Q1) Develop a menu driven program demonstrating the following operations on a Stack using array: (i) push(), (ii) pop(), (iii) isEmpty(), (iv) isFull(), (v) display(), and (vi) peek().

#include <iostream>

using namespace std;

class Stack {

int\* arr;

int top, size;

public:

Stack(int n) {

size = n;

arr = new int[size];

top = -1;

}

void push(int x) {

if (top == size - 1) {

cout << "Stack is full!" << endl;

return;

}

arr[++top] = x;

cout << x << " added to stack." << endl;

}

void pop() {

if (top == -1) {

cout << "Stack is empty!" << endl;

return;

}

cout << arr[top--] << " removed from stack." << endl;

}

void peek() {

if (top == -1) cout << "Stack is empty!" << endl;

else cout << "Top element: " << arr[top] << endl;

}

void display() {

if (top == -1) {

cout << "Stack is empty!" << endl;

return;

}

cout << "Stack elements: ";

for (int i = top; i >= 0; i--) cout << arr[i] << " ";

cout << endl;

}

bool isEmpty() { return (top == -1); }

bool isFull() { return (top == size - 1); }

};

int main() {

int n;

cout << "Enter stack size: ";

cin >> n;

Stack s(n);

int choice, val;

do {

cout << "\n1.Push 2.Pop 3.Peek 4.Display 5.Check Empty 6.Check Full 0.Exit\n";

cout << "Enter choice: ";

cin >> choice;

switch (choice) {

case 1: cout << "Enter value: "; cin >> val; s.push(val); break;

case 2: s.pop(); break;

case 3: s.peek(); break;

case 4: s.display(); break;

case 5: cout << (s.isEmpty() ? "Stack is empty" : "Stack not empty") << endl; break;

case 6: cout << (s.isFull() ? "Stack is full" : "Stack not full") << endl; break;

case 0: cout << "Exiting..." << endl; break;

default: cout << "Invalid option!" << endl;

}

} while (choice != 0);

}

Q2) Given a string, reverse it using STACK. For example “DataStructure” should be output as “erutcurtSataD.”

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int main() {

string str;

cout << "Enter a string: ";

getline(cin, str);

stack<char> st;

for (char c : str) st.push(c);

cout << "Reversed string: ";

while (!st.empty()) {

cout << st.top();

st.pop();

}

cout << endl;

}

Q3) Write a program that checks if an expression has balanced parentheses.

#include <iostream>

#include <stack>

#include <string>

using namespace std;

bool checkBalanced(string exp) {

stack<char> st;

for (char c : exp) {

if (c == '(' || c == '{' || c == '[') st.push(c);

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

if (st.empty()) return false;

char top = st.top(); st.pop();

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

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

(c == ']' && top != '[')) return false;

}

}

return st.empty();

}

int main() {

string exp;

cout << "Enter an expression: ";

getline(cin, exp);

if (checkBalanced(exp)) cout << "Balanced" << endl;

else cout << "Not Balanced" << endl;

}

Q4) Write a program to convert an Infix expression into a Postfix expression.

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int precedence(char op) {

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

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

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

return 0;

}

string toPostfix(string infix) {

stack<char> st;

string post = "";

for (char c : infix) {

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

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

else if (c == ')') {

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

post += st.top(); st.pop();

}

st.pop();

} else {

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

post += st.top(); st.pop();

}

st.push(c);

}

}

while (!st.empty()) {

post += st.top(); st.pop();

}

return post;

}

int main() {

string infix;

cout << "Enter infix expression: ";

cin >> infix;

cout << "Postfix expression: " << toPostfix(infix) << endl;

}

Q5) Write a program for the evaluation of a Postfix expression.

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int evaluate(string exp) {

stack<int> st;

for (char c : exp) {

if (isdigit(c)) st.push(c - '0');

else {

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

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

if (c == '+') st.push(a + b);

else if (c == '-') st.push(a - b);

else if (c == '\*') st.push(a \* b);

else if (c == '/') st.push(a / b);

}

}

return st.top();

}

int main() {

string exp;

cout << "Enter postfix expression (single-digit numbers): ";

cin >> exp;

cout << "Result: " << evaluate(exp) << endl;

}