1 Introduction

c4d is a stripped-down version of Dennis Ritchie's and Ken Thompson's C, for the purpose of learning compiler design in a nutshell (i.e. in 8 weeks).

c4d is essentially C, minus structures, unions, pointers, print functions and typedefs. Constructions related to these simplifications (e.g. malloc and free) are subsequently eliminated. Since we're all C programmers, thanks to DR and KT, we know what else must be eliminated to that effect.

There are four types, including `int, float` and `char`. There is only one kind of array, a `vector` (i.e. a one-dimensional array). It has the special syntax, e.g. below, where the constant after ':' is its size. There can be comma-separated vectors of same size.

```
int vector x:20;
```

From the C instruction set, the following are eliminated: `switch, case, default, goto`. Preprocessor commands (`#`) and global declarations are eliminated too.

Four basic arithmetic operators and three basic logical operators are kept (and, or, not). Comparative operators are as usual. There is only one assignment operator (=), rather than += etc.etc.

All rules of C apply (for parameter passing, array indices, declare before use, type compatibility, start of execution, etc.etc.)

I/O is very simple. There are two built-in functions, to print space-separated values, or to take space-separated values. Examples are below:

```
output(x,y,25.4);
input(x,y);
```

There is no **c4d**-internal file management; all input redirects from standard input, and output to standard output. Report your errors to standard error.

In this project, there is no bonus work. We will not accept a superset of **c4d** as a better design, because we already have the superset's perfection in our hands floating around publicly, namely the C compilers. The perfection in specs endures as Kernighan and Ritchie's *C language* book.

2 Phase 1: Due November 23, 2011

In this phase you are to design a lexical analyzer and parser for **c4d**. The output must be the abstract syntax tree. antLR has tools for pretty-printing them. Do some error reporting too. We expect one error report for every major construction.

3 Phase 2: Due January 6, 2012

In this phase, you upgrade your parser to a code generator, to generate MIPS code as output. You will execute the object code as well, using *spim* as the MIPS emulator. All inputs in this phase will be well-formed **c4d** programs.

4 Submissions

For both phases, we submit through COW.

Bundle your files as a `tar`-compatible single archive. Your submission should contain only one archive file, to be extracted to a *dotted* current directory `'./444phase1-gn'` where gn is your group motto or identifier.

For example, for phase1, this archive file *should* only include the following files with exact names:

`c4d.g` : antLR code for **c4d** analyser

`Main.java` : your java code to call **c4d** analyser

`Makefile` : to make the system so that we can run it as `$ java Main input-file`

`phase1-data/input1` : Your own experimental data for us to see performance at your end. Name your data as `input1.c4d`, `input2.c4d`, ...

`README.txt` : You must include **a description of who did what part of the work.** If you have something special to draw our attention, this is where it goes. Ideally, it should contain no special intructions.

Keep it clean. Keep it simple. Remember, you're not just a pretty face, you're a bloody computer scientist. Dennis is watching.

Happy compiling, Cem Bozşahin