COP4020 Programming Assignment 8

To study and practice tail recursion optimizations applied by hand, let’s improve the performance of the Java CalcAST application by applying tail recursion optimizations. Note that powerful C compilers such as gcc and some Java compilers will do this automatically.

Recall that tail recursion optimization changes a recursive call into a jump (goto) statement back to the beginning of the function. The arguments to the function are set accordingly to ensure that the jump passes the new argument values to the beginning of the function. This requires overwriting the argument values with updated values if necessary.

For example, when we have a Java method such as

```java
private int method(MyClass arg)
{
    if (...)
    {
        ...
        return method(someNewValue);
    }
    else
    {
        ...
        return method(someOtherNewValue);
    }
    return 0;
}
```

we can transfer control from the return statements back to the beginning of the method to avoid the overhead of the recursive call. Before the jump, we have to reassign the method’s parameters to mimic parameter passing as is illustrated in the following example:

```java
private int method(MyClass arg)
{
    /* start: */
    if (...)
    {
        ...
        arg = someNewValue;
        /* goto start */
    }
    else
    {
        ...
        arg = someOtherNewValue;
        /* goto start */
    }
    return 0;
}
```

Fortunately (!), Java does not support goto statements, so you will have to figure out a way to implement the jump back to the start of the method to achieve a loop. In addition, methods may have more than one argument, so be careful to handle the argument passing correctly.
First obtain your CalcAST.java program (and AST.java) that you implemented earlier for programming assignment 7. Modify the program CalcAST.java by elimination of tail-recursion for the following two methods:

```java
    term_tail
    factor_tail
```

You are **not** required to implement the simplification as outlined in programming assignment 7, but you can leave them in if you have a working version of the program for assignment 7. Therefore, your program may either output the simplified or non-simplified Lisp representation.