1. Write code for Infix to postfix conversion by using stack (The code should be able to treat with parenthesis, braces and at least the following operators: +, −, ×, /, mod).

本人的lab支持计算{}，[], (), +, -, \*, /, %, >>, >>>, <<, ’（lab中把这个符号设为乘方，即2’2=2^2=4）, log，^,&,|

Infix至postfix的转化：ArrayList<String> inToPost(String inputStr)

具体实现：

①调用ArrayList<String> checkExpression(inputStr)函数检查输入字符串的数学表达正确性，本人在此处判断了{}[]()的顺序（即必须以{[()]}的顺序进行输入），以及判断了两个非括号操作符（如+\*）无法连用，以及去除了字符串表达式中的空格

同时此函数还以操作符为分割将字符串分为Arraylist数组infixArr

②函数主体部分：判断，遇到运算符号压入栈，遇到数字直接存入用于存放postfixArr字符的arraylist数组（本人的lab将{[(都以(的形式压入栈中以便计算，这样遇见)]}时只用寻找栈中上一个（的位置）。在结束infixArr循环后，将栈中剩余的符号pop进postfixArr。

③又使用了一个循环，将postfixArr里的（{[去除

④return postfixArr数组，并在main函数里使用循环输出postfix表达式。

p.s.

boolean isNum(String str) 判断某字符串是否是一个数字（整数&小数）

boolean isOper(String str) 判断某字符串是否是一个本lab可用的操作符

boolean ifPrior(String str Stack stack) 判断某运算符是否比当前栈顶运算符优先级高，高返回true。本人lab中判断优先级的方式为：

String partern = "(^<+\*l'0&0-/000|";

if((partern.indexOf(str)%7) > (partern.indexOf(stack.peek())%7)) return true;

else return false;

用mod7得数来区别优先级。

Boolean ifPush(String str, Stack stack) 用于判断{}[]()的左括号是否可以压入新的栈中，本人lab判断括号顺序的方式是将左括号以一个优先级顺序压入栈中，遇到左括号时判断栈顶是否非配对的右括号。不是则提示输入错误。

2. Write code for Postfix expression evaluation by using stack.

Postfix表达式的计算：double calaulate(ArrayList<String> postfixArr)

具体实现：

postfixArr遍历循环

判断，是数字则压进栈中，是操作符则调用

void judgeOperAndCal(Stack stack, String str)，运用switch对不同的运算符进行运算。并pop进行运算的数字，push进运算完毕的结果。

最后栈中只剩一个数字结果，pop的同时作为本函数的返回值，即最后的结果。