Lab-2

Write a program to convert the given valid infix expression to postfix expression. The expression consist of single character operands and the binary operators +,-,* and / .

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#define max 100
char stack[max];
int top=-1;
void push(char c)
{
  if(top==max-1)
  {
    printf("stack overflow\n");
  }
  else
  {
    top=top+1;
    stack[top]=c;
  }
}
char pop()
{

  char val;
  if(top==-1)
  {
    printf("stack underflow\n");
    return -1;
  }
  else
  {
    val=stack[top];
    top=top-1;
    return val;
  }
}

char peek()
{
  if(top==-1)
  return '\0';
  return stack[top];
}

int precedence(char c)
{
  if(c=='+'||c=='-')
    {return 1;}
```

```
   if(c=='*'||c=='/')
     {return 2;}
return 0;
}
void infixtopostfix(char infix[],char postfix[])
{
   int i,k=0;
   char c;
   for(i=0;infix[i]!='\0';i++)
   {
      c=infix[i];
      if(isalnum(c))
      {
         postfix[k]=c;
         k=k+1;
      }
      else if(c=='(')
      {
         push(c);
      }
      else if(c==')')
      {
         while(top!=-1&&peek()!='(')
         {
            postfix[k]=pop();
            k=k+1;
         }

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

```
int main()
{
   char infix[max],postfix[max];
   printf("enter the valid parenthesized infix expression:");
   scanf("%s",&infix);
   infixtopostfix(infix,postfix);
   printf("postfix expression:%s\n", postfix);
   return 0;
}
```