*Muhammad Mufeez*

*BCS-3J*

***Coal Lab 06***

**Q1:**

*Code:*

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

*using namespace std;*

*template <typename T>*

*class Stack*

*{*

*T \*arr;*

*int size;*

*int top;*

*public:*

*Stack(int s)*

*{*

*size = s;*

*top = -1;*

*arr = new T[size];*

*}*

*void push(T c)*

*{*

*if (top < size - 1)*

*{*

*arr[++top] = c;*

*}*

*else*

*{*

*cout << "Stack OverFlow" << endl;*

*}*

*}*

*void pop()*

*{*

*if (top >= 0)*

*{*

*top--;*

*}*

*else*

*{*

*cout << "Stack is Empty" << endl;*

*}*

*}*

*T peek()*

*{*

*if (top >= 0 && top < size)*

*{*

*return arr[top];*

*}*

*else*

*{*

*cout << "Stack is Empty" << endl;*

*return -1;*

*}*

*}*

*bool isEmpty()*

*{*

*return top == -1;*

*}*

*};*

*int main()*

*{*

*string st = "BORRORROB";*

*Stack<char> st1(st.length());*

*string ans;*

*for(int i=0;i<st.length();++i){*

*st1.push(st[i]);*

*}*

*for(int i=st.length();i>0;--i){*

*ans += st1.peek();*

*st1.pop();*

*}*

*if (ans==st)*

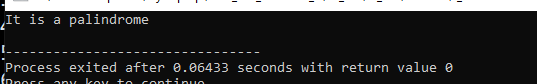
*cout << "It is a palindrome" << endl;*

*else*

*cout << "It is not a palindrome" << endl;*

*}*

*Output:*

**

**Q2:**

*Code:*

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

using namespace std;

template <typename T>

class Stack

{

T *\**arr;

int size;

int top;

public:

*Stack*(int s)

{

size = s;

top = -1;

arr = new T[size];

}

void *push*(T c)

{

*if* (top < size - 1)

{

arr[++top] = c;

}

*else*

{

cout *<<* "Stack OverFlow" *<<* *endl*;

}

}

void *pop*()

{

*if* (top >= 0)

{

top--;

}

*else*

{

cout *<<* "Stack is Empty" *<<* *endl*;

}

}

T *peek*()

{

*if* (top >= 0 && top < size)

{

*return* arr[top];

}

*else*

{

cout *<<* "Stack is Empty" *<<* *endl*;

*return* *T*();

}

}

bool *isEmpty*()

{

*return* top == -1;

}

};

int *main*()

{

Stack<string> *stack1*(5);

stack1.*push*("Water the plants");

stack1.*push*("Do the lab tasks :(");

stack1.*pop*();

*if*(stack1.*isEmpty*()){

cout*<<*"Nothing in the stack"*<<endl*;

}

*else*{

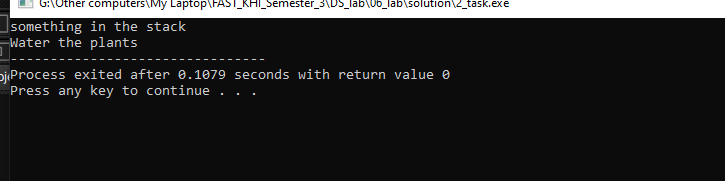
cout*<<*"something in the stack"*<<endl*;

}

cout*<<*stack1.*peek*();

}

*Output:*

**

**Q3:**

*Code:*

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

using namespace std;

template <typename T>

class Node

{

public:

T url;

Node *\**next;

Node *\**prev;

*Node*(T url)

{

this->url = url;

this->next = *NULL*;

this->prev = *NULL*;

}

};

template <typename T>

class LLStack

{

Node<T> *\**top;

public:

*LLStack*(T url)

{

top = new *Node*<T>(url);

}

void *visit*(T homepage)

{

Node<T> *\**newNode = new *Node*<T>(homepage);

top->next = newNode;

newNode->prev = top;

top = newNode;

}

T *back*(int steps)

{

*while* (steps)

{

*if* (top->prev)

top = top->prev;

*else*

*break*;

steps--;

}

*return* top->url;

}

T *forward*()

{

*if* (top->next)

top = top->next;

*return* top->url;

}

T *current*()

{

*if* (!top)

*return* *T*();

*return* top->url;

}

*~LLStack*()

{

*while* (top && top->prev)

{

Node<T> *\**prev = top->prev;

top = top->prev;

delete prev;

}

}

};

int *main*()

{

LLStack<string> *web1*("Google.cam");

web1.*visit*("Facebook.com");

web1.*visit*("Twitter.com");

web1.*visit*("Linkedin.com");

web1.*visit*("Instagram.com");

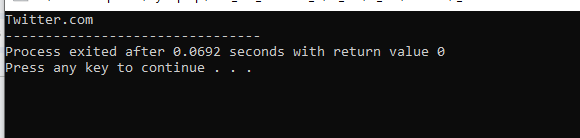
web1.*back*(2);

cout *<<* web1.*current*();

*return* 0;

}

*Output:*

**

**Q4:**

*Code:*

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

using namespace std;

template <typename T>

class Stack

{

T *\**arr;

int size;

int top;

public:

*Stack*(int s)

{

size = s;

top = -1;

arr = new T[size];

}

void *push*(T c)

{

*if* (top < size - 1)

{

arr[++top] = c;

}

*else*

{

cout *<<* "Stack OverFlow" *<<* *endl*;

}

}

void *pop*()

{

*if* (top >= 0)

{

top--;

}

*else*

{

cout *<<* "Stack is Empty" *<<* *endl*;

}

}

T *peek*()

{

*if* (top >= 0 && top < size)

{

*return* arr[top];

}

*else*

{

// *cout << "Stack is Empty" << endl;*

*return* *T*();

}

}

bool *isEmpty*()

{

*return* top == -1;

}

};

int *returnPrecedence*(char c)

{

*if* (c == '^')

{

*return* 3;

}

*else* *if* (c == '\*' || c == '/')

{

*return* 2;

}

*else* *if* (c == '+' || c == '-')

{

*return* 1;

}

*return* -1;

}

string *infixToPostfix*(string exp)

{

int len = exp.*length*();

string ans = "";

Stack<char> *stack*(len);

*for* (int i = 0; i < len; ++i)

{

*if* ((exp*[*i*]* >= 'a' && exp*[*i*]* <= 'z'))

{

ans *+=* exp*[*i*]*;

}

*else* *if* (exp*[*i*]* == '(')

{

stack.*push*(exp*[*i*]*);

}

*else* *if* (exp*[*i*]* == ')')

{

*while* (!stack.*isEmpty*() && stack.*peek*() != '(')

{

ans *+=* stack.*peek*();

stack.*pop*();

}

*if* (!stack.*isEmpty*())

stack.*pop*();

}

*else*

{

*while* ((*returnPrecedence*(stack.*peek*()) >= *returnPrecedence*(exp*[*i*]*)) && !stack.*isEmpty*())

{

ans *+=* stack.*peek*();

stack.*pop*();

}

stack.*push*(exp*[*i*]*);

}

}

*while* (!stack.*isEmpty*())

{

ans *+=* stack.*peek*();

stack.*pop*();

}

*return* ans;

}

int *main*()

{

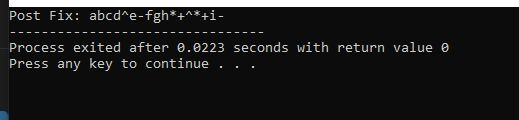
string equation = "a+b\*(c^d-e)^(f+g\*h)-i";

cout *<<* "Post Fix: " *<<* *infixToPostfix*(equation);

*return* 0;

};

*Output:*

**

**Q5:**

*Code:*

*Output:*

**Q6:**

*Code:*

*Output:*

**Q7:**

*Code:*

*Output:*