Rajalakshmi Engineering College

Name: CHEMBETI JAHNAVI
Email: 240701089@rajalakshmi.edu.in
Roll no: 240701089
Phone: 6300874727
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

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

Siri is a computer science student who loves solving mathematical problems. She recently learned about infix and postfix expressions and was fascinated by how they can be used to evaluate mathematical expressions.

She decided to write a program to convert an infix expression with operators to its postfix form. Help Siri in writing the program.

Input Format

The input consists of a single line containing an infix expression.

Output Format

The output prints a single line containing the postfix expression equivalent to the

given infix expression.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: (2 + 3) * 4

Output: 23+4*

Answer

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

#define MAX 50

struct Stack {
 char items[MAX];
 int top;
};

void initStack(struct Stack *s) {
 s->top = -1;
}

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

int isFull(struct Stack *s) {
 return s->top == MAX - 1;
}

void push(struct Stack *s, char item) {
 if (!isFull(s)) {
  s->items[++(s->top)] = item;
 }
}
```

```
char pop(struct Stack *s) {
 if (!isEmpty(s)) {
  return s->items[(s->top)--];
 }
 return '\0';
}

char peek(struct Stack *s) {
 if (!isEmpty(s)) {
  return s->items[s->top];
 }
 return '\0';
}

int precedence(char op) {
 switch (op) {
  case '+':
  case '-':
   return 1;
  case '*':
  case '/':
   return 2;
  default:
   return 0;
 }
}

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

void infixToPostfix(char *infix, char *postfix) {
 struct Stack stack;
 initStack(&stack);
 int i = 0, j = 0;

 while (infix[i] != '\0') {
  if (isspace(infix[i])) {
   i++;
   continue;
  }
```

```
 if (isdigit(infix[i])) {
  postfix[j++] = infix[i];
 } else if (infix[i] == '(') {
  push(&stack, '(');
 } else if (infix[i] == ')') {
  while (!isEmpty(&stack) && peek(&stack) != '(') {
   postfix[j++] = pop(&stack);
  }
  pop(&stack);
 } else if (isOperator(infix[i])) {
  while (!isEmpty(&stack) && precedence(peek(&stack)) >= precedence(infix[i])) {
   postfix[j++] = pop(&stack);
  }
  push(&stack, infix[i]);
 }
 i++;
}

 while (!isEmpty(&stack)) {
  postfix[j++] = pop(&stack);
 }
 postfix[j] = '\0';
}

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

 fgets(infix, sizeof(infix), stdin);
 infixToPostfix(infix, postfix);
 printf("%s\n", postfix);
 return 0;
}
```

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

2. 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>

#define MAX_SIZE 100


typedef struct {
    double items[MAX_SIZE];
    int top;
} Stack;


void initStack(Stack *s) {
    s->top = -1;
}
```

```
int isFull(Stack *s) {
    return s->top == MAX_SIZE - 1;
}


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


void push(Stack *s, double item) {
    if (!isFull(s)) {
        s->items[++(s->top)] = item;
    } else {
        printf("Stack is full!\n");
    }
}


double pop(Stack *s) {
    if (!isEmpty(s)) {
        return s->items[(s->top)--];
    } else {
        printf("Stack is empty!\n");
        return 0.0;
    }
}


double evaluatePostfix(char* expression) {
    Stack s;
    initStack(&s);

    for (int i = 0; expression[i] != '\0'; i++) {
        if (isdigit(expression[i])) {

            push(&s, (double)(expression[i] - '0'));
        } else {

            double operand2 = pop(&s);
            double operand1 = pop(&s);
```

```
        double result;


        switch (expression[i]) {
            case '+':
                result = operand1 + operand2;
                break;
            case '-':
                result = operand1 - operand2;
                break;
            case '*':
                result = operand1 * operand2;
                break;
            case '/':
                result = operand1 / operand2;
                break;
            default:
                printf("Invalid operator: %c\n", expression[i]);
                exit(EXIT_FAILURE);
        }

        push(&s, result);
    }
  }


  return pop(&s);
}


int main() {
  char postfix[MAX_SIZE];



  scanf("%s", postfix);


  double result = evaluatePostfix(postfix);
  printf("\n%.0f\n", result);

  return 0;
```

}

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

3. 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 <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAX 50
struct Stack {
    int top;
    char items[MAX];
};
void initStack(struct Stack *s) {
    s->top = -1;
}


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

int isFull(struct Stack *s) {
    return s->top == MAX - 1;
}

void push(struct Stack *s, char item) {
    if (!isFull(s)) {
        s->items[++(s->top)] = item;
    }
}

char pop(struct Stack *s) {
    if (!isEmpty(s)) {
        return s->items[(s->top)--];
    }
    return '\0';
}


int precedence(char op) {
    switch(op) {
        case '+':
```

```
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '(':
            return 0;
        default:
            return -1;
    }
}


void infixToPostfix(char *infix, char *postfix) {
    struct Stack stack;
    initStack(&stack);
    int k = 0;

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


        if (isalnum(ch)) {
            postfix[k++] = ch;
        }

        else if (ch == '(') {
            push(&stack, ch);
        }

        else if (ch == ')') {
            while (!isEmpty(&stack) && stack.items[stack.top] != '(') {
                postfix[k++] = pop(&stack);
            }
            pop(&stack);
        }

        else {
            while (!isEmpty(&stack) && precedence(stack.items[stack.top]) >=
precedence(ch)) {
                postfix[k++] = pop(&stack);
            }
```

```
            push(&stack, ch);
        }
    }

    while (!isEmpty(&stack)) {
        postfix[k++] = pop(&stack);
    }

    postfix[k] = '\0';
}

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

    scanf("%s", infix);

    infixToPostfix(infix, postfix);

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

    return 0;
}
```

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