Rajalakshmi Engineering College

Name: Rithesh Madhav S
Email: 240701428@rajalakshmi.edu.in
Roll no: 240701428
Phone: 9884267696
Branch: REC
Department: I CSE FD
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

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;

void push(char c){
   stack[++top] = c;
}
char pop(){
   return stack[top--];
}
char peek(){
   return stack[top];
}
int precedence(char c){
   if(c == '^') return 3;
   if(c == '*' || c == '/') return 2;
```

```
    if(c == '+' || c == '-') return 1;
    return 0;
}

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

void infixToPostfix(char *infix,char *postfix){
    int i, k = 0;
    for(i=0; infix[i] !='\0'; i++){
        char c = infix[i];
        if(isdigit(c) || isalpha(c)){
            postfix[k++] = c;
        }else if (c=='('){
            push(c);
        }else if(c==')'){
            while(top !=-1 && peek() != '('){
                postfix[k++]  = pop();
            }
            if(top != -1) pop();
        }else if (isOperator(c)){
            while(top !=-1 && precedence(peek()) >= precedence(c)){
                postfix[k++] = pop();
            }
            push(c);
        }
    }
    while(top!=-1){
        postfix[k++] = pop();
    }
    postfix[k] = '\0';
}

int main(){
    char infix[MAX],postfix[MAX];
    scanf("%s",infix);
    infixToPostfix(infix,postfix);
    printf("%s\n",postfix);
    return 0;
}
```

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

2. Problem Statement

You are required to implement a stack data structure using a singly linked list that follows the Last In, First Out (LIFO) principle.

The stack should support the following operations: push, pop, display, and peek.

Input Format

The input consists of four space-separated integers N, representing the elements to be pushed onto the stack.

Output Format

The first line of output displays all four elements in a single line separated by a space.

The second line of output is left blank to indicate the pop operation without displaying anything.

The third line of output displays the space separated stack elements in the same line after the pop operation.

The fourth line of output displays the top element of the stack using the peek operation.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 11 22 33 44

Output: 44 33 22 11

33 22 11
33

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct Node{
   int data;
   struct Node* next;
};

struct Node* top = NULL;
void push(int value){
   struct Node* newNode=(struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = top;
   top = newNode;
}
void pop(){
   if(top!=NULL){
     struct Node* temp = top;
     top = top->next;
     free(temp);
  }
}
void display(){
   struct Node* current = top;
   while(current!=NULL){
      printf("%d ",current->data);
      current = current->next;
   }
   printf("\n");
}
void peek(){
   if(top!=NULL){
      printf("%d\n",top->data);
   }
}

int main(){
   int a,b,c,d;
   scanf("%d %d %d %d",&a,&b,&c,&d);
   push(a);
   push(b);
   push(c);
```

```
    push(d);
    display();
    printf("\n");
    pop();
    display();
    peek();
    return 0;
}
```

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

3. 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

```
// You are using GCC
```

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

#define MAX 50

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

void push(char c) {
    if(top < MAX - 1)
       stack[++top] = c;
}
char pop(){
if(top >= 0)
      return stack[top--];
     return '\0';
}
char peek(){
    if(top >= 0)
        return stack[top];
    return '\0';
}
int precedence(char op){
    if(op == '+' || op == '-') return 1;
    if(op == '*' || op == '/') return 2;
    return 0;
}
int  isOperator(char c){
    return c == '+' || c == '-' || c == '*' || c == '/';
}

void infixToPostfix(char* infix, char* postfix){
    int i = 0, j =0;
    char c;
    while(infix[i]){
        c = infix[i];
        if(c == ' '){
            i++;
            continue;
        } else if (isdigit(c)) {
            postfix[j++] = c;
        }else if(c == '(') {
            push(c);

        }else if(c == ')'){
            while (peek() !='(')
              postfix[j++] = pop();
             pop();
        }else if (isOperator(c)){
            while(precedence(peek()) >= precedence(c))
              postfix[j++] = pop();
              push(c);
        }
        i++;
    }
    while(top!=-1)
        postfix[j++] = pop();
    postfix[j] = '\0';
}

int main(){
    char infix[MAX], postfix[MAX];
    fgets(infix,MAX,stdin);
    infixToPostfix(infix, postfix);
    printf("%s\n",postfix);
    return 0;
}
```

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