PRACTICAL NO. :- 6

AIM :- Implement a Stack and perform the stack operations: Infix to Postfix, Infix to Prefix, Evaluation
1. Infix to Postfix, 2. Infix to Prefix,
of Postfix Expression, Print using Menu Driver Program such as and 3. Evaluation of Postfix Expression and 4. Exit.

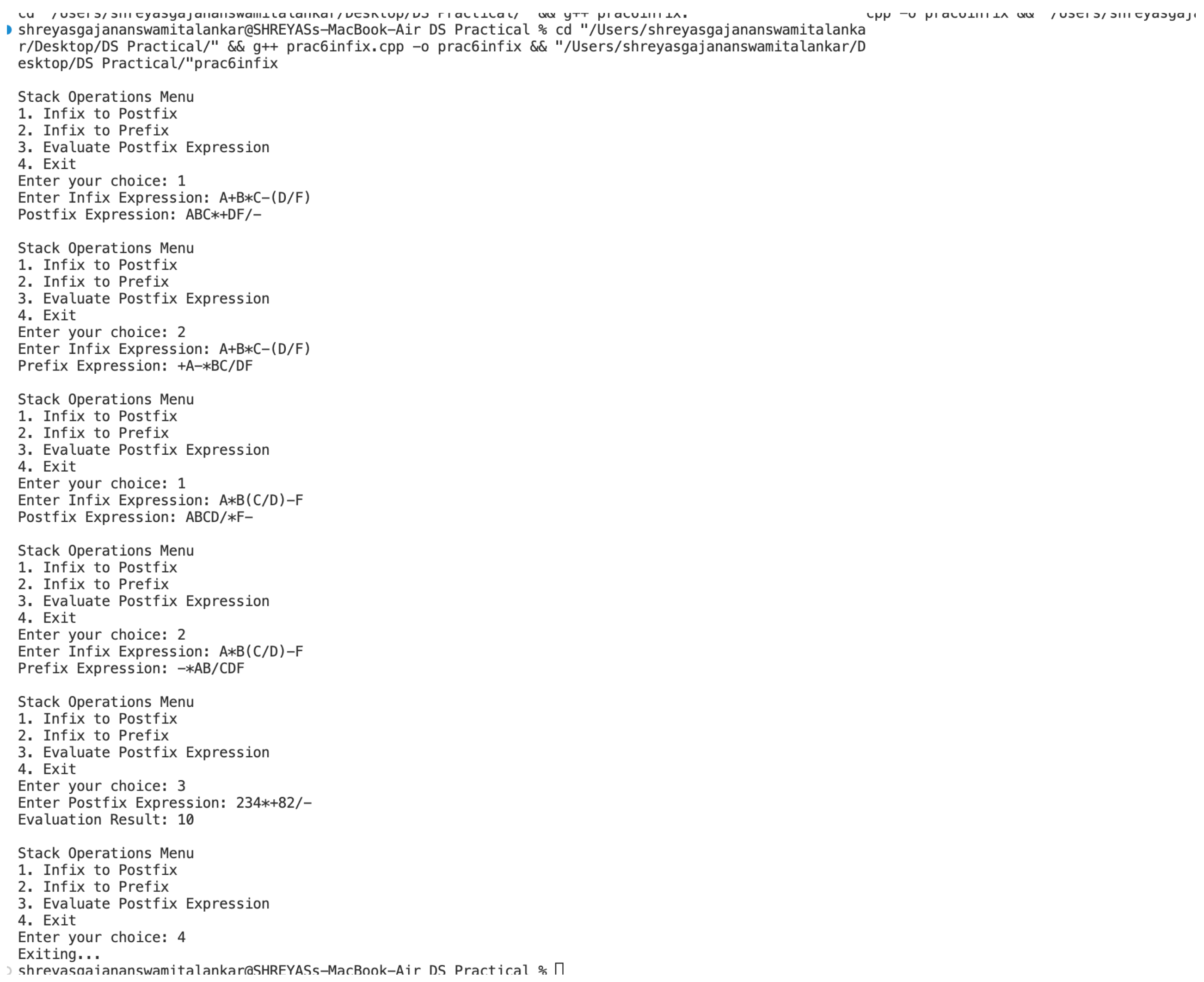
PROGRAM :-

```
#include <iostream>
#include <cstring>
using namespace std;

#define MAX 100

// Stack implementation using an array
struct Stack {
    int top;
    char arr[MAX];

    Stack() {
        top = -1;
    }

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

    void push(char item) {
        if (top >= MAX - 1) {
            cout << "Stack Overflow\n";
        } else {
            arr[++top] = item;
        }
    }

    char pop() {
        if (top < 0) {
            cout << "Stack Underflow\n";
            return '\0';
        } else {
            return arr[top--];
        }
    }

    char peek() {
        if (top < 0) {
            return '\0';
        } else {
            return arr[top];
        }
    }
```

```
};

// Function to return precedence of operators
int precedence(char op) {
    if (op == '+' || op == '-') {
        return 1;
    }
    if (op == '*' || op == '/') {
        return 2;
    }
    if (op == '^') {
        return 3;
    }
    return 0;
}

// Function to check if the character is operand
bool isOperand(char c) {
    return (c >= '0' && c <= '9') || (c >= 'a' && c <= 'z') || (c
>= 'A' && c <= 'Z');
}

// Custom reverse function (without <algorithm>)
void customReverse(string &exp) {
    int n = exp.length();
    for (int i = 0; i < n / 2; i++) {
        swap(exp[i], exp[n - i - 1]);
    }
}

// Function to convert Infix to Postfix
string infixToPostfix(string exp) {
    Stack stack;
    string result;
    for (int i = 0; i < exp.length(); i++) {
        char c = exp[i];

        if (isOperand(c)) {
            result += c;
        } else if (c == '(') {
            stack.push(c);
        } else if (c == ')') {
            while (!stack.isEmpty() && stack.peek() != '(') {
                result += stack.pop();
            }
            stack.pop();
        } else {
            while (!stack.isEmpty() && precedence(c) <=
precedence(stack.peek())) {
                result += stack.pop();
            }
            stack.push(c);
        }
    }

    while (!stack.isEmpty()) {
        result += stack.pop();
    }

    return result;
}

// Function to convert Infix to Prefix
string infixToPrefix(string exp) {
    // Reverse the infix expression
    customReverse(exp);

    // Replace '(' with ')' and vice versa
    for (int i = 0; i < exp.length(); i++) {
        if (exp[i] == '(') exp[i] = ')';
        else if (exp[i] == ')') exp[i] = '(';
    }

    // Convert to postfix
    string postfix = infixToPostfix(exp);

    // Reverse postfix to get prefix
    customReverse(postfix);

    return postfix;
}

// Function to evaluate Postfix Expression
int evaluatePostfix(string exp) {
    Stack stack;

    for (int i = 0; i < exp.length(); i++) {
        char c = exp[i];

        if (isdigit(c)) {
            stack.push(c - '0');
        } else {
            int val1 = stack.pop();
            int val2 = stack.pop();

            switch (c) {
            case '+': stack.push(val2 + val1); break;
            case '-': stack.push(val2 - val1); break;
            case '*': stack.push(val2 * val1); break;
            case '/': stack.push(val2 / val1); break;
            }
        }
    }
    return stack.pop();
}

// Menu-driven program
void menu() {
    int choice;
    string expression;

    do {
        cout << "\nStack Operations Menu\n";
        cout << "1. Infix to Postfix\n";
        cout << "2. Infix to Prefix\n";
        cout << "3. Evaluate Postfix Expression\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "Enter Infix Expression: ";
            cin >> expression;
            cout << "Postfix Expression: " <<
infixToPostfix(expression) << endl;
            break;
        case 2:
            cout << "Enter Infix Expression: ";
            cin >> expression;
            cout << "Prefix Expression: " <<
infixToPrefix(expression) << endl;
            break;
        case 3:
            cout << "Enter Postfix Expression: ";
            cin >> expression;
            cout << "Evaluation Result: " <<
evaluatePostfix(expression) << endl;
            break;
        case 4:
            cout << "Exiting...\n";
            break;
        default:
            cout << "Invalid choice! Please try again.\n";
        }
    } while (choice != 4);
}

int main() {
    menu();
    return 0;
}
```

PRACTICAL NO. :- 6

OUTPUT :-



```
cd "/Users/shreyasgajananswamitalankar/Desktop/DS Practical/" && g++ prac6infix.                    cpp -o prac6infix && "/Users/shreyasgaj
shreyasgajananswamitalankar@SHREYASs-MacBook-Air DS Practical % cd "/Users/shreyasgajananswamitalanka
r/Desktop/DS Practical/" && g++ prac6infix.cpp -o prac6infix && "/Users/shreyasgajananswamitalankar/D
esktop/DS Practical/"prac6infix

Stack Operations Menu
1. Infix to Postfix
2. Infix to Prefix
3. Evaluate Postfix Expression
4. Exit
Enter your choice: 1
Enter Infix Expression: A+B*C-(D/F)
Postfix Expression: ABC*+DF/-

Stack Operations Menu
1. Infix to Postfix
2. Infix to Prefix
3. Evaluate Postfix Expression
4. Exit
Enter your choice: 2
Enter Infix Expression: A+B*C-(D/F)
Prefix Expression: +A-*BC/DF

Stack Operations Menu
1. Infix to Postfix
2. Infix to Prefix
3. Evaluate Postfix Expression
4. Exit
Enter your choice: 1
Enter Infix Expression: A*B(C/D)-F
Postfix Expression: ABCD/*F-

Stack Operations Menu
1. Infix to Postfix
2. Infix to Prefix
3. Evaluate Postfix Expression
4. Exit
Enter your choice: 2
Enter Infix Expression: A*B(C/D)-F
Prefix Expression: -*AB/CDF

Stack Operations Menu
1. Infix to Postfix
2. Infix to Prefix
3. Evaluate Postfix Expression
4. Exit
Enter your choice: 3
Enter Postfix Expression: 234*+82/-
Evaluation Result: 10

Stack Operations Menu
1. Infix to Postfix
2. Infix to Prefix
3. Evaluate Postfix Expression
4. Exit
Enter your choice: 4
Exiting...
shreyasgajananswamitalankar@SHREYASs-MacBook-Air DS Practical %
```

GITHUB LINK : - https://github.com/shreyastalankar/DATA-STRUCTURE-PRACTICALS.git