1. 程序实现功能

- 完成类型检查，对于存在语义错误的输入，输出语义错误信息（错误类型编号及出错的token首个字符所在行的行号）。
- 对于不存在语义错误的输入，打印对特定变量重命名后的语法树。

2. 如何实现

2.1 类型检查

1. 类型类（Type, ArrayType, FunctionType）、符号类（Symbol、BaseSymbol、BasicTypeSymbol、ConstantSymbol、FunctionSymbol、VariableSymbol）、作用域类（Scope、BaseScope、GlobalScope、LocalScope）均按照老师上课示例设计。FunctionType类中增添了ArrayList paramsNames属性，用于之后的变量重命名。
2. 建立作用域：重写了visitProgram、visitFuncDef和visitBlock三个方法，在里面添加了scope的新建和切换。思路都是新建scope，遍历子节点，然后回到父scope。
3. 定义符号：分析SysY的语法可知，函数声明、函数形参、常量声明和变量声明四处需要定义符号，因此重写visitFuncDef、visitFuncFParam、visitConstDecl、visitVarDecl四个方法，在其中利用currentScope.define添加符号到当前作用域。
4. 遍历语法树并发现语义错误：此处我以错误类型5（赋值号两侧类型不匹配）为例进行说明：
    - 首先分析错误类型5以及SysY的语法，发现这一错误类型只可能出现在constDef、varDef和stmt的第一种情况（我取名为StmtAssign）三处。此处我以StmtAssign为例进行说明，其他两处处理方法是类似的。

```
stmt : lVal ASSIGN exp SEMICOLON # StmtAssign | (exp)? SEMICOLON # Stmt2| block # Stmt3
       | IF L_PAREN cond R_PAREN stmt ( ELSE stmt )? # Stmt4
       | WHILE L_PAREN cond R_PAREN stmt # Stmt5
       | BREAK SEMICOLON # Stmt6 | CONTINUE SEMICOLON # Stmt7
       | RETURN (exp)? SEMICOLON # Stmt8;
```

    - 重写visitStmtAssign方法，先得到lValContext和expContext，分别代表左值和右值。并用两个变量leftDimension和rightDimension记录左值和右值的维度。
    - 先计算leftDimension：使用currentScope.resolve(lValContext.IDENT().getText())得到左边的符号left。如果它的类型为FunctionSymbol，那么会报错；如果是BasicTypeSymbol，那么leftDimension为0；如果类型是ArrayType，那么用原来这个数组的维度arrayType.getNum()减去这次取了多少次下标lValContext.exp().size()，就得到了实际的维度。
    - 再计算rightDimension：如果expContext类型为NumContext，则维度为0；如果类型为LvalContext，那么和左值一样分为FunctionSymbol、BasicTypeSymbol和ArrayType三种情况计算rightDimension；如果类型为CallContext，那么先解析符号得到functionSymbol，再得到这个符号的返回类型。如果返回类型是int，那么rightDimension为0，否则报错。
    - 最后比较leftDimension和rightDimension，如果两个值不相等，则报错。

2.2 变量重命名

我使用了助教提供的第一种思路：在BaseSymbol类中添加了ArrayList<ArrayList> usedPosition属性，产生符号表时，用usedPosition额外保存该符号曾在哪一行哪一列被使用，之后可以遍历整个符号表根据保存的信息得到哪一个变量需要被重命名。在打印语法树时，重写visitTerminal函数，一旦发现token的起始行列在这个变量的usedPosition中就不打印原本名称而是变更后的名称。

3. 精巧设计

我在两次遍历语法树（一次类型检查，一次打印语法树）的时候，使用的是用一个visitor。第一次遍历的时候，进行新建作用域、解析符号、类型检查等操作，并且使用一个全局变量HashMap<String, Scope> map记录作用域名和作用域的对应关系。第二次遍历的时候，由于对作用域取名字的规则是相同的，我就从map中直接取出定义好的作用域并进入，而不会新建作用域，这样所有的符号表信息就得到了。在visitTerminal时，如果这个node是IDENT，直接使用currentScope.resolve(node.getText())就可以得到相应的符号，并利用这个符号的usedPosition属性得到它使用过的位置信息。对于不需要切换作用域的语句，第二次遍历时只需要visitChildren(ctx)这一个操作。虽然这样有点麻烦，需要在每个重写的函数里判断是第一次还是第二次遍历，并相应地做不同的操作，但是我用这种方法重命名的四个用例一下子满分了，都没有debug，所以我觉得还是不错的。

4. 印象深刻的bug

- hardtest02和03找了很久的bug，hardtest03的报错是Your output contains ground truth but also contains 1 extra incorrect output，多输出了错误，所以我以为一行内有多个变量未声明或变量重复声明等错误的话只会报一次错，找了很久bug。后来问了助教说一行内多次变量重复声明会报多次错，改了这个bug就都对了。
- normaltest06一直差十分，后来发现忘记考虑!这个运算符的情况了。

这次的bug实在是太多了...一开始写完类型检查只有几百分，断断续续debug一周才ac，不过还是很有成就感的。