INF 212 Analysis of Programming Languages
Project 6 – Type Systems
Due date: 5/13
Demo date: 5/14

Goal: Get in-depth knowledge of type checking.

Code base: http://www.ics.uci.edu/~lopes/teaching/inf212W12/projects/INF212-Types.zip
The code provided performs all the functionalities of a compiler except type checking which is left for you to implement. The challenge in this assignment is not the programming part, but to really understand how the entire process works.

Brief description relevant to the assignment:

One of the main functionalities of type checking is the collection of information of the symbols used in the program i.e. preparation of the "Symbol Table". This table associates the symbol with its declaration.
e.g. int x
Symbol table stores this information ----> x, its scope (global/local) and its type -----> int

The code for this has been provided. At this point you can safely assume that symbol resolution has passed and the program contains no symbol-related semantic errors.

The compiler parses the source code and constructs the intermediate representation for this code. The "DumpAST.java" traverses this intermediate representation and outputs the same source program if there is no syntax or semantic error.

It is this file that you need to work on and make sure it throws a sematic error if there is a type checking error. This is the only file you can and need to work on.

At any moment you can check the contents of this symbol table by using the DUMP function for the symbol table.

Due to time constraints, we'll keep the semantic rules simple:

- While and if conditions must be of type bool.
- Boolean operators &&, || and ! must involve operand(s) of type bool.
- All comparisons must be of identical types.
- All assignments must be of identical types.
- For multiplication, division, and mod, both operands must be int. Likewise for unary + and -.
- For addition and subtraction, the rules are a bit more complicated:

- Both types must be int, or
- Both types must be char, or
- One type can be char and the other must be int, or
- One type can be a pointer and the other must be an int.

Making it simpler, compute the type of an expr E, perform type-checking and throw a Semantic Error if the types are incompatible.

Test cases are provided in Tests/Type.

As of now, for each of the test cases, there is no error! Once you correctly implement type checking, all of them should give an error.

What to turn in: a zip file with the entire code again, this time with your additions in it.