Rajalakshmi Engineering College

Name: Sneha Raju R
Email: 240701519@rajalakshmi.edu.in
Roll no: 240701519
Phone: 7550004064
Branch: REC
Department: 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

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>
struct node{
   char data;
   struct node *next;
};
typedef struct node Node;
Node *top=NULL;

void push(char sym){
   Node *newnode=(Node *)malloc(sizeof(Node));
   newnode->data=sym;
   newnode->next=NULL;
   if(top==NULL){
      top=newnode;
      return;
   }
   newnode->next=top;
   top=newnode;
}

char pop(){
   Node *temp=top;
   char popped=top->data;
   top=top->next;
   free(temp);
   return popped;
}
```

```
char peek(){
    return top->data;
}

int isoperand(char sym){
    if(isalnum(sym))
        return 1;
    else
        return 0;
}

int prec(char sym){
    if(sym=='^')
        return 3;
    if(sym=='*' || sym=='/')
        return 2;
    if(sym=='+' || sym=='-')
        return 1;
    return 0;
}

void convert(char* inw, char* po){
    char in[100];
    int j=0;

    int le=strlen(inw);
    for(int i=0;i<le;i++){
        if(inw[i]!=' '){
            in[j++]=inw[i];
        }
    }
    int l=strlen(in);
    int k=0;
    for(int i=0;i<l;i++){
        char ch=in[i];
        if(isoperand(ch)){
            po[k++]=ch;
        }
        else if(ch=='('){
            push(ch);
        }
```

```
        else if(ch==')'){
            while(top!=NULL && peek()!='('){
                po[k++]=pop();
            }
            pop();
        }
        else{
            while(top!=NULL && peek()!='(' && prec(peek())>=prec(ch)){
                po[k++]=pop();
            }
            push(ch);
        }
    }

    while(top!=NULL){
        po[k++]=pop();
    }
    po[k]='\n';
    printf("%s",po);
}

int main(){
    char inw[100];
    char po[100];
    fgets(inw,sizeof(inw),stdin);
    convert(inw,po);
    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

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node{
   char data;
   struct node *next;
};
typedef struct node Node;
Node *top=NULL;

void push(char ele){
   Node *newnode=(Node *)malloc(sizeof(Node));
   newnode->data=ele;
   newnode->next=top;
   top=newnode;
}

char pop(){
   if(top==NULL)
      return '\0';
   else{
      Node *temp=top;
      char popped=top->data;
      top=top->next;
      free(temp);
      return popped;
   }
}

int ismatching(char open,char close){
   if((open=='(' && close==')') || (open=='{' && close=='}') || (open=='[' && close==']'))
{
      return 1;
   }else{
      return 0;
   }
}
int main(){
   char c;
   char exp[100];
   fgets(exp,sizeof(exp),stdin);
   exp[strcspn(exp,"\n")]='\0';
```

```
    for(int i=0;i<strlen(exp);i++){
        if(exp[i]=='(' || exp[i]=='{' || exp[i]=='['){
            push(exp[i]);
        }
        else if(exp[i]==')' || exp[i]==']' || exp[i]=='}'){
            char c=pop();
            if(!ismatching(c,exp[i])){
                printf("Invalid string");
                return '\0';
            }
        }
    }
    if(top!=NULL){
        printf("Invalid string");
    }else{
        printf("Valid string");
    }
    return 0;
}
```

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

3. 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

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int num1,num2,num3,num4;
    scanf("%d %d %d %d",&num1,&num2,&num3,&num4);
    printf("%d %d %d %d\n",num4,num3,num2,num1);
    printf("\n");
    printf("%d %d %d\n",num3,num2,num1);
    printf("%d",num3);
    return 0;
}
```

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