***Exercises: LAB 02***

***Data Structure:***

**Student Name: \_\_\_Muhammad Faizan \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Roll No: \_\_\_\_\_\_\_074\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Instructor: \_\_\_\_\_Sir Sharjeel\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date: \_\_\_\_\_\_\_30-09-2025\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Task:**

1. **Write a program to take numbers from user and stored into stack. Pop the values from**

**stack and show only even values on screen and show highest marks on the screen.**

**Code:**

#include <iostream>

#include <stack>

using namespace std;

int main() {

stack<int> st;

int n, num, highest = -1;

cout << "Enter how many numbers you want to store: ";

cin >> n;

// Taking input and pushing into stack

cout << "Enter " << n << " numbers:\n";

for (int i = 0; i < n; i++) {

cin >> num;

st.push(num);

if (num > highest) {

highest = num; // keep track of highest number

}

}

cout << "\nEven numbers (popped from stack): ";

while (!st.empty()) {

int val = st.top();

st.pop();

if (val % 2 == 0) {

cout << val << " ";

}

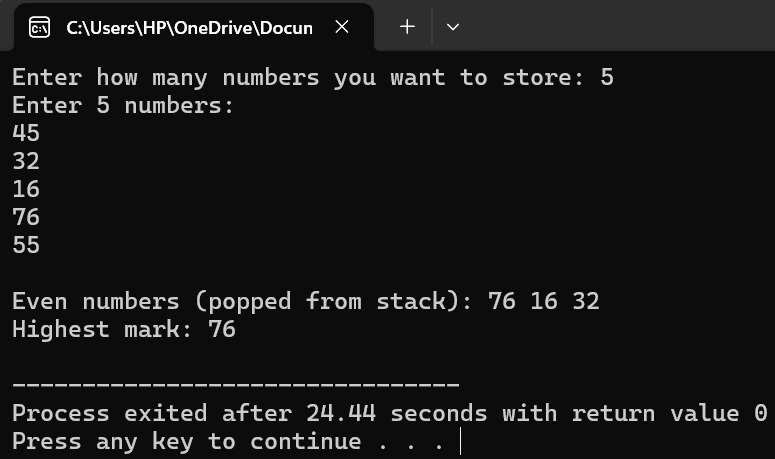
}

cout << "\nHighest mark: " << highest << endl;

return 0;

}

**Output:**



**Task:**

**2. Write a program to input an infix expression into a string variable. Convert the expression**

**into post-fix notation by using a stack**.

**Code:**

#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> st;

string postfix = "";

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

char ch = infix[i];

if (isalnum(ch)) {

postfix += ch;

}

else if (ch == '(') {

st.push(ch);

}

else if (ch == ')') {

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

postfix += st.top();

st.pop();

}

st.pop();

}

else if (isOperator(ch)) {

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

postfix += st.top();

st.pop();

}

st.push(ch);

}

}

while (!st.empty()) {

postfix += st.top();

st.pop();

}

return postfix;

}

int main() {

string infix;

cout << "Enter an infix expression (example: A+B\*C): ";

cin >> infix;

string postfix = infixToPostfix(infix);

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

return 0;

}

**Output:**

