**Department of Engineering Technology**



**Foundation University Islamabad** **School of Science and Technology**

**DSA Lab**

**Name: Muhammad Usman**

**Roll no: 079**

**Lab#: 02**

**Topic: \_\_\_\_\_\_\_ OBJECTIVES:**

i. Objective - 1 ii. Objective - 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Performance** | |  | **Lab Report** | |  |
| **Description** | **Total**  **Marks** | **Marks Obtained** | **Description** | **Total**  **Marks** | **Marks** **Obtained** |
| Implementation of Code | 5 |  | Lab Exercises | 5 |  |
| **Total Marks obtained** | |  |  | |  |

**Q1:**

#include <iostream>

using namespace std;

const int MAX = 100;

int main() {

int stack[MAX];

int top = -1;

int n, number, max = -1;

cout << "Enter how many numbers (marks) you want to input: ";

cin >> n;

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

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

cout << "Enter number " << i + 1 << ": ";

cin >> number;

if (top < MAX - 1) {

top++;

stack[top] = number;

} else {

cout << "Stack Overflow! Cannot push more items.\n";

break;

}

if (number > max) {

max = number;

}

}

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

while (top >= 0) {

int val = stack[top];

top--;

if (val % 2 == 0) {

cout << val << " ";

}

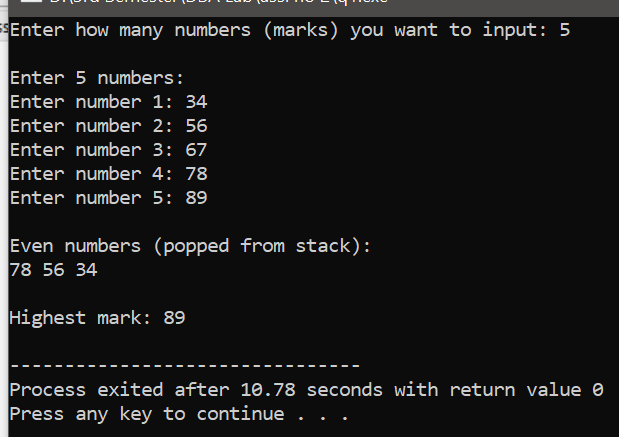
}

cout << "\n\nHighest mark: " << max << endl;

return 0;

}

**Output:**

****

**Q2:**

#include <iostream>

#include <stack>

#include <string>

using namespace std;

int precedence(char op) {

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

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

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

else return 0;

}

bool isOperator(char c) {

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

}

string infixToPostfix(string infix) {

stack<char> s;

string postfix = "";

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

char c = infix[i];

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 an infix expression: ";

getline(cin, infix);

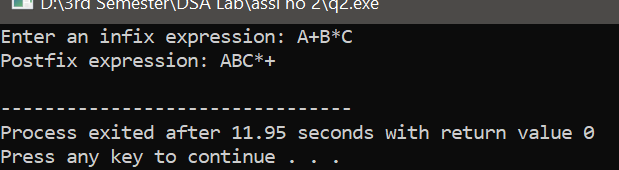
string postfix = infixToPostfix(infix);

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

return 0;

}

**Output:**

****