Faculty of Science
FINAL EXAMINATION

COMPUTER SCIENCE COMP 250
INTRODUCTION TO COMPUTER SCIENCE

Examiner:	Prof. Michael Langer	April 20, 2009
Associate Examiner:	Mr. Joseph Vybihal	9 A.M. – 12 P.M.

Instructions:

Read all questions before you start. The number of points per question is unrelated to the difficulty of the questions, so use your time wisely.

The exam is five pages (doublesided), including this cover page.

There are seven questions, worth a total of 30 points.

Answer all questions in the exam book. You may keep the exam question sheet.

No calculators, notes, or books are allowed.

1. **(3 points)**

   Consider the following binary numbers, which are each between 0 and 1:

   $$p_1 = .0011, \quad p_2 = .0101, \quad p_3 = .0001, \quad p_4 = .0101 \quad , p_5 = .001$$

   Define

   $$F_i = \sum_{k=1}^{i} p_k \ , \quad i = 1, 2, 3, 4, 5.$$

   (a) Write each F_i as a binary number.

   (b) Draw a complete binary tree of height $h = 4$ and *label the leaves* with labels from $\{1, \ldots, 5\}$ increasing left to right, such that p_i is the fraction of leaves with label i.

2. **(2 points)**

   (a) Describe what the following *recursive method* does. The method has one parameter which is a reference to a node in a list.

```
myPrintMethod(ListNode node){
   if (node.prev != null){
      myPrintMethod( node.prev );
   }
   System.out.println(" " + node.toString());
}
```

   (b) Give and solve a recurrence relation for the runtime of this method in terms of the number n of nodes in the list.

3. **(5 points)**

State the formal definitions of "$f(n)$ is $O(g(n))$", and "$f(n)$ is *not* $O(g(n))$".
Use these formal definitions to prove the following:

(a) $3n + 8$ is $O(n)$.

(b) $3n + 8$ is *not* $O(1)$.

(c) $6n$ is $O(2^n)$. (You *must* prove this one by induction.)

You will lose marks if your definitions or proofs take the limit as $n \to \infty$.

4. **(5 points)**

What is the $O(\,)$ for the runtime of the following operations, in terms of the number of elements n in the data structure ? If you are unsure, then state any assumptions you make.

Note: 'get' returns an element but does not 'remove' it.

(a) Get the i^{th} element from an array.

(b) Insert a new element into a sorted array.

(c) Add an element to the front of a linked list.

(d) Get the i^{th} element from a linked list.

(e) Remove the i^{th} element from a linked list.

(f) Add an element to a minHeap.

(g) Get the minimum element from a minHeap.

(h) Get the maximum element from a minHeap.

For (f)-(h), assume the heap is represented using an array.

5. **(3 points)**

   (a) Show the final contents a *queue*, after the following sequence of operations have been performed. Be sure to indicate the front and back.

```
initialize empty queue
enqueue(a)
enqueue(b)
enqueue(c)
dequeue()
enqueue(d)
dequeue()
enqueue(e)
enqueue(f)
dequeue()
enqueue(g)
enqueue(h)
```

   (b) Suppose we implement the queue as a circular queue, using an *array* that is initialized with size $m = 4$. Show the contents of the array after *each* of the above operations have been performed. Note that the number of `enqueue` operations is five more than the number of `dequeue` operations, so it will be necessary at some point to increase the size of the array.

6. **(4 points)**

   (a) What is the maximum number of nodes in a *binary tree* of height h ?

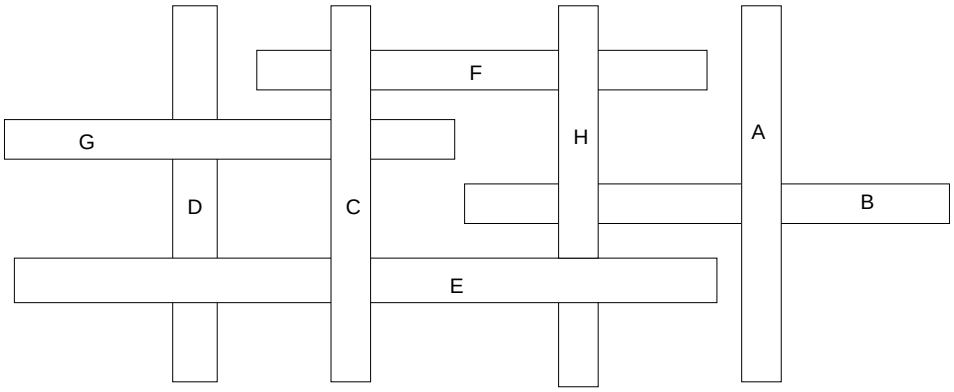
   (b) What is the sum of the depths of all nodes in a *binary tree* of height h, assuming all levels of the tree are full? It is sufficient to write a summation expression here. You are not required to evaluate this summation.

   (c) Give a *recursive* algorithm `NumNodes(node)` for computing the number of nodes in a tree whose root node is the argument. The tree is not necessarily a binary tree.

   (d) Give a *recursive* algorithm `SumDepths( ... )` for computing the sum of the depths of all nodes in a rooted tree. You will need to choose the argument(s) appropriately. As in (c), the tree is not necessarily binary.

7. **(8 points)**

The figure below shows a set of rectangles whose overlap relationships can be represented with a directed graph. Each rectangle is represented by one vertex, and there is a directed edge whenever one rectangle overlaps another rectangle. For example, there is an edge `(A,B)`.



(a) Give the adjacency list for this graph. *The vertices must be ordered alphabetically.*

(b) Draw the graph, labelling the vertices `A,B,C,D,E,F,G,H`.

(c) Give the ordering of vertices visited in a *breadth first search* traversal of the graph, starting from vertex `C`. *Show the BFS tree.*

NOTE: In this question and the next, there is only one answer since you must order the vertices alphabetically.

(d) Give the ordering of vertices visited in a *depth first search* traversal, starting from vertex `C`. *Show the DFS tree.*

(e) Give a valid topological sorting of the vertices in this graph.

(f) Give a *new* example having four rectangles `I,J,K,L`, such that corresponding graph contains a *cycle*. Draw the rectangles and the graph.

```
1.a)     F1 =  .0011
         F2 =  .1000
         F3 =  .1001
         F4 =  .1110
         F5 = 1.0000

  b)    The labels are 1112222234444455.

MARKING SCHEME:  (3 points in total)

      2 points for the F values.
      1 point  for the labels

---------------------------------------------------------------


2. a)   It prints the nodes in the list, starting from the front of
        the list and up to (and including) the argument node.

   b)   t(n) = c1 + t(n-1)
             = c1 (n-1) + c2

MARKING SCHEME:  (2 points in total)

   1 point for (a).

   1 point for (b).
        0.5 point for t(n) = 1 + t(n-1)
        0.5 point for saying 't(n) is O(n)'.
        (0   points for the base case, i.e. we ignored it..)

---------------------------------------------------------------

3.  a)    3n + 8 <=  3n + 8n  for n>=1
                  =  11n      for n>=1   so take n0 = 1, c = 11.

    b)   I want to show that no matter what c or n0 I choose,
         there exists an n> n0 such that   3n + 8 > c .

              3n + 8 > c
     <->           n > (c - 8)/3   so I just take n > (c-8)/3

    c)   I want to show there exists c, n0 such that
          6n <= c 2^n   for any n > n0.
```

Proving this by induction is subtle. I start by
choosing c=6 and n0 = 1. Then, the statement I
want to prove by induction is:
"6n <= 6 2^n for all n >=1", which is equivalent
"n <= 2^n for all n >=1".

To prove the latter by induction is easy: the base
case n=1 is trivial. Next assume for n=k (inducation
hypothesis) and prove for n=k+1.

```
 n+1 <= 2^n + 1  (by induction hypothesis)
     <= 2^n * 2  for any n  (obvious)
      = 2^{n+1}.
```

MARKING SCHEME:

Definitions: 1 point (0.5 for each)

a) 1 point (assuming the definition is stated correctly, and
is used to provide the result)

b) 1 points (again, assuming the definition is stated correctly)

c) 2 points (0.5 points if only the base case was given, but
rest was confused)

---

4 a) 1
b) n (you need to shift)
c) 1
d) n
e) n
f) lg n
g) 1
h) n

MARKING SCHEME:

Because you could guess O(1), O(n), O(lg n) on each question,
we marked this question by taking subtracting the number of

incorrect answers (including blanks) from the number of questions (8). Thus if you got only 3 out of 8 correct (5 incorrect), then you received a grade of 0/5. If you got 6 out of 8 correct (i.e. 2 incorrect), then you received a grade of 3/5).

---

5. a) defgh (with d at front and h and back)

b)
```
a***
ab**
abc*
*bc*
*bcd
**cd
e*cd
efcd  (c front, f back)
ef*d
efgd  (d front, g back)
defgh (expand and shift with d at front)
```

MARKING SCHEME:

1 point for (a) and 2 points for (b), the latter being broken down into 1 point for correct add/delete and 1 point for correct expansion.

---

6. a) 2^{h+1} - 1

b) sum_{i=0}^h i 2^i

c)
```
NumNodes(node){
  if node == null
    return 0
  else{
    sum = 1
    for each child c of node   // does nothing if
       sum = sum + NumNodes(c)  // there are no children
    return sum
  }
}
```

```
d)  SumDepths(node,depth){
      if node == null
        return 0
      else{
        sum = depth
        for each child c of node
          sum = sum + SumDepths(c,depth+1)
        return sum
      }
```

MARKING SCHEME:

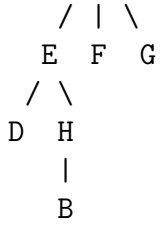
1 point for each. For (c),(d), no points were deducted if the null case was ignored.

7.)

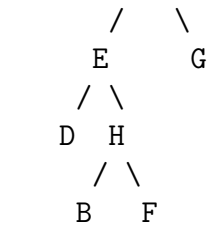
a)
```
A - B
B -
C - E,F,G
D -
E - D,H
F -
G - D
H - B,F
```

b) not shown

c) ordering is CEFGDHB



d)



ordering is CEDHBFG

e) There are many. Here is one: A C GE HD F B

f) Use the overlap method for the four (folding) top pieces of a packing box.