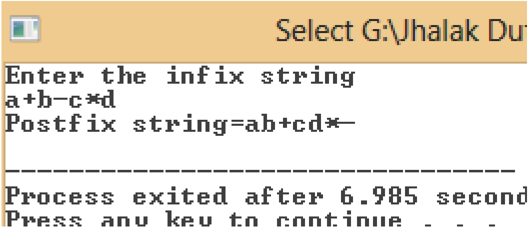
Write a program to convert an infix expression into its equivalent postfix notation.



```
Enter the infix string
a+b-c*d
Postfix string=ab+cd*-

--------------------------------
Process exited after 6.985 second
Press any key to continue . . .
```

```
#include<stdio.h>
#define SIZE 40
char stack[SIZE];
int top=-1;

void push(char data)
{
   if(top==SIZE-1)
   {
      printf("Stack is full\n");
      return;
   }
   else
   {
      top=top+1;
      stack[top]=data;
      //printf("Pushed element is %c\n",data);
   }
}

char pop()
{
   char ch;
   if(top<0)
   {
      //printf("stack is empty\n");
      return;
   }
   else
   {
      ch=stack[top];
      //printf("poped element is%c\n",ch);
      top=top-1;
      return(ch);
   }
}

int check_pre(char a ,char b)
{
   //operators are arranged in the array based
   //on their priority. from low to high
```

```
    char op[]={'-','+','%','/','*','(',')'};
    int i,c1=0,c2=0;
    for(i=0;i<7;i++)
    {
        if(a==op[i])
            c1=i+1;
        else if(b==op[i])
            c2=i+1;
    }
    if(c1>c2)
        return(1);
    else if(c1<c2)
        return(-1);
    else
        return(0);
}


int main()
{
    char in_str[50],out_str[50];
    char ch,temp;
    int x=0,y=0,pre;
    printf("Enter the infix string\n");
    scanf("%s",in_str);
    ch=in_str[x];
    while(ch!='\0')
    {
        //for operand
        if((ch>='a' && ch<='z') ||
(ch<='A' && ch>='Z') ||
(ch>='0' && ch<='9'))
               out_str[y++]=ch;
        //for '(' paranthesis
        else if(ch=='(')
            push(ch);
        //for ')' parenthesis
        else if(ch==')')
        {
            temp=pop();
            while(temp!='(')
            {
                out_str[y++]=temp;
                temp=pop();
            }
          //  if(temp=='(')
                // pop();

        }
        //for operator
        else
        {
        //if the stack is empty or
        // the stack top element is '('
        //just push the operator in to the stack
            if (top==-1 || stack[top]=='(')
                push(ch);
            else
            {
                temp=stack[top];
                //check the precedence
                pre=check_pre(ch,temp);
                if(pre<0 )
                {
                    do{
                    out_str[y++]=pop();
                    temp=stack[top];
}while(top!=-1 && temp!='(' &&  (check_pre(ch,temp)<0));
                    push(ch);
                }
                else
                {
                    push(ch);
                }
            }
        }
        x++;
        ch=in_str[x];
    }
    while(top!=-1)
    {
        out_str[y++]=pop();
    }
    out_str[y]='\0';
    printf("Postfix string=%s\n",out_str);
    return;
}
```