CS 164, Spring 2005 **CS 164: Homework #1** **P. N. Hilfinger**

Due: Wednesday, 2 February 2005

General instructions about homework. Please use the **submit** command to turn in your homework. Your account must be properly configured for CS164 first. You'll find templates for some solutions in the directory `~cs164/hw/hw1`. Place answers to any questions that don't require programs in a file called **hw1.txt**. The command to use for submission is '**submit hw1**', which you should issue inside the directory containing your submission. If you do not understand any of these instructions, *FIND OUT ABOUT THEM!* Consider it part of the course material.

Problem 3 calls for **JFlex** programs. **JFlex** is available on your class accounts (as `~cs164/bin/jflex`). Problems 4 and 5 call for the use of the **fsasim** program, which is available from your instructional account (it is in `~cs164/bin/arch/`M`/bin`, where M is either sun4u or i86pc depending on your workstation). There are manuals for **Jflex** and **fsasim** available in the Miscellaneous Tools section of the CS164 home page.

1. A *subsequence* of a string S (or any kind of sequence, really) is simply a sequence consisting of zero or more characters from S in the same order as in S; for example, `"ack"`, `""`, `"bk"`, and `"back"` are all subsequences of `"back"`, but `"kb"` is not. A *substring* of S is a *contiguous* subsequence of zero or more characters of S; for example `"ack"`, `"ba"`, `""` and `"back"` are substrings of `"back"`, but `"bk"` is not. Given a string S of length N, how many subsequences does it have? How many substrings? To simplify your life, you *may* assume that letters in S are not repeated (the problem becomes rather "interesting" if you count only distinct subsequences when S may contain repetitions).

2. [From Aho, Sethi, Ullman] Give as simple a description as possible of the languages denoted by the following regular expressions. For example, it is better to describe `((a*b*)*(b*a*)*)*` as "The set of all strings from the alphabet $\{a, b\}$" rather than "A sequence of 0 or more a's followed by 0 or more b's, all repeated 0 or more times and then followed by *etc.*" The latter might be correct, but shows no thought.

a. `0(0|1)*0`

b. `((`ϵ`|0)1*)*`

c. `(0|1)*0(0|1)(0|1)`

d. `0*10*10*10*`

e. `(00|11)*((01|10)(00|11)*(01|10)(00|11)*)*`

More problems on next page.

3. [Adapted from Aho, Sethi, Ullman] Write the simplest regular expression you can for each of the following languages, using the notation of `jflex`. Turn in files called **P3a.lex**, **P3b.lex**, etc., each containing that contain the `jflex` solutions to these problems, using the templates for these files provided in **~cs164/hw/hw1** on the instructional machines. In the following, "letters" mean lower-case letters.

a. Strings of letters that contain all five vowels in order, exactly once.

b. Strings composed of letters a–f in which the letters are in alphabetical order (not all letters have to appear in a string, but those that do must be in order).

c. Comments consisting of a string starting with `/*` and ending with `*/` without any `*/` in between unless it occurs inside balancing quotes `"` and `"`. Quotes must be balanced, so that `/*"*/` is illegal, and only balancing pairs of quotes "deactivate" the `*/` symbol, so that `/*""*/""*/` is illegal (the `*/` occurs between quotes, but they don't balance each other).

d. All strings of 0's and 1's with an even number of 0's and an odd number of 1's.

e. All strings of 0's and 1's that do not contain the substring '011'.

f. All strings of 0's and 1's that do not contain the subsequence '011'.

4. Find the simplest NFA you can for problem 2c, above. Turn in a file called **2c.nfa** containing your answer in **fsasim** format.

5. Find the simplest DFA you can for 2c. Turn in a file called **2c.dfa** containing your solution in **fsasim** form (see above).

6. Suppose that I have two NFAs, M_1 and M_2, recognizing the languages L_1 and L_2, respectively. Give a general algorithm for constructing an NFA that recognizes the language $L_1 - L_2$, which is the set of all strings in L_1 that is not in L_2. You don't need to write a program, just the give the algorithm in sufficient detail that a reasonable programmer could fill in the details.

7. Give the simplest description you can of the language described by the DFA below: