ASSIGNMENT-5 - B

```
/*Name: Sahil Badve PRN: B24CE1114 Div: S.Y.B-Tech 2 Batch C*/
#include <iostream>
#include <string>
using namespace std;

class Stack {
private:
    char arr[50];
    int top;
public:
    Stack() { top = -1; }

    void push(char x) {
        if (top >= 49) {
            cout << "Stack overflow!" << endl;
        } else {
            arr[++top] = x;
        }
    }

    char pop() {
        if (isEmpty()) {
            return '\0';
        } else {
            return arr[top--];
        }
    }

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

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

// Function to get precedence of operators
```

```
int precedence(char ch) {
    if (ch == '^')
        return 3;
    else if (ch == '*' || ch == '/')
        return 2;
    else if (ch == '+' || ch == '-')
        return 1;
    else
        return 0;
}

// Function to check if a character is an operator
bool isOperator(char ch) {
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^');
}

// Function to convert infix expression to postfix
string infixToPostfix(string infix) {
    Stack s;
    string postfix = "";

    for (int i = 0; infix[i] != '\0'; i++) {
        char ch = infix[i];

        // If operand, add to postfix
        if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z') || (ch >= '0' && ch <= '9')) {
            postfix += ch;
        }
        // If '(', push to stack
        else if (ch == '(') {
            s.push(ch);
        }
        // If ')', pop until '('
        else if (ch == ')') {
            while (!s.isEmpty() && s.peek() != '(') {
                postfix += s.pop();
            }
            s.pop(); // remove '('
        }
        // If operator
        else if (isOperator(ch)) {
            while (!s.isEmpty() && precedence(s.peek()) >= precedence(ch)) {
                postfix += s.pop();
            }
```

```
            s.push(ch);
        }
    }

    // Pop remaining operators
    while (!s.isEmpty()) {
        postfix += s.pop();
    }

    return postfix;
}

int main() {
    string infix;
    cout << "Enter an infix expression: ";
    cin >> infix;

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

    return 0;
}
```

OUTPUT:-

Enter an infix expression: K+L-M*N+(0^P)*W/U/V*T

Postfix Expression: KL+MN*-0P^W*U/V/T*+