INDEX

Ex. No.:3

Date :

CONVERSION OF INFIX EXPRESSION TO POSTFIX

Aim

To write a C-program to convert the given infix expression to its postfix format.

Algorithm to convert infix to postfix notation

Let two stacks opstack and poststack are used and otos & ptos represents the opstack top and poststack top respectively.

```
1. Start

2. Initialize stacks

3. Scan one character at a time of an infix expression from left to right

4. Repeat till there is data in infix expression

    4.1 if scanned character is '(' then push it to opstack
    4.2 else if scanned character is operand then push it to poststack
    4.3 else if scanned character is operator then
    if(otos!=-1)
    while(precedence (opstack[otos])>=precedence(scan character)) then
        pop from opstack and push it into poststack
        push (scan character) into opstack

    otherwise
        push (scan character) into opstack
    4.4 else if scanned character is ')' then
    pop and push into poststack until '(' is not found and ignore both brackets

5. pop and push into poststack until opstack is not empty.
6. return
```

Program

```
/*program to convert infix to postfix expression*/
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<string.h>
#include<ctype.h>

int precedency(char);

int main()
{
	int i,otos=-1,ptos=-1,len,length;
	char infix[100],poststack[100],opstack[100];
	printf("Enter a valid infix:\n");
	gets(infix);
	length=strlen(infix);
	len=length;
	for(i=0;i<=length-1;i++)
	{
		if(infix[i]=='(')
		{
			opstack[++otos]=infix[i];
			len--;
		}
		else if(isalpha(infix[i]))
		{
			poststack[++ptos]=infix[i];
		}
		else if (infix[i]==')')
		{
			len--;
			while(opstack[otos]!='(')
			{
				poststack[++ptos]=opstack[otos];
				otos--;
			}
			otos--;
		}else //operators
		{
			if(precedency(opstack[otos])>=precedency(infix[i]))
			{
				poststack[++ptos]=opstack[otos--];
			}
				opstack[++otos]=infix[i];

		}
	}

	while(otos!=-1)
	{
		poststack[++ptos]=opstack[otos];
		otos--;
	}
	/********for displaying**************/
	for(i=0;i<len;i++)
	{
```

```
                printf("%c",poststack[i]);
        }
        getch();
        return 0;
}

/***************precedency function********************/
int precedency(char ch)
{
        switch(ch)
        {
                case '$':
                case '^':
                        return(4);
                        // break;
                case'*':
                case'/':
                        return(3);
                        // break;
                case'+':
                case'-':
                        return(2);
                        // break;
                default:
                        return(1);
        }
}
```

Output:

```
Enter the infix expression :: (a+b)/(c*d)

Postfix Expression is ::  ab+cd*/
```

Manual Calculation: (A+B)/(C-D+E)+F-G

SE EXPRESSION	STACK	RESULT FIELD
(	(	
A	(	A
+	(+	A
B	(+	AB
)		AB+
/	/	AB+
(	/ (	AB+
C	/ (	AB+C
-	(/ -	AB+C
D	(/ -	AB+CD
+	(/ +	AB+CD-
E	(/ +	AB+CD-E
)		AB+CD-E+/
+	+	AB+CD-E+/
F	+	AB+CD-E+/F
-	-	AB+CD-E+/F+
G	-	AB+CD-E+/F+G
		AB+CD-E+/F+G-