Midterm Exam
General CS 2 (320102)

March 27, 2007

NAME:

MATRICULATION NUMBER:

You have one hour (sharp) for the test;
Write the solutions to the sheet.

You can reach 26 points if you solve all problems. You will only need 23 points for a perfect score, i.e. two points are bonus points.

You have ample time, so take it slow and avoid rushing to mistakes!

Different problems test different skills and knowledge, so do not get stuck on one problem.

To be used for grading, do not write into this box								
prob.	1.1	1.2	2.1	2.2	3.1	3.2	Sum	grade
total	2	4	4	5	5	6	26	
reached								

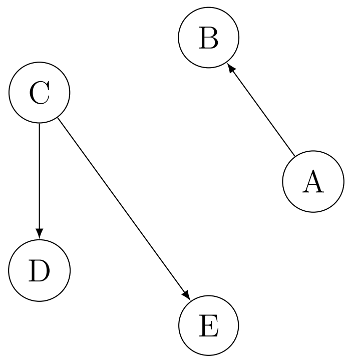
1 Graphs

Problem 1.1 (Directed Graphs) 2pt 6min

We call a directed graph (strongly) connected, iff for any two nodes $n_1 \neq n_2$ there is a path starting at n_1 and ending at n_2.

Complete the unconnected directed graph below by adding directed edges such that it becomes a strongly connected graph where each $indeg(n) = outdeg(n)$ for all nodes n.

How many initial and terminal nodes and how many paths does your final graph have?



Problem 1.2 (Constructing Fully Balanced Binary Trees in SML) 4pt 10min

Write an SML function `MakeTree` that takes an integer $n \geq 0$ and returns a fully balanced binary tree with n nodes if one exists, and raises an exception `WrongInput` otherwise. The following datatype is used to construct binary trees:

```
datatype btree = leaf | parent of btree*btree;
```

2 Combinatorial Circuits

Problem 2.1 (Right and Left Shift on PNS) 4pt 12min

Consider for this problem the signed bit number system and the two's complement number system. Given a binary string $b = a_n \ldots a_0$. We define

1. the left shift function $lshift$ that maps the $n+1$-bit number $a_n \ldots a_0$ to the $n+2$-bit number $a_n \ldots a_0 0$
2. the right shift function $rshift$ that maps the $n+1$-bit number $a_n \ldots a_0$ to the n-bit number $a_n \ldots a_1$, discarding a_0.

Prove or refute the following two statements

- The $lshift$ function has the same effect in both number systems; i.e. for any integer z:

$$\langle\!\langle lshift(B(z)) \rangle\!\rangle^- = \langle\!\langle lshift(B_n^{2s}(z)) \rangle\!\rangle_{n+1}^{2s}$$

- The $rshift$ function has the same effect in both number systems; i.e. for any integer z:

$$\langle\!\langle rshift(B(z))\rangle\!\rangle^- = \langle\!\langle rshift(B_n^{2s}(z))\rangle\!\rangle_{n-1}^{2s}$$

Problem 2.2 (A Binary Counter) 5pt 10min

Implement a 3-bit binary counter that counts from 111 down to 000 in steps of 1 and again starts from 111, doing one step with each clock pulse. You may use any combinatorial or sequential logic circuit that has been introduced in the lecture. Draw the circuit of your implementation with sufficient explanation.

Note: Assume that, initially, each storage element contains a 0.

3 Machine Programming

Problem 3.1 (Array Indexing in Assembler) 5pt 10min

Given $n \geq 1$, stored in $D(0)$, and $a_1, \ldots, a_n$, stored in $D(1), \ldots, D(n)$, with $1 \leq a_i \leq n, i = 1, \ldots, n$, compute the "$n$-th order subscript" $a_{a_{\ldots a_n}}$ of a and store it in $D(1)$.

Note: With the above definition, the first-order subscript of a would be a_n, the second-order subscript would be a_{a_n}, and so on. An example initial setup for $n = 3$ could be:

i	0	1	2	3
$D(i)$	3	3	1	2

In this case, the program should compute $a_{a_{a_n}} = a_{a_{a_3}} = a_{a_2} = a_1 = 3$.

Problem 3.2 (Static Procedure for Logarithm) 6pt 12min

Write a $\mathcal{L}$(`VM`) program that implements the `log` function for the integer logarithm defined as $\lfloor \log_b a \rfloor$ as a static procedure and calls that procedure to compute $\log_2 3$, as in the following μML listing (given in an SML-like syntax):

```
let
  fun log(b, a) =
    ...
in
  log(2, 3)
end
```

1. Complete the function in the above μML listing, using an an SML-like syntax.
2. Write down the $\mathcal{L}$(`VM`) program (in concrete, not abstract syntax) that results from compiling the μML program[1]. You may use any $\mathcal{L}$(`VM`) instruction except `peek` and `poke`.
3. Draw the evolution of the stack, including all intermediate steps.

Note: Assume a built-in `div` instruction that performs integer division. You may confine yourself to the cases $b > 1$ and $a > 0$.

[1] You need not remember the exact definition of the compiler. Just give a $\mathcal{L}$(`VM`) program that computes the same function as the μML program and explain to which lines of the μML program the parts of the $\mathcal{L}$(`VM`) code relate.