S.No: 7	Exp. Name: ***Write a C program to Convert an Infix expression into Postfix expression***	Date:2024-06-02

ID: 92132321021

Aim:

Write a program that contains the functions `convertInfix(char *e)`, `priority(char x)`, `push(char x)`, `pop()` and `isEmpty` so that it uses stack operations to converts an expression in Infix to Postfix.

Note: The max size of stack is 20

Source Code:

Infix2PostfixMain.c

```
#include<stdlib.h>
#include<string.h>
#include<stdio.h>
#include<ctype.h>
#define STACK_MAX_SIZE 20
#include "Infix2PostfixOperation.c"
int main() {
   char exp[20];
   char *e, x;
   printf("Enter the expression : ");
   scanf("%s",exp);
   e = exp;
   convertInfix(e);
}
```

Infix2PostfixOperation.c

```
// write your code here
char stack[STACK_MAX_SIZE];
int top = -1;


int isEmpty() {
return top ==-1;

}


void push(char x) {
stack[++top] = x;
//printf("%c",stack[top]);

}


char pop() {
return stack[top--];

}
```

```
int isop(char x){
 return (x == '+' || x == '-' || x == '*' || x == '/' || x =='^');
}

char associate(char x){
if (x == '^')
return 'R';
return 'L';
}


int priority(char x) {
if (x == '*' || x == '/')
return 2;
else if (x == '^')
return 3;
else if(x=='+'||x=='-')
return 1;
else
return -1;

}

void convertInfix(char * e) {
char postfix[20];
int k = -1, balance =0;
int len = strlen(e);
for (int i = 0; i<len;i++){
char c = e[i];
if(isalnum(c)){
postfix[++k] = c;
}
else if (c == '('){
push(c);
balance++;
}
else if (c== ')' && !isEmpty()){
while(stack[top] != '(' && !isEmpty()){
postfix[++k] = pop();
}
pop();
balance--;
}
else if(isop(c)){
while(!isEmpty() && (priority(stack[top]) >= priority(c)) && associate(c)== 'L'){
postfix[++k] = pop();
}
push(c);
}
else{
printf("Invalid symbols in infix expression\n");
exit(0);
}
}
while(!isEmpty()){
postfix[++k] = pop();
```

```
//printf("%s",postfix);
}
postfix[++k] = NULL;
if(balance != 0){
printf("Invalid infix expression : unbalanced parenthesis\n");
exit(0);
}
printf("Postfix expression : %s\n",postfix);



}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the expression : A+B*(C-D)
Postfix expression : ABCD-*+

Test Case - 2
User Output
Enter the expression : A+B*(C+D)*E+(F*G
Invalid infix expression : unbalanced parenthesis

Test Case - 3
User Output
Enter the expression : A+B*(S+W&L)
Invalid symbols in infix expression

Test Case - 4
User Output
Enter the expression : A+B&(C-D
Invalid symbols in infix expression