Program-3 Infix to postfix conversion

WAP to convert a given infix arithmatic expression to postfix expression. The expression consists of single character operands and binary operators.

```
#include <stdio.h>
#include <ctype.h>
# include < string.h>
#define N 100
int stack[N];
int top = -1;
void push (char c)
{ if (top== N-1)
  { printf (" ~~Overflow~~ Stack overflow\n");
    return;
  }
  Stack[++top] = c;
  return;
 }
Char pop(){
  if (top==-1)
  { printf(" Stack Underflow\n");
    return -1;
  }
  return stack[top--];
}
Char peek(){
  if (top==-1)
  { printf ("Stack Underflow\n");
    return -1;
  }
  return stack[top];
}
```

Pseudocode.

```
int is_operator(char c)
{ return (c=='^' || c=='*' || c=='/' || c=='+' ||
          c=='-' || c=='%');
}
int precedence (char c)
 { switch(c)
   { case '^' : return 3;
      case '*' :
      case '%' :
      case '/' : return 2;
      case '+' :
      case '-' : return 1;
      default : return 0;
   }
 }
void infix to postfix (char *infix , ~~char *~~ postfix)
 { int k=0;
   char symbol;
   push('(');
   int len= strlen(infix);
   infix(len]= ')';
   infix[len+1] = '\0';
   for (int i=0 ; i<=len ; i++)
   { ~~Entrde~~-Symbol = infix[i];
     if ( isalnum (symbol){
          postfix[k++] = Symbol;
```

```
else if ( symbol = '(')
 { push ('('); }
else if ( isoperator (symbol))
 { while (top!=0 && precedence (peek()) > precedence (symbol))
      { postfix[k++] = pop();
      }
   push (symbol);
   } else if (symbol ==')'){
      while (top!=1 && peek()='('){
         postfix [k++] = pop();

      } pop();
   }
  } postfix[k]='\0';
 }
~~int main()~~
{ ~~char infix[N], postfix[N];~~
  ~~printf ("Enter an infix expression~~
```

Output :

```
Enter an infix expression
(a+b) * (c-d)
The postfix expression is : ab+cd-*
```

[illegible]