Rajalakshmi Engineering College

Name: Kirubashini R
Email: 241501087@rajalakshmi.edu.in
Roll no: 241501087
Phone: 9843749339
Branch: REC
Department: I AIML AD
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are a software developer tasked with building a module for a scientific calculator application. The primary function of this module is to convert infix mathematical expressions, which are easier for users to read and write, into postfix notation (also known as Reverse Polish Notation). Postfix notation is more straightforward for the application to evaluate because it removes the need for parentheses and operator precedence rules.

The scientific calculator needs to handle various mathematical expressions with different operators and ensure the conversion is correct. Your task is to implement this infix-to-postfix conversion algorithm using a stack-based approach.

Example

Input:

a+b

Output:

ab+

Explanation:

The postfix representation of (a+b) is ab+.

Input Format

The input is a string, representing the infix expression.

Output Format

The output displays the postfix representation of the given infix expression.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a+(b*e)
Output: abe*+

Answer

```
#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));

   if (!stack)
```

```
        return NULL;

    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (char*)malloc(stack->capacity * sizeof(char));

    return stack;
}

int isEmpty(struct Stack* stack) {
    return stack->top == -1;
}

char peek(struct Stack* stack) {
    return stack->array[stack->top];
}

char pop(struct Stack* stack) {
    if (!isEmpty(stack))
        return stack->array[stack->top--];
    return '$';
}

void push(struct Stack* stack, char op) {
    stack->array[++stack->top] = op;
}
// You are using GCC
int isOperand(char ch) {
    //type your code here
    return (ch >= 'a' && ch <= 'z') ||
           (ch >= 'A' && ch <= 'Z') ||
           (ch >= '0' && ch <= '9');
}

int Prec(char ch) {
   //type your code here
    if (ch == '^') return 3;
    if (ch == '*' || ch == '/') return 2;
    if (ch == '+' || ch == '-') return 1;
    return 0;
```

```
}

void infixToPostfix(char* exp) {

    //type your code here
    int i, k = 0;
    struct Stack* stack = createStack(strlen(exp));
    char* result = (char*)malloc(strlen(exp) + 1);

    for (i = 0; exp[i]; i++) {
        char ch = exp[i];
        if (isOperand(ch)) {
            result[k++] = ch;
        } else if (ch == '(') {
            push(stack, ch);
        } else if (ch == ')') {
            while (!isEmpty(stack) && peek(stack) != '(')
                result[k++] = pop(stack);
            if (!isEmpty(stack) && peek(stack) == '(')
                pop(stack);
        } else {
            while (!isEmpty(stack) && peek(stack) != '(' &&
                  (Prec(peek(stack)) > Prec(ch) ||
                   (Prec(peek(stack)) == Prec(ch) && ch != '^')))
                result[k++] = pop(stack);
            push(stack, ch);
        }
    }

    while (!isEmpty(stack))
        result[k++] = pop(stack);

    result[k] = '\0';
    printf("%s\n", result);
    free(result);

}

int main() {
    char exp[100];
    scanf("%s", exp);

    infixToPostfix(exp);
```

```
    return 0;
}
```

Status : Correct ***Marks : 10/10***