CS 337 **Test 2** **4/2/08**

Open book and notes.

Max points = 75 Time = 75 min Do all questions.

1. (Finite State Machine Design; 25 points) Design finite state machines for the following problems. In each case, explain your solution by describing the properties of the states of the machine.

   (a) (7 points) For a binary string, let $n0$ denote the number of zeroes and $n1$ the number of 1s. Accept a binary string in which $n0 \leq n1 \leq n0 + 2$ for all prefixes of the string. Thus, ϵ, 1, 11 and 101 are acceptable whereas 0, 100 and 111 are not.

   (b) (5 points) Accept a binary string if the corresponding number is divisible by 3. Thus, ϵ, 0, 00, 11 and 1001 are acceptable whereas 1, 111 and 1000 are not.

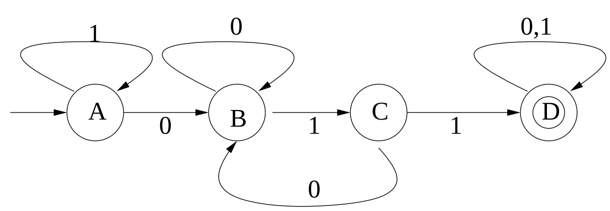
   (c) (6 points) Design a finite state transducer that accepts a string of symbols, and outputs the same string by (1) removing all white spaces in the beginning, (2) reducing all other blocks of white spaces (consecutive white spaces) to a single white space. Below, we denote a white space by `-`. Thus, the string

   `----Mary----had--a-little---lamb---`
   is output as
   `Mary-had-a-little-lamb-`

   (d) (7 points) Accept a simple arithmetic expression which is defined as follows. The expression is over single digits, operators + and × and parentheses "(" and ")", and it follows all the laws of well-formed expressions. The only restriction is that there is no nesting of parentheses. Thus, $3 + 4 \times 5$, $3 + (4 \times 5)$, $(3 + 4) \times (3 + 5)$ are acceptable whereas $3 + *4$, $3(+4 \times 5)$, $((3 + 4) \times 5)$ and $3 + ()$ are not.

2. (Finite State Machine Theory; 25 points)

   (a) (10 points) The following machine is claimed to accept any binary string that includes 011 as a substring. Prove this claim. (Show the theorems that need to be proved. You don't have to actually prove the theorems.)



(b) (4 points) Enumerate the strings of the language defined by the regular expression $(a \mid ab)(c \mid bc)$. Write another regular expression using both alternation and concatenation that denotes the same language.

(c) (11 points) Write regular expressions for the following languages over the alphabet $\{0, 1\}$.

i. (2 points) Each string (in the language) has at least one 1.

ii. (3 points) Each string has at most one 1.

iii. (2 points) Each string has exactly one 1.

iv. (4 points) Every block of 1s in a string is of even length.

3. (Functional Programming; 25 points) Code functions to solve the following. You will need to use `even`, `odd` and `` `div` `` in some of these problems.

(a) (8 points) Let `x` and `y` be nonnegative integers. Then `xor x y` returns an integer which is the exclusive-or of `x` and `y` treated as numbers written in binary. Thus, `xor 0 0 = 0`, `xor 0 1 = 1`, `xor 3 5 = 6` and `xor 3 3 = 0`.

(b) (8 points) Function g is defined as follows:

$$g(0) = 0$$
$$g(1) = 1$$
$$g(2n) = g(2n - 2) + g(2n - 1), \text{ for } n > 0$$
$$g(2n + 1) = g(2n - 2), \text{ for } n > 0$$

Write an efficient program to compute $g(m)$ for any m, $m \geq 0$.

(c) (4 points) Function f takes a nonnegative integer as input and returns an integer. Code function g which takes integer n as input, $n \geq 0$, and returns `True` if $f(0), f(1), \cdots, f(n)$ are all zero. Assume that the code for f is given elsewhere.

(d) (5 points) The definition of `power2` given in the notes uses the identity $2^{n+1} = 2 \times 2^n$. Use the following identities to construct a more efficient version of `power2` for all n, $n \geq 0$.

$$2^{2 \times n} = (2^n)^2$$
$$2^{2 \times n + 1} = (2^n)^2 \times 2$$

Do not use exponentiation in your solution.