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

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

```
        top=top-1;
        return val;
    }
}

int precedence(char c){
    if(c=='+' || c=='-') return 1;
    if(c=='/' || c=='*') return 2;
    return 0;
}

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

void infixToPostfix(char infix[],char postfix[]){
    int k=0;
    char c;
    for(int 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];
    char postfix[MAX];
```

```
    printf("Enter a valid parenthesized infix expression: \n");
    scanf("%s",infix);
    infixToPostfix(infix,postfix);
    printf(" Postfix Expression: %s\n",postfix);

return 0;
}
```

Output: