Wellesley College ◊ CS301 Compiler Design ◊ September 15, 2000
Handout #06

PROBLEM SET 2

Due on Friday, September 29 *(PS1 is also due on this day.)*

READING

- Appel, Chapter 2 (Lexical Analysis)
- ML-Lex manual (http://nike.wellesley.edu/~cs301/doc/ML-Lex/manual.html).

OVERVIEW

The purpose of this problem set is to help you gain familiarity with the theory and practice of lexical analysis. On the theory side, you will work with DFAs, NFAs, and regular expressions. On the practice side, you will use ML-Lex to build a working lexical analyzer for Kitty.

This assignment has four problem worth a total of 100 points. There are two extra credit problems worth a total of 50 points.

SUBMISSION DETAILS

Each team should turn in a single hardcopy submission packet for all problems by slipping it under Lyn's office door by 5pm on the due date. The packet should include (1) written solutions to Problems 1, 2, and 3; (2) your final version of the file `Kitty.lex` from Problem 4; (3) transcripts of your test cases for Problem 4; and (5) any auxiliary files you created for Problem 4. Your hardcopy submission packet should also include a header sheet for each team member (see the end of this assignment for the header sheet), and should indicate where the softcopy submission can be found.

Your softcopy submission should consist of your local version of the `kitty/scanner` program directory described in Problem 4.

PROBLEM 1 [25]

For each of the five language descriptions in Appel Exercise 2.1, parts a through e, do the following:

i. Draw a finite automaton (DFA or NFA) accepting the language;
ii. Give a regular expression for the language.

PROBLEM 2 [10]

Appel Exercise 2.3. For part c, it helps to enumerate the smaller strings of the language and look for a pattern.

PROBLEM 3 [15]

Appel Exercise 2.5. In your DFAs, you may assume the existence of a non-final dump state that is the target of all transitions not explicitly listed.

PROBLEM 4 [50]

In this problem you will implement and test a lexical analyzer for Kitty. Lexical conventions for Kitty are explained in the Kitty Reference Manual (Handout #03).

You should create your lexical analyzer using the ML-Lex lexical analyzer generator. ML-Lex is briefly described in Section 2.5 of Appel. A more detailed description may be found in the on-line ML-Lex manual.

Start the problem by making a local copy of the directories `~cs301/download/kitty/scanner` and `~cs301/download/kitty/test`. You should make both of these subdirectories of your local `kitty` directory from PS1. The `scanner` directory contains two SML source files that you should study.

1. A file `Token.sml` that specifies the structure of Kitty tokens and some operations on tokens. You should study this file.
2. A file `Scanner.sml` that contains a few top-level functions for running the Kitty scanner.

The `test` directory contains a small suite of Kitty programs. You will want to extend this suite as part of your testing.

Your task is to define an ML-Lex lexical analyzer specification file `Kitty.lex` that describes the structure of Kitty tokens. Study the ML-Lex documentation to understand how a .lex file is structured.

Once you have define Kitty.lex, you can generate a lexical analyzer by invoking the following in a Unix shell (not in the `*sml-cm*` buffer!):

```
ml-lex Kitty.lex
```

If successful, this will create a file `Kitty.lex.sml` that is an automatically generated SMLNJ program for scanning tokens according to the specification file.

To test the generated scanner, execute the following in the `*sml-cm*` buffer:

```
CM.make'("Scanner.cm");
Scanner.printTokensInFile(filename-goes-here);
```

For example, suppose that `../test/count.kty` contains the following Kitty program:

```
/* Count the number of characters in the input stream */

let count := 0;
    c := readc()
 in while c >= 0 do
     (count := count + 1;
      c := readc());
    writei(count)
end;
```

Here is the result that should be produced by scanning `../test/count.kty`:

```
- Scanner.printTokensInFile("../test/count.kty");
[LET <60, 63>]
[ID("count") <64, 69>]
[GETS <70, 72>]
[INT(0) <73, 74>]
[SEMI <74, 75>]
[ID("c") <80, 81>]
[GETS <82, 84>]
[READC <85, 90>]
[LPAREN <90, 91>]
[RPAREN  <91, 92>]
[IN <94, 96>]
[WHILE <97, 102>]
[ID("c") <103, 104>]
[GEQ  <105, 107>]
[INT(0) <108, 109>]
[DO <110, 112>]
[LPAREN <118, 119>]
[ID("count") <119, 124>]
[GETS <125, 127>]
[ID("count") <128, 133>]
[ADD <134, 135>]
[INT(1) <136, 137>]
[SEMI <137, 138>]
[ID("c") <145, 146>]
[GETS <147, 149>]
[READC <150, 155>]
[LPAREN <155, 156>]
[RPAREN  <156, 157>]
[RPAREN  <157, 158>]
[SEMI <158, 159>]
[WRITEI <164, 170>]
[LPAREN <170, 171>]
[ID("count") <171, 176>]
[RPAREN  <176, 177>]
[END <178, 181>]
[SEMI <181, 182>]
val it = () : unit
```

A few notes:

- It is helpful to include `open Token` at the top of your `Kitty.lex` file. This makes all token operations available without qualification.
- Any time you modify your `Kitty.lex` file, you need to first remake Kitty.lex.sml (by invoking `ml-lex Kitty.lex` in a shell) before you invoke `CM.make'("Scanner.cm")` in the `*sml-cm*` buffer.
- Any ML errors in your `Kitty.lex` file will not be caught until the invocation of `CM.make'("Scanner. cm")`. You will need to fix such errors in your `Kitty.lex` file, which necessitates generating the scanner again before you can see if your fixes worked.
- The `.lex` file is divided into three sections. You can only use ML-style comments in (1) the topmost section and (2) within the parenthesized right-hand-sides of productions in the third section. As far as I can tell, the second and third sections do not support any other commenting conventions. (There is no documentation about comments in the ML-Lex manual, and no one has responded to my query about this on tbe `comp.lang.ml` newsgroup.)
- Test your scanner on a range of Kitty files that exercise all possible token types. Include in your submission the test Kitty programs and transcripts of the scanner acting on these.

EXTRA CREDIT 1 [15]

Give a formal proof that your description of the automaton in Appel Exercise 2.3(c) is correct.

EXTRA CREDIT 2 [25]

Appel Exercise 2.7.

Problem Set Header Page
Please make this the first page of your hardcopy submission.

CS251 Problem Set 2
Due Friday, September 26, 2000

Names of Team Members:

Date & Time Submitted:

Soft Copy Directory:

Collaborators (*any teams collaborated with in the process of doing the problem set)*:

*In the **Time** column, please estimate the total time each team member spent on the parts of this problem set. (Note that spending less time than your partner does not necessarily imply that you contributed less.) Please try to be as accurate as possible; this information will help me to design future problem sets. I will fill out the **Score** column when grading your problem set.*

Part	**Time For** _____ **(Team Member #1)**	**Time For** _____ **(Team Member #2)**	**Score**
General Reading			
Problem 1 [25]			
Problem 2 [10]			
Problem 3 [15]			
Problem 4 [50]			
Extra Credit 1 [15]			
Extra Credit 2 [15]			
Total			