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

```
#include<stdio.h>
#include<stdlib.h>
#include<string.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 '/': return 3;
case '^':
case '$': return 6;
case '(': return 9;
case ')': return 0;
default: return 7;
 }
}
void infix_postfix(char infix[]){
 int top=-1, j, i;      j=0;
 char S[30], postfix[30];
 char symbol;      int i=0;
 for(i=0; i<strlen(infix); i++){
     symbol = infix[i];
   while(FS[top] > G(symbol)){
       S[++top] = symbol;
     }
   else
     top--;
 }
 while[s[top]!='#'){
   postfix[j++]=S[top--];
 }
   postfix[j]='\0';
   puts(postfix);
}
  int main()
{ char exp[30];
```

```
printf("Enter an expression ");
gets(exp);
infix_postfix(exp);
return 0;
}
```