Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering &Technology

Name :Dumpeta Varshitha . Email :varshithadumpeta@gmail.com
Roll no:23071A1023 Phone :8919552327
Branch :VNRVJIET Department :EIE
Batch :2027 Degree :B.Tech EIE

2023_27_II_EIE 1_Data Structures_lab

DATA STRUCTURES_CODING_WEEK 2

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : CODING

1. Problem Statement

Write a program that converts an infix expression to its postfix form using a stack.

Answer

```
// You are using GCC
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
//stack type
struct Stack
{
    int top;
    unsigned capacity;
    char *array;//change the data type to char
};
//Stack operations
struct Stack *createStack(unsigned capacity)
{
```

```
    struct Stack*stack=(struct Stack*)malloc(sizeof(struct Stack));
    if(!stack)
    return NULL;
    stack->top=-1;
    stack->capacity=capacity;
    stack->array=(char*)malloc(stack->capacity*sizeof(char));//change the data
type to char
    if(!stack->array)
    return NULL;
    return stack;
}
int isEmpty(struct Stack*stack)
{
    return stack->top==-1;
}
char peek(struct Stack *stack)
{
    return stack->array[stack->top];
}
char pop(struct Stack *stack)
{
    if(!isEmpty(stack))
return stack->array[stack->top--];
return '$';
}
void push(struct Stack *stack,char op)
{
    stack->array[++stack->top]=op;
}
//A utility function to check if the given character is an operand
int isOperand(char ch)
{
    return(ch>='a'&&ch<='z')||(ch>='A'&&ch<='Z');
}
//A utility function to return the precedence of a given operator
//Higher returned value means higher precedence
int prec(char ch)
{
    switch (ch)
    {
        case '+':
        case '-':
```

```
        return 1;
        case '*':
        case '/':
        return 2;
        case '^':
        return 3;
    }
    return -1;
}
//The main function thet converts the given infix expression
//to prefix expression.
void infixToPostfix(char*exp)
{
    int i,k;
    //Create a stact of capacity equal to expression size struct
    struct Stack *stack=createStack(strlen(exp));
    if(!stack)//see if the stack was created successfully return;
    return;
    for(i=0,k=-1;exp[i];++i)
    {
        //if the scanned character is an operand,add it to output.
        if(isOperand(exp[i]))
        exp[++k]=exp[i];
        //if the scanned character is an '(',push it to the stack.
        else if(exp[i]=='(')
        push(stack,exp[i]);
        //if the scanned characternis an')',pop and output from the stack
        //until an'(' is en countered.
        else if(exp[i]==')')
        {
            while(!isEmpty(stack)&&peek(stack)!='(')
            exp[++k]=pop(stack);
            if(!isEmpty(stack)&&peek(stack)!='(')
            return;//invalid expression
            else
            pop(stack);
        }
        else//an operator is encountered
        {
            while(!isEmpty(stack)&&prec(exp[i])<=prec(peek(stack)))
            exp[++k]=pop(stack);
            push(stack,exp[i]);
```

```
        }
    }
    //Pop all the elements from the stack
    while(!isEmpty(stack))
    exp[++k]=pop(stack);
    exp[++k]='\0';
    printf("%s",exp);
}
//Drive program to test the above functions
int main()
{
    char exp[100];
    //printf("enter infix expression\n");
    scanf("%s",exp);
    infixToPostfix(exp);
    return 0;
}
```

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

2. Problem Statement

You are required to implement a program that converts an infix expression to its corresponding postfix expression. The infix expression contains operands (single-character variables, uppercase or lowercase), operators (+, -, *, /, ^), and parentheses. The program should output the postfix expression.

Answer

```
// You are using GCC
#include<stdio.h>
#include<ctype.h>
#define max 100
char stack[100];
char postfix[100];
char infix[100];
int top=-1;
void push(char);
char pop(void);
int precedence(char);
```

```
int main()
{
   int i=0,j=0;
   char c;
   scanf("%s",infix);
   while(infix[i]!='\0')
   {
      if(infix[i]=='('
      push(infix[i]);
      else if(isalpha(infix[i]))
      postfix[j++]=infix[i];
      else if(infix[i]==')')
      {
         while(stack[top]!='('
         postfix[j++]=pop();
         c=pop();
      }
      else
      {
         while(precedence(stack[top])>=precedence(infix[i]))
         postfix[j++]=pop();
         push(infix[i]);
         }
         i++;
   }
   while(top!=-1)
   postfix[j++]=pop();
   postfix[j]='\0';
   printf("%s",postfix);
}
void push(char c)
{
   top++;
   stack[top]=c;
}
char pop(void)
{
   return(stack[top--]);
}
int precedence(char c)
{
   switch(c)
```

```
    {
        case '*':
        case '/':return (5);
        break;
        case '+':
        case '-':return(3);
        break;
        case'(':return(0);
    }
}
```

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

3. Problem Statement

Given string S representing a postfix expression, the task is to evaluate the expression and find the final value. Operators will only include the basic arithmetic operators like *, /, + and -.

Answer

```
// You are using GCC
#include<stdio.h>
#include<ctype.h>
int stack[100];
int top=-1;
void push(int c)
{
    stack[++top]=c;
}
char pop(void)
{
    return(stack[top--]);
```

```
}
int main()
{
   int a,b,i;
   char postfix[100];
   scanf("%s",postfix);
   for(i=0;postfix[i]!='\0';i++)
   {
      if(isdigit(postfix[i]))
      push(postfix[i]-'0');
      else
      {
         a=pop();
         b=pop();
         switch(postfix[i])
         {
            case '+':push(b+a);
            break;
            case '-':push(b-a);
            break;
            case '*':push(a*b);
            break;
            case '/':push(b/a);
            break;
         }
      }
   }
   printf("%d",stack[top]);

}
```

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

4. Problem Statement

Write a program to check whether the parentheses in a given expression are balanced or not. The expression may contain parentheses of three types: (), {}, and []. A balanced expression has each opening parenthesis closed by a corresponding closing parenthesis in the correct order.

Answer

```
// You are using GCC
#include<stdio.h>
char stack[100];
int top=-1;
void push(char c){
   stack[++top]=c;
}
char pop(){
   return stack[top--];
}
int main() {
   int i;
   int flag=1;
   char s[100];
   scanf("%s",s);
   for(i=0;s[i]!='\0';i++){
      if(s[i]=='{'|| s[i]=='('){
         push(s[i]);
      }else if(s[i]=='['|| s[i]=='('){
         push(s[i]);
      }else if(s[i]=='}'|| s[i]==']'|| s[i]==')'){
         if(top==-1){
            flag=0;
            break;
         }else{
            char c=pop();
            if((s[i]=='}'&&c!='{') || (s[i]==']'&& c!='[') || (s[i]==')'&& c!='(')){
               flag=0;
               break;
            }
         }
      }
   }
   if(flag==1 && top==-1){
      printf("BALANCED");
   }else{
      printf("NOT BALANCED");
   }
   return 0;
}
```

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