CS 164, Spring 2005 **CS 164: Homework #2** **P. N. Hilfinger**

Due: Wednesday, 9 February 2005

General instructions about homework. Use the command '`submit hw2`' to submit your homework. We will not accept homework in any other form. Unless the problem specifies otherwise, please put your solutions in a file named `hw2.txt`.

1. [From Aho, Sethi, Ullman] Indicate what language is described by each of the following grammars. In each case, S is the only non-terminal. Some symbols are quoted to make it clear that they are terminals.

a. S → 0 S 1 | 0 1

b. S → + S S | - S S | a

c. S → S "(" S ")" S | ϵ

d. S → a S b S | b S a S | ϵ

e. S → a | S + S | S S | S "*" | "(" S ")"

2. Identify each ambiguous grammar in problem 1 above, and give an unambiguous grammar that recognizes the same language (any such grammar—don't worry about associativity or precedence, since there are no semantic actions.)

3. For 1d above, give two distinct leftmost derivations for the string *abab*. For each derivation, show the corresponding parse tree and the rightmost derivation for that same parse tree.

4. [From Aho, Sethi, Ullman] Show that all binary (base 2) numerals produced by the following grammar denote numbers that are divisible by 3:

N → 11 | 1001 | N 0 | N N

Does this grammar generate all non-negative binary numerals that are divisible by 3?

5. A context-free grammar is *regular* if every production has either the form $A \rightarrow xB$, or the form $A \rightarrow x$, where A and B are non-terminals and x is a string of 0 or more terminal symbols. Show that the language described by any such grammar can be recognized by a NDFA (hence the term *regular*).

6. [From Aho, Sethi, Ullman] Try to design a context-free grammar for each of the following languages (it is not always possible). Whenever possible, make it a regular grammar.

a. The set of all strings of 0's and 1's where every 0 is immediately followed by at least one 1.

b. Strings of 0's and 1's with an equal number of 0's and 1's.

c. Strings of 0's and 1's with an unequal number of 0's and 1's.

d. Strings of 0's and 1's that do not contain the substring 011.

e. Strings of 0's and 1's of the form xy where x and y are equal-length strings and $x \neq y$.

f. Strings of 0's and 1's of the form xx.

7. Write a BNF grammar describing the language of boolean expressions whose value is `true`. The terminal symbols are '1' (true) , '0' (false), '`*`' (logical and), '`+`' (logical or), unary '`-`' (logical not) and left and right parentheses (for grouping). Assume the usual precedence rules, with logical "not" having highest precedence. That is, `1`, `1*1`, `1+0`, `1*1*-(0+1*0)`, and `-0` are all in the language, while `0`, `0+0`, prog—0*1—, and `1*1*(0+1*0)` are not. Your grammar may be ambiguous (that is, you may specify operator precedence and associativity separately). Put your solution in a file `7.y`. Start with the skeleton in `~cs164/hw/hw2/7.template`.

8. Write a Python program that reads text from the standard input, keeping track of the number of times each word in the text appears, and then prints out the ten most frequently occurring words that are at least 4 letters long, one per line, in order of decreasing frequency. A "word" here is a contiguous substring of letters; anything other than a letter delimits a word. Ignore case (and print words in lower case). Place the result in the file `count.py`.