Homework #5: CMPT-379

Distributed on Nov 27; due on Dec 4

Anoop Sarkar – anoop@cs.sfu.ca

Only submit answers for questions marked with †. Provide a `makefile` such that `make` compiles all your programs, and `make test` runs each program, and supplies the necessary input files.

(1) † **Decaf Compiler**

Submit your compiler as a self-contained package that can be used to compile **Decaf** programs into MIPS assembly and subsequently execute them using the `spim` simulator for the MIPS processor. Make sure that the binary of your compiler can be created by running `make`.

Create a script called `decafcc` (or `decafcc.sh`) that is used to run the entire compiler chain from lexical analysis to code generation to running the MIPS simulator (assume `spim` is in the PATH).

In your submission, provide in a subdirectory called `positives` any number of **Decaf** programs that work with your compiler (the programs should be valid **Decaf** based on the language definition and execute using `spim`) along with the legitimate output for that **Decaf** program, e.g. for a program called `exprTest.decaf` also include the legitimate output in a file called `exprTest.decaf.output`. Also provide a subdirectory called `negatives` with **Decaf** programs that should exit with an error. Your `makefile` should include an entry such that when `make testall` is run, it should run your **Decaf** compiler on *all* the **Decaf** programs in the `positives` and `negatives` directory.

Please check that you do not have any spelling errors in the names of the directories (to enable automatic testing). Also, for non-trivial **Decaf** programs, provide a readme file explaining the code and the desired output or why it should not produce any output (the syntax or semantic error involved). For `exprTest.decaf` provide a readme file called `exprTest.decaf-readme.txt`.

Note that in your `makefile`, to ignore errors in a command line that executes a program, write a '-' at the beginning of the line's text (after the initial tab). The '-' is discarded before the command is passed to the shell for execution. For example,

```
test:
        -sh decafcc.sh negatives/exprTest.decaf > negatives/exprTest.decaf.output
```

You could try to break the compilers written by your peers, but only if your compiler can survive those **Decaf** programs itself.

The grade for this homework will be determined by reading the source code of your compiler implementation as well as the performance on the global set of positive and negative **Decaf** programs.