- **C program to convert infix to postfix expression using stack method**

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Stack {
    int top;
    unsigned capacity;
    char *array;
};

struct Stack *createStack(unsigned capacity) {
    struct Stack *stack = (struct Stack *)malloc(sizeof(struct Stack));
    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (char *)malloc(capacity * sizeof(char));
    return stack;
}

int isOperator(char ch) {
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/');
}

int precedence(char op) {
    if (op == '+' || op == '-')
        return 1;
```

```
    if (op == '*' || op == '/')
        return 2;
    return 0;
}

void push(struct Stack *stack, char item) {
    if (stack->top == stack->capacity - 1) {
        printf("Stack Overflow\n");
        return;
    }
    stack->array[++stack->top] = item;
}

char pop(struct Stack *stack) {
    if (stack->top == -1) {
        printf("Stack Underflow\n");
        return '\0';
    }
    return stack->array[stack->top--];
}

void infixToPostfix(char *infix) {
    int length = strlen(infix);
    struct Stack *stack = createStack(length);
    int outputIndex = 0;
    char postfix[length];

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

```
    char c = infix[i];
    if (isalnum(c)) {
        postfix[outputIndex++] = c;
    } else if (c == '(') {
        push(stack, c);
    } else if (c == ')') {
        while (stack->top != -1 && stack->array[stack->top] != '(') {
            postfix[outputIndex++] = pop(stack);
        }
        if (stack->top != -1 && stack->array[stack->top] != '(') {
            printf("Invalid expression\n");
            return;
        } else {
            pop(stack);
        }
    } else {
        while (stack->top != -1 && precedence(c) <= precedence(stack->array[stack->top])) {
            postfix[outputIndex++] = pop(stack);
        }
        push(stack, c);
    }
}

while (stack->top != -1) {
    postfix[outputIndex++] = pop(stack);
}

postfix[outputIndex] = '\0';
```

```
    printf("Postfix expression: %s\n", postfix);
}

int main() {
    char infix[100];
    printf("Enter infix expression: ");
    scanf("%s", infix);
    infixToPostfix(infix);
    return 0;
}
```

Output:

```
Enter infix expression: a*b-c/d
Postfix expression: ab*cd/-
```