Compiler Theory: Assignment 1

Marc Moreno Maza
University of Western Ontario
CS447a – Fall 2004

Posted: Thursday, September 23, 2004
Due: Thursday, October 14, 2004

1 Guidelines

Date of postage. Thursday, September 23, 2004.

Normal date for submission. Thursday, October 14, 2004 (no penalties).

Last date for submission. Tuesday, October 19, 2004 (late penalties apply from Friday 15).

Due format. The assignment should be returned to the instructor by email. Your submission will have the form of a UNIX *archive* created with the UNIX command `tar`. This archive will have a structure similar to those of the example archives available on the course web site. More precisely, a `src` directory will contain your `flex` (or `lex`) source programs, a `test` directory will contain your test files, a `Makefile` will define the targets and a `README` will explain these targets. Moreover, your comments or answers to the assignment questions (that are not computer programs) will be made in a file

- which may be a text file, or a `postscript` file or a `pdf` file (No other formats are allowed. Moreover, I recommend that you generate your postscript and pdf files under a recent UNIX/Linux distribution.)

- and which will be placed in a `doc` directory.

In addition, you could have a `bin` directory for binaries (as in the course example archives) and a `ps` directory for postscript versions of computer programs (this is convenient to print them!). The computer programs of this assignment are all input files for the `flex` (or `lex`) program, and standard C program or header files. C++ is **not** allowed. The preferred version of `flex/lex` is `flex version 2.5.4`. Please, contact me if you were using a more recent one.

Your `Makefile` must work under Linux (with a recent distribution) or under a UNIX system of the GAUL network. You can reuse the Makefile of the archive `lex.tar.gz` from the course web site. Note that this Makefile was developed with `bash` under Red Hat 7.3 (Linux) and using GCC version 2.96.

Your `README` must specify which UNIX system you are using (Linux or SunOS) and which shell (command language interpreter) you are using (`tcsh` or `bash`). For the compilation options of `flex/lex` please refer to the Makefile of the archive `lex.tar.gz`.

Late penalties. See the course outlines.

Collaboration. You are expected to do this assignment on your own without assistance from anyone except from the instructor.

Marking. This assignment will be marked out of 40. The allocation of marks is given below.

2 The Assignment

The objective is to write a lexer for the ALLCOT language. Please, refer to the document titled *The* ALLCOT *Language* and available on the course web site.

2.1 Lists for ALLCOT (6 marks)

Propose the addition of lists to the ALLCOT language, as a built-in data-structure. More precisely,

- Describe the semantics of your lists,
- Explain how to modify the grammar of ALLCOT in order to take your lists into account.

For this question, you may refer to the

- ALDOR documentation for the syntax and semantics of lists in this language,
- ALLCOT document to the study how arrays are defined, as a built-in data-structure.

2.2 A Lexer for ALLCOT (20 marks)

Write a lexical analyzer for your version of the ALLCOT language using `lex` or `flex`. Your lexer must

- recognize the tokens that are part of the ALLCOT language,
- print error messages for the strings that are not valid ALLCOT lexemes.

You will be using your lexer in future assignments, so design your lexer with this in mind.

2.3 Test Programs for ALLCOT (7 marks)

Write three ALLCOT test programs, each of them being about 30 lines long. Make sure that they cover all the syntax features of the language and that your lexer can process them correctly.

2.4 Preparing for Symbol Tables (7 marks)

Enhance your `lex` (or `flex`) program such that it can display all identifiers that are found after parsing an input ALLCOT program. In addition, for each of these identifiers, print

- its number of occurrences,
- the number of the line where it appears first.

Subsidiary question. Can you modify your `lex` (or `flex`) program such that it can detect identifiers that are used (or defined) before they are declared?