DSA

Lab assignment - 3

Name - ABHIRAJ SINGH

Roll no. - 1024030350

**Q1.**  
  
#include <iostream>

using namespace std;

#define MAX 100

class Stack {

int arr[MAX];

int top;

public:

Stack() { top = -1; }

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

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

void push(int val) {

if (isFull()) {

cout << "Stack Overflow!\n";

return;

}

arr[++top] = val;

}

void pop() {

if (isEmpty()) {

cout << "Stack Underflow!\n";

return;

}

cout << "Popped: " << arr[top--] << endl;

}

void peek() {

if (isEmpty()) {

cout << "Stack is Empty!\n";

return;

}

cout << "Top Element: " << arr[top] << endl;

}

void display() {

if (isEmpty()) {

cout << "Stack is Empty!\n";

return;

}

cout << "Stack Elements: ";

for (int i = top; i >= 0; i--)

cout << arr[i] << " ";

cout << endl;

}

};

int main() {

Stack s;

int choice, val;

while (true) {

cout << "\n\*\*\* STACK MENU \*\*\*\n";

cout << "1. Push\n2. Pop\n3. Peek\n4. Display\n5. IsEmpty\n6. IsFull\n7. 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\n" : "Stack is not Empty\n");

break;

case 6:

cout << (s.isFull() ? "Stack is Full\n" : "Stack is not Full\n");

break;

case 7:

exit(0);

default:

cout << "Invalid choice!\n";

}

}

}

**Q2.**

#include <iostream>

#include <stack>

using namespace std;

string reverseString(string str) {

stack<char> s;

for (char ch : str)

s.push(ch);

string rev = "";

while (!s.empty()) {

rev += s.top();

s.pop();

}

return rev;

}

int main() {

string str = "DataStructure";

cout << "Original String: " << str << endl;

cout << "Reversed String: " << reverseString(str) << endl;

return 0;

}

**Q3.**

#include <iostream>

#include <stack>

using namespace std;

bool isBalanced(string expr) {

stack<char> s;

for (char ch : expr) {

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

s.push(ch);

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

if (s.empty()) return false;

char top = s.top();

s.pop();

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

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

(ch == ']' && top != '['))

return false;

}

}

return s.empty();

}

int main() {

string expr;

cout << "Enter expression: ";

cin >> expr;

if (isBalanced(expr))

cout << "Balanced Expression\n";

else

cout << "Unbalanced Expression\n";

return 0;

}

**Q4.**

#include <iostream>

#include <stack>

using namespace std;

int precedence(char op) {

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

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

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

return -1;

}

string infixToPostfix(string infix) {

stack<char> s;

string postfix = "";

for (char ch : infix) {

if (isalnum(ch)) {

postfix += ch;

} else if (ch == '(') {

s.push(ch);

} else if (ch == ')') {

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

postfix += s.top();

s.pop();

}

s.pop();

} else {

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

postfix += s.top();

s.pop();

}

s.push(ch);

}

}

while (!s.empty()) {

postfix += s.top();

s.pop();

}

return postfix;

}

int main() {

string infix;

cout << "Enter Infix Expression: ";

cin >> infix;

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

return 0;

}

**Q5.**

#include <iostream>

#include <stack>

using namespace std;

int evaluatePostfix(string postfix) {

stack<int> s;

for (char ch : postfix) {

if (isdigit(ch)) {

s.push(ch - '0'); // convert char to int

} else {

int val2 = s.top(); s.pop();

int val1 = s.top(); s.pop();

switch (ch) {

case '+': s.push(val1 + val2); break;

case '-': s.push(val1 - val2); break;

case '\*': s.push(val1 \* val2); break;

case '/': s.push(val1 / val2); break;

}

}

}

return s.top();

}

int main() {

string postfix;

cout << "Enter Postfix Expression (single-digit operands): ";

cin >> postfix;

cout << "Evaluated Result: " << evaluatePostfix(postfix) << endl;

return 0;

}