# ASSIGNMENT-3 DSA

### TOPIC-STACK

Name-Meet

Roll\_no-1024030343

Batch-2C24



Ques-1)

#include<iostream>

using namespace std;

class stack

{

int \*arr;

int size;

int top=-1;

public:

stack(int capacity)

{

arr=new int[capacity];

size=capacity;

}

void push(int value)

{

if(top!=size-1)

arr[++top]=value;

else

cout<<"stack is already empty"<<endl;

}

void isFull()

{

if(top==size-1)

cout<<"stack is already full you can't add now"<<endl;

else

cout<<"keep on adding"<<endl;

}

void pop()

{

if(top!=-1)

top--;

else

cout<<"staxk is already empty, what you wanna pop add something bro first"<<endl;

}

void clear()

{

if(top==-1)

cout<<"stack is already empty"<<endl;

else

top=-1;

}

void isEmpty()

{

if(top==-1)

cout<<"stack has been cleared successfully"<<endl;

else

cout<<"error in clear function"<<endl;

}

void display()

{

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

{

cout<<i+1<<" "<<"element:"<<arr[i]<<endl;

}

}

};

int main()

{

stack s(11);

s.push(10);

s.push(20);

s.push(30);

s.push(40);

s.push(50);

s.isFull();

cout<<"before popping"<<endl;

s.display();

s.pop();

s.pop();

cout<<"after popping"<<endl;

s.display();

s.clear();

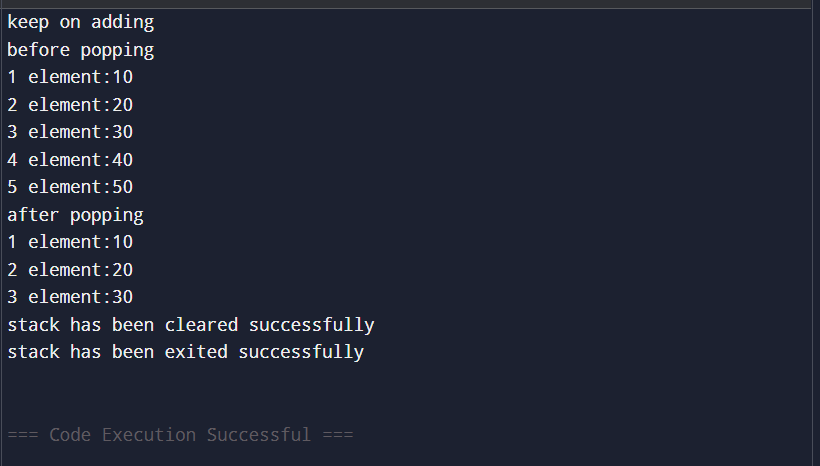
s.isEmpty();

s.exit();

return 0;

}

OUTPUT-



Ques-2)

#include <iostream>

#include <stack>

using namespace std;

int main() {

string input = "DataStructure";

stack<char> s;

for (char c : input) {

s.push(c);

}

string reversed;

while (!s.empty()) {

reversed += s.top();

s.pop();

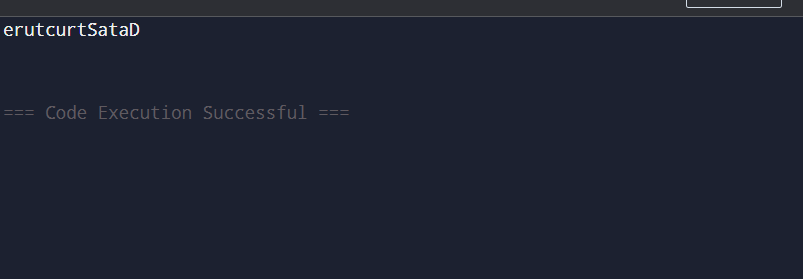
}

cout << reversed << endl;

return 0;

}

OUTPUT->



Ques-3)

#include <iostream>

#include <stack>

using namespace std;

bool isBalanced(string expr) {

stack<char> s;

for (char c : expr) {

if (c == '(' || c == '{' || c == '[') {

s.push(c);

} else if (c == ')' || c == '}' || c == ']') {

if (s.empty()) return false;

char top = s.top();

s.pop();

if ((c == ')' && top != '(') ||

(c == '}' && top != '{') ||

(c == ']' && top != '[')) {

return false;

}

}

}

return s.empty();

}

int main() {

string expr;

cout << "Enter expression: ";

cin >> expr;

if (isBalanced(expr))

cout << "Balanced" << endl;

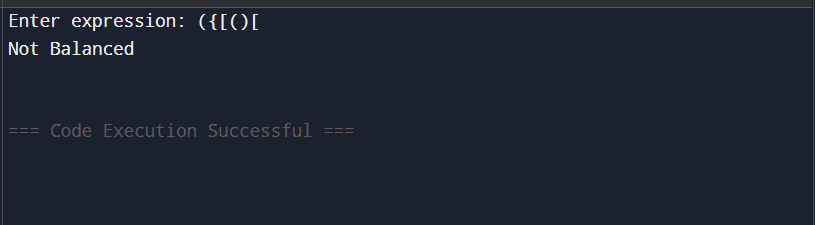
else

cout << "Not Balanced" << endl;

return 0;

}

OUTPUT-



Ques- 4)

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int precedence(char op) {

if (op == '^') return 3;

if (op == '\*' || op == '/') return 2;

if (op == '+' || op == '-') return 1;

return 0;

}

bool isOperator(char c) {

return c == '+' || c == '-' || c == '\*' || c == '/' || c == '^';

}

string infixToPostfix(string infix) {

stack<char> s;

string postfix;

for (char c : infix) {

if (isalnum(c)) postfix += c;

else if (c == '(') s.push(c);

else if (c == ')') {

while (!s.empty() && s.top() != '(') {

postfix += s.top();

s.pop();

}

if (!s.empty()) s.pop();

}

else if (isOperator(c)) {

while (!s.empty() && precedence(s.top()) >= precedence(c)) {

postfix += s.top();

s.pop();

}

s.push(c);

}

}

while (!s.empty()) {

postfix += s.top();

s.pop();

}

return postfix;

}

int main() {

string infix;

cout << "Enter infix expression: ";

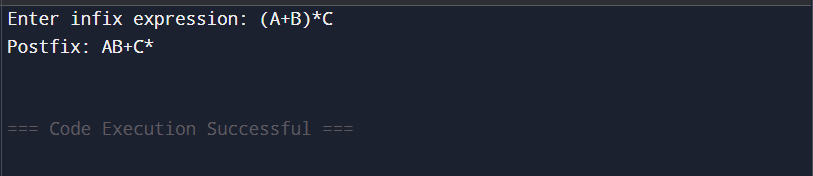
cin >> infix;

cout << "Postfix: " << infixToPostfix(infix) << endl;

return 0;

}

OUTPUT-



Ques-5)

#include <iostream>

#include <stack>

#include <string>

#include <cmath>

using namespace std;

int applyOp(int a, int b, char op) {

if (op == '+') return a + b;

if (op == '-') return a - b;

if (op == '\*') return a \* b;

if (op == '/') return a / b;

if (op == '^') return (int)pow(a, b);

return 0;

}

int evaluatePostfix(string expr) {

stack<int> s;

for (char c : expr) {

if (isdigit(c)) {

s.push(c - '0');

} else {

int b = s.top(); s.pop();

int a = s.top(); s.pop();

s.push(applyOp(a, b, c));

}

}

return s.top();

}

int main() {

string expr;

cout << "Enter postfix expression: ";

cin >> expr;

cout << "Result: " << evaluatePostfix(expr) << endl;

return 0;

}

