include <iterator>

using namespace std;

int main()

{

deque <int> l = {1, 2, 3, 4, 5, 6, 7};

int pos = 0;

cout<<"enter position = ";

cin>>pos;

deque <int>::iterator it = l.begin() + pos;

cout<<\*it;

}

Q9. Generate a permutation of first N natural numbers having count of unique adjacent differences equal to K | Set 2 using a queue.

#include <iostream>

#include <deque>

#include <iterator>

using namespace std;

int main()

{

deque <int> f1;

int N = 0, K = 0, flag = 0;

cout<<"Enter value of n = ";

cin>>N;

cout<<"Enter value of k = ";

cin>>K;

if(K < N)

{

for(int i = 1; i <= N; i++)

{

f1.push\_back(i);

}

for(; K != 0; K--)

{

if(flag == 0)

{

cout<<f1.front()<<" ";

f1.pop\_front();

flag = 1;

}

else

{

cout<<f1.back()<<" ";

f1.pop\_back();

flag = 0;

}

}

if(!f1.empty())

{

for(int x : f1) cout<<x<<" ";

}

}

else

{

cout<<"K should be less then N";

}

}

Q10. Check if given Strings can be made equal by inserting at most 1 String using deque.

#include <iostream>

#include <deque>

#include <iterator>

using namespace std;

deque <char> charDeque(deque <string> s1);

deque <char> areSame(deque <char> X, deque <char> Y);

int main()

{

deque <string> s1 = {"My Name Is Yash"};

deque <string> s2 = {"My Yash"};

deque <char> X, Y;

X = charDeque(s1);

Y = charDeque(s2);

Y = areSame(X, Y);

if(X.size() == Y.size())

{

for(char x : X) cout<<x;

cout<<endl;

for(char x : Y) cout<<x;

}

else

{

cout<<"string 2 is wrong";

}

}

deque <char> charDeque(deque <string> s1)

{

string word = s1[0];

deque <char> X;

int size = word.length();

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

{

X.push\_back(word[i]);

}

return X;

}

deque <char> areSame(deque <char> X, deque <char> Y)

{

int flag = 1;

int i = 0;

int Xj = X.size() - 1;

int Yj = Y.size() - 1;

//checking first word

while(true)

{

if(X[i] == Y[i])

{

if(X[i] == ' ')

{

break;

}

else

{

i = i + 1;

}

}

else

{

return Y;

}

}

//checking last word

while(true)

{

if(X[Xj] == Y[Yj])

{

if(X[Xj] == ' ')

{

break;

}

else

{

Xj = Xj - 1;

Yj = Yj - 1;

}

}

else

{

return Y;

}

}

deque <char>::iterator Yit = Y.begin() + i + 1;

deque <char>::iterator Xit\_begin = X.begin() + i + 1;

deque <char>::iterator Xit\_end = X.begin() + Xj + 1;

Y.insert(Yit, Xit\_begin, Xit\_end);

return Y;

}

Q11. How to get the first and last elements of Deque in c++?

#include <iostream>

#include <deque>

using namespace std;

int main()

{

deque <int> l = {1, 2, 3, 4, 5, 6, 7};

cout<<"first = "<<l.front()<<endl;

cout<<"last = "<<l.back();

}

Q12. Given a string S containing letters and ‘#‘. The ‘#” represents a backspace. The task

is to print the new string without ‘#‘. String after processing backspace characters

using deque

Examples:

Input : S = "abc#de#f#ghi#jklmn#op#"

Output : abdghjklmo

Input : S = "##iNeuron##Education##hub#"

Output : iNeurEducatihu

#include <iostream>

#include <deque>

#include <string>

using namespace std;

int main()

{

string S = "##iNeuron##Education##hub#";

deque <char> d;

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

{

if(S[i] == '#')

{

if(d.empty())

{

continue;

}

else

{

d.pop\_back();

}

}

else

{

d.push\_back(S[i]);

}

}

for( char x : d)cout<<x;

}

Q13. Segregate even and odd nodes in a Linked List using Deque.

#include <iostream>

#include <deque>

#include <list>

#include <iterator>

using namespace std;

int main()

{

list <int> l = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

deque <int> d;

list <int>::iterator lit = l.begin();

for(; lit != l.end(); lit++)

{

if( (\*lit) % 2 > 0 )

{

d.push\_front(\*lit);

}

else

{

d.push\_back(\*lit);

}

}

cout<<"odd = ";

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

{

if(d[i] % 2 > 0)

cout<<d[i];

}

cout<<endl<<"even = ";

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

{

if(d[i] % 2 == 0)

cout<<d[i];

}

}