NEEHAL - 1BM19CS097

Lab Program-2

WAP to convert a given valid parenthesized infix arithmetic expression to postfix exp'n. The exp'n consists of single character operands and the binary operators +(plus), -(minus), *(multiply) and / (divide).

```
#include <stdio.h>
# include <string.h>
# include <process.h>
int F (char symbol)
{
Switch (symbol)
{
 Case '+':
 Case '-': return 2;
 Case '*':
 Case '/': return 4;
 Case '^':
 Case '$': return 5;
 Case '(': return 0;
 Case '#': return -1;
 default : return 8;
 }
}
int G (char symbol)
{
Switch (Symbol)
{
Case '+':
Case '-': return 1;
Case '*':
Case '/': return3;
Case '^':
Case '$': return 6;
Case '(': return 9;
Case ')': return 0;
default: return 7;
}
}
Void infix-postfix (Char infix[], Char postfix[])
{
int top, i, j;
Char s[30], Symbol;
top = -1;
S[++ top] = '#';
j = 0;
for (i=0; i< strlen (infix); i++)
{
Symbol = infix [i];
While (F(S[top]) > G (symbol))
{
postfix[j] = S[top--];
j++;
}
if (F (S[top]) != G (Symbol))
S[++ top] = Symbol;
else
top--;
}
While (S[top] != '#')
{
```

```
postfix[j++]= s[top--];
}
postfix[j]='\0';
}
void main()
{
char infix[20];
char postfix[20];
printf("enter the valid infix expression\n");
scanf("%s",infix);
infix_postfix(infix,postfix);
printf("the postfix exp is\n");
printf("%s\n", postfix);
getch();
}
```