Faculty of Science
FINAL EXAMINATION

COMPUTER SCIENCE   COMP 250
INTRODUCTION TO COMPUTER SCIENCE

Examiner:	Prof. Michael Langer	April 27, 2010
Associate Examiner:	Mr. Joseph Vybihal	9 A.M. – 12 P.M.

Instructions:

The exam is 6 pages, including this cover page.

There are 9 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)**

   (a) Convert the decimal number 231 to binary.

   (b) What is the value of the hash function,

   $$h(i) = i \bmod 16$$

   for $i = 231$? You may express your answer in binary.

   (c) How many multiplications are used to compute 13^{231} using the recursive method discussed in class *i.e.* power(13,231) ?
   Note: The right side of "$13^{231} = 13^{115} \times 13^{115} \times 13$" is two multiplications.

2. **(2 points)**

   State the formal definition of "$f(n)$ is $O(g(n))$" and use it to prove $f(n)$ is $O(g(n))$ for the following:

   $$f(n) = (n + 237)^3 + 4n \log n + 3$$
   $$g(n) = n^3.$$

   Hint: You may assume $\log n \leq n$ for all n.

3. **(2 points)**

   (a) Show the final state of a *queue*, after the following sequence of operations is finished:

   ```
   add(a), add(b), add(c), add(d), remove(), add(e), add(f), remove(),
   remove(), add(g), add(h)
   ```

   Assume the queue is initially empty. Be sure to indicate the front of the queue.

   (b) Answer the same question as (a), but now use a *stack* instead of a queue. Here, indicate the top of the stack.

4. **(3 points)**

   State *without proof* the solution of the following recurrence relations.

   For (a)-(d), assume $t(0) = 1$. For (e),(f), assume $t(1) = 1$ and n is a power of 2.

   (a) $t(n) = 1 + t(n-1)$

   (b) $t(n) = c + t(n-1)$

   (c) $t(n) = n + t(n-1)$

   (d) $t(n) = c\ t(n-1)$

   (e) $t(n) = c + t(\frac{n}{2})$

   (f) $t(n) = n + 2\ t(\frac{n}{2})$

5. **(2 points)**

   Consider a mergesort algorithm that partitions a set of n comparable elements into three equal-size subsets, sorts each of them, and then merges the three sorted subsets. You may assume n is a power of 3.

   (a) Give a recurrence relation that describes the number of operations used for this algorithm as a function of n.

   (b) Solve this recurrence relation. Be sure to show the steps used to obtain your solution.

6. **(3 points)**

   Consider the following array of characters:

   $$\texttt{minHeap} = [\texttt{b}, \texttt{c}, \texttt{m}, \texttt{h}, \texttt{d}, \texttt{p}, \texttt{w}]$$

   which defines a min-heap.

   (a) What is the result of applying the heap's `removeMin()` operation? Your answer must be a new array with six elements.

   (b) Give a binary search tree of height 2 that contains the original characters of `minHeap`, *i.e.* before the operation in (a).

   (c) Apply the binary search tree operation `remove(h)` to your tree in (b).

7. **(5 points)**

   (a) Using pseudocode, write a *recursive* algorithm `findRoot(node)`, whose input is a node in a tree and whose output is the root node of the tree. For example, in the trees below, the input could be any node and the output would always be node '`a`'.

   Assume that each node in the tree has a field `parent`, and that the input node is valid, *i.e.* it references a node in the tree.

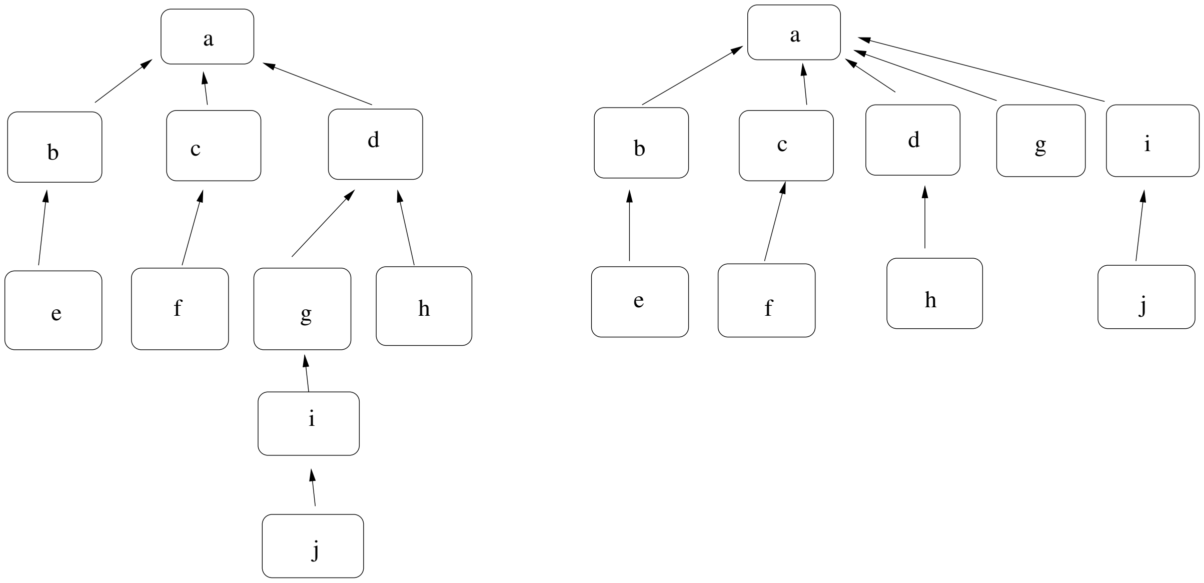
   (b) Consider an algorithm that transforms a tree such that all non-root nodes in the path from the input `node` to the tree `root` will have depth 1 in the transformed tree. The algorithm is called using:

   ```
   flatten(node, root) .
   ```

   An example of the transformation is shown below (left to right). For this example, what is the input `node` that explains the transformation?

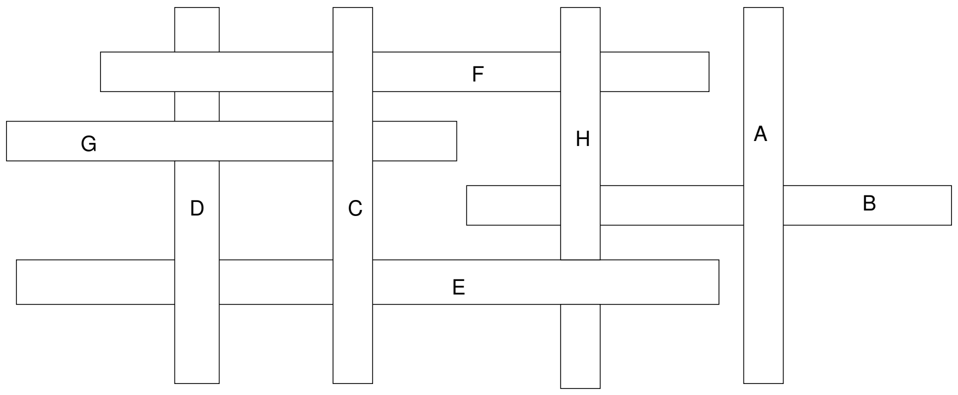
   (c) Using pseudocode, write a *recursive* algorithm `flatten( ..  )`. Assume that the inputs `node` and `root` are valid.

   (d) What might be the advantage of calling `flatten` after you call `findRoot`?



8. **(3 points)**

The figure below shows a set of rectangles whose overlap relationships can be represented using a directed graph. Each rectangle is represented in the graph by one vertex, and whenever one rectangle overlaps another rectangle, there is a directed edge. For the example below, the graph would contain the edges $\{(\texttt{C}, \texttt{G}),\ (\texttt{C}, \texttt{E}), \ldots, etc\}$.



(a) Give the adjacency list for this graph, such that *the vertices are ordered alphabetically.*

(b) Give the ordering of vertices visited in a *breadth first* traversal of the graph, starting from vertex `C`.

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

(c) Give the ordering of vertices visited in a *depth first* traversal, again starting from `C`.

9. **(7 points)**

   (a) What is the difference between *overriding* and *overloading* ?

   (b) When you write an `equals()` and `hashCode()` method for a class, what (if any) are the required relationships between the outputs of these method?

Next, consider the Java code below. This code does not do anything meaningful. Rather, it is used to test your understanding of inheritance and polymorphism.

Which (if any) of the instructions `(c)-(g)` of `Test` generate compiler errors?

What is the output of the program after removing those lines (if any) that cause compilation errors? *Be sure to mark which instructions produce which output.*

```
public class A {
   public  int   n  = 3;

   A( ){ System.out.println("A"); }           //  constructor
   public void  foo() { System.out.println( n ); }
}

public class B extends A{
   public  int     n;

   B() { }                                     //  constructor
   B(int m){                                   //  constructor
      System.out.println( "B" );
      this.n = m;
   }
   int  foo(int n) {
           System.out.println(this.n) ;
           return n;
   }
}

public class Test {

    public static void main(String[] args) {
       int n = 2;
       A   a = new A();                         //          (c)
       a.foo();                                 //          (d)
       a.foo( 7 );                              //          (e)
       B   b  =  new B( 11 );                   //          (f)
       n = b.foo( 4 );                          //          (g)
    }
}
```

Solutions

1. (a) 11100111

   (b) 0111 (or 7).

   (c) This one was not an easy question.

$$\begin{aligned}13^{231} &= 13^{115} \cdot 13^{115} \cdot 13 \\ 13^{115} &= 13^{57} \cdot 13^{57} \cdot 13 \\ 13^{57} &= 13^{28} \cdot 13^{28} \cdot 13 \\ 13^{28} &= 13^{14} \cdot 13^{14} \\ 13^{14} &= 13^{7} \cdot 13^{7} \\ 13^{14} &= 13^{7} \cdot 13^{7} \\ 13^{7} &= 13^{3} \cdot 13^{3} \\ 13^{3} &= 13 \cdot 13 \cdot 13\end{aligned}$$

   So, you need 13 multiplications (pure coincidence that the number happens to be 13).

   Marking scheme: 1 point for each of (a),(b),(c). If part (a) was wrong, and part (b) had the last 4 bits of your answer in part (a), then I gave the point. Part (c) was difficult, and we gave the point if you gave an answer close to 13 and explained your reasoning.

2. We want to show there exist $n_0 > 0$ and $c > 0$ such that, for all $n > n_0$,

$$(n + 237)^3 + 4n \log n + 3 < cn^3.$$

   We try to find an upper bound:

$$\begin{aligned}(n + 237)^3 + 4n \log n + 3 &< (n + 237n)^3 + 4n^3 + 3n^3, \quad \text{for all n} > 0 \\ &= (238^3 + 4 + 3)n^3\end{aligned}$$

   So take $c = 238^3 + 4 + 3$ and $n_0 = 1$.

   Marking scheme: 1 point for the definition. You needed to provide quantifiers ("for all..", "there exists..."). You lost 0.5 points out of 1 for each quantifier you left out.

   1 point for a correct proof in (b). There are other ways to do it. I provided the one I think is the simplest.

3. (a) d,e,f,g,h

   (b) a,b,c,g,h (with h on top)

   One point for each.

4. (a) $t(n) = n + 1$

   (b) $t(n) = nc + 1$

(c) $t(n) = \frac{n(n+1}{2} + 1$

(d) $t(n) = c^n$

(e) $t(n) = c \log n + 1$

(f) $t(n) = n \log n + n$

0.5 points for each, and we ignored the "1"'s. i.e. you got the 0.5 point even if you left them out.

5. (a) This is similar to 4(f), but now we split into 3 rather than 2.

$$t(n) = 3t(\frac{n}{3}) + n.$$

Each of the three sets of size $\frac{n}{3}$ needs to be sorted, and then the three (now) sorted sets need to be merged (n steps).

(b)

$$\begin{aligned}
t(n) &= n + 3t(\frac{n}{3}) \\
&= n + 3(\frac{n}{3} + 3t(\frac{n}{9})) \\
&= n + n + 9\ t(\frac{n}{9})) \\
&= 2n + 3^2\ t(\frac{n}{3^2})) \\
&= : \\
&= n \log_3 n + 3^{\log_3 n}\ t(\frac{n}{3^{\log_3 n}}) \\
&= n \log_3 n + nt(1) \\
&= n \log_3 n + n
\end{aligned}$$

6. (a) `c d m h w p`

(b)

```
         h
       /   \
      c     p
     / \   / \
    b   d m   w
```

(c) Either of the following are ok:

```
         m                          d
       /   \                      /   \
      c     p                    c     p
     / \     \                  /     / \
    b   d     w                b     m   w
```

7. (a)
```
findRoot(node){
   if (node.parent == null)
      return node
   else
      return findRoot(node.parent)
}
```
Many students left out the second `return` statement but no marks were taken off.

   (b) The input node is `i`.

   Some students interpreted the question incorrectly and forgot that the two end vertices of the path belong to the path. Such students answered `j`. From Assignment 4, it should have been very clear in your mind that a path from a node to the root includes both the node and the root.

   We were generous and gave 0.5 points instead of 1 point if you wrote `j`, provided that the algorithm you gave was correct for the (mis)interpretation.

   (c)
```
flatten(node,root){
   if (node != root) {
      flatten( node.parent, root)
      node.parent = root
   }
}
```
Many students used a temporary variable to point to *node.parent* (like when swapping two elements) but wrote Java commands like
`Node tmp = new Node();`
This is nonsense. First, the question asks for pseudocode, not Java code. Second, even if it did ask for Java code, there is no need to create new nodes. Such students typically lost a point for writing this. (And if you do this in COMP 251, you will be penalized more heavily.)

   This part was worth 2 points. (Note: some students who wrote `j` in (b) gave an algorithm here that would yield `j` at depth 1, which is inconsistent with the original question.

   (d) The point of this question was that the next time you want to find the root of the input `node`, you will only have to follow one reference, since the `node` is at depth 1.

   This question was interpreted in so many different ways that we threw out the question (and gave everyone the point).

Marking scheme: 1 point for (a,b,d). 2 points for c.

8. Some students misinterpreted "overlap" as a symmetric relation: `A` overlaps `B` is the same as `B` overlaps `A`. This would (incorrectly) give you an undirected graph. Note: a similar question was in the graph exercises, so students who got this wrong evidently did not do those exercises.

   Anyhow, we decided to only take 1 mark off for this (out of 3), since part of the question was to determine whether you know how to do a depth-first and breadth first search, and you can demonstrate this even if you misinterpreted the question as just described. Here I give solutions for the correct interpretation only.

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

(c) `C, E, H, B, F, D, G`

Marking scheme: 1 point for each of (a-c).

9. (a) Overriding means that a method in a child class has the same signature as a method in a parent class.

   Overloading is when multiple methods share the same name but have different signatures. (These could be in the same class or different classes).

   0.5 points for each. The key is to say in which ones the signatures are same or different.

   (b) If `a.equals(b)` is true, then `a.hashCode() == b.hashCode()` is true. (Equivalently, if `a.hashCode() == b.hashCode()` is false, then `a.equals(b)` is false.

   If `a.equals(b)` is false, then we cannot say anything about `a.hashCode() == b.hashCode()`, i.e. the latter could be true or false.

   See lecture 29, page 4.

The compilation error occurs in (e). The problem is that the `A` class has no `foo` method with an argument, and `a` is declared to be of class `A`.

```
Output:
A             from (c)
3             from (e)
A             from (f)    <---  many student missed this (and lost 0.5 points)
B             from (f)
11            from (g)
```

Marking scheme: 7 points in total. 1 point for each of (a)-(g).