Rajalakshmi Engineering College

Name: Divya Shree R
Email: 240801071@rajalakshmi.edu.in
Roll no: 240801071
Phone: 9363738849
Branch: REC
Department: I ECE FA
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 3_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Suppose you are building a calculator application that allows users to enter mathematical expressions in infix notation. One of the key features of your calculator is the ability to convert the entered expression to postfix notation using a Stack data structure.

Write a function to convert infix notation to postfix notation using a Stack.

Input Format

The input consists of a string, an infix expression that includes only digits(0-9), and operators(+, -, *, /).

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1+2*3/4-5

Output: 123*4/+5-

Answer

```
// You are using GCC
 #include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

typedef struct {
    char* items;
    int top;
} Stack;

void push(Stack* s, char c) {
    s->items[++s->top] = c;
}

char pop(Stack* s) {
    return s->items[s->top--];
}

int prec(char c) {
    if (c == '+' || c == '-') return 1;
    if (c == '*' || c == '/') return 2;
    return 0;
}

void infixToPostfix(char* s) {
    if (s == NULL || strlen(s) == 0) return;
    Stack stack;
    stack.items = (char*)malloc((strlen(s) + 1) * sizeof(char));
    stack.top = -1;
```

```
    for (int i = 0; i < strlen(s); i++) {
        char c = s[i];
        if (isdigit(c)) printf("%c", c);
        else if (c == '(') push(&stack, c);
        else if (c == ')') {
            while (stack.top >= 0 && stack.items[stack.top] != '(') printf("%c",
pop(&stack));
            if (stack.top >= 0) pop(&stack);
        }
        else if (c == '+' || c == '-' || c == '*' || c == '/') {
            while (stack.top >= 0 && stack.items[stack.top] != '(' && prec(c) <=
prec(stack.items[stack.top])) {
                printf("%c", pop(&stack));
            }
            push(&stack, c);
        }
    }
    while (stack.top >= 0) printf("%c", pop(&stack));
    free(stack.items);
}

int main() {
    char s[100];
    fgets(s, 100, stdin);
    s[strcspn(s, "\n")] = 0;
    infixToPostfix(s);
    return 0;
}
```

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

2. Problem Statement

Latha is taking a computer science course and has recently learned about infix and postfix expressions. She is fascinated by the idea of converting infix expressions into postfix notation. To practice this concept, she wants to implement a program that can perform the conversion for her.

Help Latha by designing a program that takes an infix expression as input and outputs its equivalent postfix notation.

Example

Input:

(3+4)5

Output:

34+5

Input Format

The input consists of a string, the infix expression to be converted to postfix notation.

Output Format

The output displays a string, the postfix expression equivalent of the input infix expression.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: A+B*C-D/E

Output: ABC*+DE/-

Answer

```
// You are using GCC
 #include <stdio.h>
#include <ctype.h>
#include <string.h>

#define MAX 100

char stack[MAX];
int top = -1;

// Push onto stack
void push(char c) {
    stack[++top] = c;
```

```
}

// Pop from stack
char pop() {
    if (top == -1) return -1;
    return stack[top--];
}

// Peek top of stack
char peek() {
    if (top == -1) return -1;
    return stack[top];
}

// Check precedence
int precedence(char op) {
    if (op == '+' || op == '-') return 1;
    if (op == '*' || op == '/') return 2;
    return 0;
}

// Convert infix to postfix
void infixToPostfix(char* infix, char* postfix) {
    int i = 0, j = 0;
    char ch;

    while ((ch = infix[i++]) != '\0') {
        if (isdigit(ch) || isalpha(ch)) {
            postfix[j++] = ch;
        } else if (ch == '(') {
            push(ch);
        } else if (ch == ')') {
            while (peek() != '(')
                postfix[j++] = pop();
            pop();  // pop '('
        } else {
            while (precedence(peek()) >= precedence(ch))
                postfix[j++] = pop();
            push(ch);
        }
    }
```

```
    while (top != -1)
        postfix[j++] = pop();

    postfix[j] = '\0';
}

int main() {
    char infix[MAX], postfix[MAX];

    fgets(infix, MAX, stdin);
    infix[strcspn(infix, "\n")] = '\0';  // remove newline

    infixToPostfix(infix, postfix);

    printf("%s\n", postfix);

    return 0;
}
```

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

3. Problem Statement

In an educational setting, Professor Smith tasks Computer Science students with designing an algorithm to evaluate postfix expressions efficiently, fostering problem-solving skills and understanding of stack-based computations.

The program prompts users to input a postfix expression, evaluates it, and displays the result, aiding students in honing their coding abilities.

Input Format

The input consists of the postfix mathematical expression.

The expression will contain real numbers and mathematical operators (+, -, *, /), without any space.

Output Format

The output prints the result of evaluating the given postfix expression.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 82/

Output: 4

Answer

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

#define MAX 100

// Stack Implementation
int stack[MAX];
int top = -1;

void push(int val) {
    stack[++top] = val;
}

int pop() {
    if (top == -1) {
        printf("Error: Stack underflow\n");
        exit(1);
    }
    return stack[top--];
}

int evaluatePostfix(char* expr) {
    for (int i = 0; expr[i]; i++) {
        if (isdigit(expr[i])) {
            // Convert char digit to int and push
            push(expr[i] - '0');
        } else if (expr[i] == ' ') {
            continue;
        } else {
            // It's an operator
```

```
            int val2 = pop();
            int val1 = pop();
            switch (expr[i]) {
                case '+': push(val1 + val2); break;
                case '-': push(val1 - val2); break;
                case '*': push(val1 * val2); break;
                case '/': push(val1 / val2); break;
                default:
                    printf("Unsupported operator: %c\n", expr[i]);
                    exit(1);
            }
        }
    }
    return pop();
}

int main() {
    char postfix[100];
    fgets(postfix, sizeof(postfix), stdin);

    int result = evaluatePostfix(postfix);
    printf("%d\n", result);

    return 0;
}
```

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