Rajalakshmi Engineering College

Name: shreenidhi t
Email: 240701503@rajalakshmi.edu.in
Roll no: 240701503
Phone: 9150942326
Branch: REC
Department: I CSE FE
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

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

```
// You are using GCC
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#define MAX 100
float stack[MAX];
int top=-1;
void push(float value){
   stack[++top]=value;
}
float pop(){
   return stack[top--];
}
float evaluatePostfix(char*expr){
   int i;
   for(i=0;expr[i]!='\0';i++){
      if(isdigit(expr[i])){
         push(expr[i]-'0');
      }
      else{
         float b=pop();
         float a=pop();
         switch(expr[i]){
            case'+':push(a+b);break;
            case '-':push(a-b);break;
            case '*':push(a*b);break;
            case '/':push(a/b);break;
```

```
        }
    }
    }
    return pop();
}
int main(){
    char expr[MAX];
    scanf("%s",expr);
    printf("%.0f",evaluatePostfix(expr));
    return 0;
}
```

Status : Correct *Marks : 10/10*

2. Problem Statement

Raj is a software developer, and his team is building an application that processes user inputs in the form of strings containing brackets. One of the essential features of the application is to validate whether the input string meets specific criteria.

During testing, Raj inputs the string "(([])){}". The application correctly returns "Valid string" because the input satisfies the criteria: every opening bracket (, [, and { has a corresponding closing bracket),], and }, arranged in the correct order.

Next, Raj tests the application with the string "([)]". This time, the application correctly returns "Invalid string" because the opening bracket [is incorrectly closed by the bracket), which violates the validation rules.

Finally, Raj enters the string "{[()]}". The application correctly identifies it as a "Valid string" since all opening brackets are matched with the corresponding closing brackets in the correct order.

As a software developer, Raj's responsibility is to ensure that the application works reliably and produces accurate results for all input strings, following the validation rules. He accomplishes this by using a method for solving such problems.

Input Format

The input comprises a string representing a sequence of brackets that need to be validated.

Output Format

The output prints "Valid string" if the string is valid. Otherwise, it prints "Invalid string".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: (([])){}
Output: Valid string

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
int isMatchingPair(char opening,char closing)
{
   return ( opening=='('&&closing==')')||(opening=='{'&&closing=='}')||
(opening=='['&&closing==']');
}
int isValidString(char*str){
   char stack[MAX];
   int top=-1;
   for(int i=0;str[i]!='\0';i++){
```

```
        char ch=str[i];
        if(ch=='('||ch=='{'||ch=='['){
            stack[++top]=ch;
        }
        else if(ch==')'||ch=='}'||ch==']'){
            if(top==-1||!isMatchingPair(stack[top--],ch)){
                return 0;
            }
        }
    }
    return (top==-1);
}
int main()
{
    char input[MAX];
    scanf("%s",input);
    if(isValidString(input)){
        printf("Valid string\n");
    }
    else{
        printf("Invalid string\n");
    }
    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 100
char stack[MAX];
int top=-1;
void push(char ch){
   stack[++top]=ch;
}
char pop(){
   if(top==-1)return-1;
   return stack[top--];
}
char peek(){
   if(top==-1)return-1;
   return stack [top];
}
int precedence(char ch){
   if(ch=='+'||ch=='-') return 1;
   if(ch=='*'||ch=='/') return 2;
   return 0;
}
int isOperator(char ch){
   return ch=='+'||ch=='-'||ch=='*'||ch=='/';
}
int main(){
```

```
    char infix[MAX],ch;
    scanf("%[^\n]",infix);
    for(int i=0;i<strlen(infix);i++){
        ch=infix[i];
        if(ch==' ') continue;
        if(isdigit(ch)){
            printf("%c",ch);
        }
        else if(ch=='('){
            push(ch);
        }
        else if(ch==')'){
            while(peek()!='('){
                printf("%c",pop());
            }
            pop();
        }
        else if(isOperator(ch)){
            while(precedence(peek())>=precedence(ch)){
                printf("%c",pop());
            }
            push(ch);
        }
    }
    while(top!=-1){
        printf("%c",pop());
    }
    printf("\n");
    return 0;
}
```

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