#include <iostream>

#include <stack>

#include <string>

#include <cctype>

#include <cmath>

using namespace std;

// 1. Balancing Parentheses

bool isBalanced(const 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();

}

// 2. Infix to Postfix

int precedence(char op) {

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

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

return 0;

}

string infixToPostfix(const 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(); // remove '('

} 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;

}

// 3. Postfix Evaluation

int evaluatePostfix(const string& postfix) {

stack<int> s;

for (char ch : postfix) {

if (isdigit(ch)) {

s.push(ch - '0');

} else {

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

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

switch (ch) {

case '+': s.push(a + b); break;

case '-': s.push(a - b); break;

case '\*': s.push(a \* b); break;

case '/': s.push(a / b); break;

case '^': s.push(pow(a, b)); break;

}

}

}

return s.top();

}

// 4. Recursive Function using Stack (Factorial)

int factorialUsingStack(int n) {

stack<int> s;

while (n > 1) {

s.push(n);

n--;

}

int result = 1;

while (!s.empty()) {

result \*= s.top();

s.pop();

}

return result;

}

// 5. Reverse String using Stack

string reverseString(const string& str) {

stack<char> s;

for (char ch : str) s.push(ch);

string reversed;

while (!s.empty()) {

reversed += s.top(); s.pop();

}

return reversed;

}

int main() {

int choice;

cout << "STACK APPLICATIONS\n";

cout << "1. Balancing Parentheses\n";

cout << "2. Infix to Postfix\n";

cout << "3. Postfix Evaluation\n";

cout << "4. Factorial using Stack\n";

cout << "5. Reverse a String\n";

cout << "Enter your choice (1-5): ";

cin >> choice;

cin.ignore(); // clear newline from buffer

if (choice == 1) {

string expr;

cout << "Enter an expression: ";

getline(cin, expr);

cout << (isBalanced(expr) ? "Balanced" : "Not Balanced") << endl;

} else if (choice == 2) {

string infix;

cout << "Enter infix expression (e.g., a+b\*c): ";

getline(cin, infix);

string postfix = infixToPostfix(infix);

cout << "Postfix: " << postfix << endl;

} else if (choice == 3) {

string postfix;

cout << "Enter postfix expression (e.g., 53+82-\*): ";

getline(cin, postfix);

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

} else if (choice == 4) {

int n;

cout << "Enter a number: ";

cin >> n;

cout << "Factorial: " << factorialUsingStack(n) << endl;

} else if (choice == 5) {

string str;

cout << "Enter a string: ";

getline(cin, str);

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

} else {

cout << "Invalid choice!" << endl;

}

return 0;

}