Program - 07	
Title :	**Write a program that uses stack operations to convert a given infix expression into its postfix expression.**

Code:

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

// Node structure
struct Node {
    int data;
    struct Node* next;
};

// Stack structure
struct Stack {
    struct Node* top;
};

// Initialize a new stack
void initStack(struct Stack* stack) {
    stack->top = NULL;
}

// Check if stack is empty
int isEmpty(struct Stack* stack) {
    return stack->top == NULL;
}

// Push an integer onto the stack
void push(struct Stack* stack, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = stack->top;
    stack->top = newNode;
}

// Pop an integer from the stack
int pop(struct Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty!\n");
        return 0;
    }
    struct Node* temp = stack->top;
    int data = temp->data;
    stack->top = temp->next;
    free(temp);
    return data;
}

// Return the top integer from the stack without popping it
int peek(struct Stack* stack) {
    if (isEmpty(stack)) {
        return 0;
    }
```

```
    return stack->top->data;
}

// Evaluate postfix expression
int evaluatePostfix(char* postfix) {
    int i, operand1, operand2, result;
    struct Stack stack;
    initStack(&stack);
    char c;

    for (i = 0; postfix[i] != '\0'; i++) {
        c = postfix[i];
        if (isdigit(c)) {
            push(&stack, c - '0');
        } else {
            operand2 = pop(&stack);
            operand1 = pop(&stack);
            switch (c) {
                case '+':
                    result = operand1 + operand2;
                    break;
                case '-':
                    result = operand1 - operand2;
                    break;
                case '*':
                    result = operand1 * operand2;
                    break;
                case '/':
                    result = operand1 / operand2;
                    break;
                default:
                    printf("Invalid operator!\n");
                    return 0;
            }
            push(&stack, result);
        }
    }

    return pop(&stack);
}

// Main function
int main() {
    char postfix[100];

    printf("Enter postfix expression: ");
    scanf("%s", postfix);

    int result = evaluatePostfix(postfix);

    printf("Result: %d\n", result);

    return 0;
}
```

Output:

```
C:\Users\hp\Desktop\c program>postfix
Enter infix expression: m*n+(p-q)-+*

==========================

Postfix expression: mn*pq-+-*+
```

Date : ___/____/_____	Teacher Sign ……