Q1. Create a stack of int type, push 5 elements in it and print it on the console screen.

#include <iostream>

#include <stack>

using namespace std;

int main()

{

stack <int> s;

int x = 0;

cout<<"Enter 5 elements"<<endl;

for(int i = 0; i < 5; i++)

{

cout<<i<<") = ";

cin>>x;

s.push(x);

}

cout<<endl<<"Popping elements from stack"<<endl;

while(!s.empty())

{

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

s.pop();

}

}

Q2. Create a stack of int type, and find the top most element in a stack.

#include <iostream>

#include <stack>

#include <vector>

using namespace std;

int main()

{

vector <int> v = {10, 20, 30, 40, 50};

stack <int, vector<int>> s(v);

cout<<"Top most element = "<<s.top();

}

Q3. Create a stack, and implement main operations like push(), pop(), peek(), empty() and size().

#include <iostream>

#include <stack>

using namespace std;

int main()

{

stack <int> s;

cout<<"pushing elements 10, 20, 30"<<endl;

s.push(10);

s.push(20);

s.push(30);

cout<<endl<<"popping element = "<<s.top()<<endl;

s.pop();

cout<<endl<<"Checking stack is empty or not = ";

if(s.empty())

cout<<"stack is empty";

else

cout<<"stack is not empty";

cout<<endl<<endl<<"stack size = "<<s.size();

}

Q4. Reverse the Words of a String using Stack.

Example:

Input: str = “I Love To Code”

Output: Code To Love I

#include <iostream>

#include <stack>

#include <vector>

#include <string>

using namespace std;

int main()

{

string str= "I love to code";

string word;

stack <string> s;

int start = 0, space = 0;

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

{

if(str[i] == ' ' || i == str.size())

{

space = space + 1;

word = str.substr(start, i - start);

if(space == 1)

{

s.push(word);

start = i + 1;

}

else

{

s.push(word + ' ');

start = i + 1;

}

}

}

while(s.size())

{

cout<<s.top();

s.pop();

}

}

Q5. Create stack1 of int type, and create another stack of the same type with name

stack2 and copy all the elements of stack1 into stack2 in the same order.

#include <iostream>

#include <stack>

#include <vector>

using namespace std;

int main()

{

stack <int> stack1;

stack <int> stack2;

vector <int> v;

int x = 0;

for(int i = 0; i < 5; i++)

{

cout<<i<<") = ";

cin>>x;

stack1.push(x);

}

while(!stack1.empty())

{

v.push\_back(stack1.top());

stack1.pop();

}

while(!v.empty())

{

stack1.push(v.back());

stack2.push(v.back());

v.pop\_back();

}

cout<<"stack1 size = "<<stack1.size()<<endl;

cout<<"stack2 size = "<<stack2.size()<<endl;

cout<<"stack1 elements = ";

while(!stack1.empty())

{

cout<<stack1.top()<<" ";

stack1.pop();

}

cout<<endl;

cout<<"stack2 elements = ";

while(!stack2.empty())

{

cout<<stack2.top()<<" ";

stack2.pop();

}

}

Q6. Reverse a string using a stack.

Example:

Input: str = "Reverse me"

Output: em esreveR

#include <iostream>

#include <string>

#include <stack>

using namespace std;

int main()

{

stack <char> s;

string str = "Reverse me";

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

s.push(str[i]);

while(!s.empty())

{

cout<<s.top();

s.pop();

}

}

Q7. Create a stack of int type and sort it.

#include <iostream>

#include <vector>

#include <stack>

#include <iterator>

using namespace std;

int main()

{

stack <int> s;

vector <int> v;

vector <int>::iterator it;

for(int i = 1; i <= 5; i++)

{

s.push(i \* 100);

}

cout<<"Before sorting = ";

while(!s.empty())

{

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

v.push\_back( s.top() );

s.pop();

}

it = v.begin();

while(it != v.end())

{

s.push(\*it);

it++;

}

cout<<endl<<"After sorting = ";

while(!s.empty())

{

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

s.pop();

}

}

Q8. Create a stack of int type and sort it in descending order.

#include <iostream>

#include <stack>

#include <vector>

using namespace std;

int main()

{

vector <int> v = {5, 4, 3, 2, 1};

stack <int, vector <int>> s(v);

cout<<"Before sorting = ";

while(!s.empty())

{

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

s.pop();

}

vector <int>::reverse\_iterator rit = v.rbegin();

while(rit != v.rend())

{

s.push(\*rit);

rit++;

}

cout<<endl<<"After sorting = ";

while(!s.empty())

{

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

s.pop();

}

}

Q9. Create back button functionality using stack.

#include <iostream>

#include <stack>

#include <limits>

using namespace std;

int main()

{

stack <string> s;

int ch = 0;

while(ch != 3)

{

cout<<"1. Go to next page"<<endl;

cout<<"2. Go to back page"<<endl;

cout<<"3. Exit"<<endl<<endl;

cout<<"Enter choice = ";

cin>>ch;

switch(ch)

{

case 1:

{

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

string str;

cout<<"Enter page name = ";

getline(cin, str);

s.push(str);

cout<<"Your are in "<<str<<" page"<<endl;

cin.get();

break;

}

case 2:

{

if(!s.empty())

{

if(s.size() == 1)

{

cout<<"You are in home page"<<endl;

s.pop();

}

else

{

s.pop();

cout<<"You are in "<<s.top()<<" page"<<endl;

}

}

else

{

cout<<"page not added"<<endl;

}

break;

}

case 3:

{

exit(0);

}

}

}

}

Q10. Given an array, print the Next Greater Element (NGE) for every element using stack.

Example:

Input: arr[] = [ 4 , 5 , 2 , 25 ]

Output: 4 –> 5

5 –> 25

2 –> 25

25 –> -1

#include <iostream>

#include <stack>

using namespace std;

int main()

{

int arr[] = {4, 5, 2, 25};

stack <int> s;

int j = 3;

for(int i = 3; i > -1; i--)

{

if(i == 3)

{

s.push(-1);

}

else

{

if(arr[i] < arr[j])

{

s.push(arr[j]);

j = j - 1;

}

else

{

for(int k = j + 1; k < 4; k++)

{

if(arr[i] < arr[k])

{

s.push(arr[k]);

j = j - 1;

}

}

}

}

}

for( int i = 0; !s.empty(); i++)

{

cout<<arr[i]<<" -> "<<s.top()<<endl;

s.pop();

}

}