Rajalakshmi Engineering College

Name: NETHRA CHANDRAGANDHI T
Email: 240701357@rajalakshmi.edu.in
Roll no: 240701357
Phone: 9487531086
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

Buvi is working on a project that requires implementing an array-stack data structure with an additional feature to find the minimum element.

Buvi needs to implement a program that simulates a stack with the following functionalities:

Push: Adds an element onto the stack.Pop: Removes the top element from the stack.Find Minimum: Finds the minimum element in the stack.

Buvi's implementation should efficiently handle these operations with a maximum stack size of 20.

Input Format

The first line of input consists of an integer N, representing the number of elements to push onto the stack.

The second line consists of N space-separated integer values, representing the elements to be pushed onto the stack.

Output Format

The first line of output displays "Minimum element in the stack: " followed by the minimum element in the stack after pushing all elements.

The second line displays "Popped element: " followed by the popped element.

The third line displays "Minimum element in the stack after popping: " followed by the minimum element in the stack after popping one element.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
5 2 8 1

Output: Minimum element in the stack: 1
Popped element: 1
Minimum element in the stack after popping: 2

Answer

```
// You are using GCC
#include<stdio.h>
#define MAX 20
int stack[MAX],minStack[MAX];
int top=-1,minTop=-1;

void push(int value){
   if(top<MAX-1){
      stack[++top]=value;
      if(minTop == -1 || value <= minStack[minTop]){
         minStack[++minTop]=value;
      }
   }
}
```

```
int pop(){
   if (top==-1){
      return -1;
   }
   int popped=stack[top--];
   if(popped==minStack[minTop]){
      minTop--;
   }
   return popped;
}

int getMin(){
   if(minTop==-1){
      return -1;
   }
   return minStack[minTop];
}

int main(){
   int n,i,value;
   scanf("%d",&n);
   for(int i=0;i<n;i++){
      scanf("%d",&value);
      push(value);
   }

   printf("Minimum element in the stack: %d\n",getMin());

   int popped=pop();
   printf("Popped element: %d\n",popped);

   printf("Minimum element in the stack after popping: %d\n",getMin());

   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<stdlib.h>
#include<string.h>
#include<ctype.h>
#define MAX 100
```

```
typedef struct{
   int top;
   char items[MAX];
} Stack;
void initStack(Stack* stack){
   stack->top=-1;
}
int isEmpty(Stack* stack){
   return stack->top==-1;
}
void push(Stack* stack,char item){
   if(stack->top<MAX-1){
      stack->items[++stack->top]=item;
   }
}
char pop(Stack* stack){
   if(!isEmpty(stack)){
      return stack->items[stack->top--];
   }
   return '\0';
}
char peek(Stack* stack){
   if(!isEmpty(stack)){
      return stack->items[stack->top];
   }
   return '\0';
}
int precedence(char operators){
   switch(operators){
      case '+':
      case '-':
         return 1;
      case '*':
      case '/':
         return 2;
      case '^':
         return 3;
      default:
         return 0;
   }
}
void infixToPostfix(char* infix,char* postfix){
    Stack stack;
    initStack(&stack);
    int j=0;

    for(int i=0;infix[i];i++){
        char c=infix[i];
        if(isalnum(c)){
            postfix[j++]=c;
        }
        else if(c == '('){
            push(&stack,c);
        }
        else if(c == ')'){
            while(!isEmpty(&stack) && peek(&stack) != '('){
                postfix[j++]=pop(&stack);
            }
            pop(&stack);
        }
        else{
            while(!isEmpty(&stack) && precedence(peek(&stack)) >= precedence(c)){
                postfix[j++]=pop(&stack);
            }
            push(&stack,c);
        }
    }
    while(!isEmpty(&stack)){
        postfix[j++]=pop(&stack);
    }
    postfix[j]='\0';
}
int main(){
    char infix[MAX];
    char postfix[MAX];

    printf("");
    fgets(infix, sizeof(infix), stdin);
    infix[strcmp(infix, "\n")] = 0;

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

    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<string.h>
#include<ctype.h>
#define MAX 50

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

```
void push(char ch){
   stack[++top]=ch;
}
char pop(){
   return stack[top--];
}

char peek(){
   return stack[top];
}

int precedence(char op){

   switch(op){

      case '+':
      case '-': return 1;
      case '*':
      case '/':
      case '%': return 2;
      default: return 0;

   }
}

int main(){
   char expr[MAX];
   fgets(expr,MAX,stdin);

   for(int i=0;i<strlen(expr);i++){
      char  ch=expr[i];

      if(ch==' ')
        continue;
      if(isdigit(ch)){
         printf("%c",ch);
      }else if(ch == '('){
         push(ch);
      } else if(ch == ')'){
         while(top != -1 && peek() != '(')
            printf("%c",pop());
```

```
          if(top!=-1)
             pop();
       }
       else if(ch == '+' || ch == '-' || ch == '*' || ch == '/'){
          while(top != -1 && precedence(ch) <= precedence(peek()))
             printf("%c",pop());
          push(ch);
       }
    }
    while(top != -1){
       printf("%c",pop());
    }

    return 0;
}
```

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