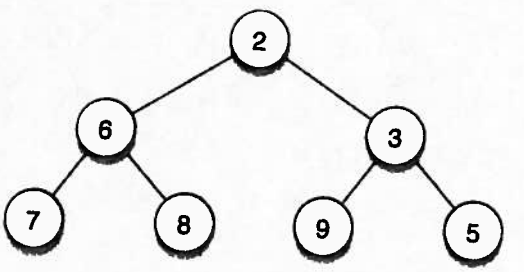
NAME: SOLUTION STUDENT ID: _____

COMP 250 – Midterm #2
March 12th 2012

- This exam has 6 pages
- This is an open book and open notes exam. No electronic equipment is allowed.

1) Tree traversal algorithms (20 points)

a) (10 points) What would the following algorithm print when executed on the root of the tree shown on the right? No justification is needed.




```
Algorithm WeirdTraversal(treeNode n)
if (n!= null) then {
        WeirdTraversal( n.getLeftChild() )
        Print n.getKey()
        WeirdTraversal( n.getRightChild() )
        if (n.getKey() is even) WeirdTraversal( n.getLeftChild() )
        else WeirdTraversal( n.getRightChild() )
}
```

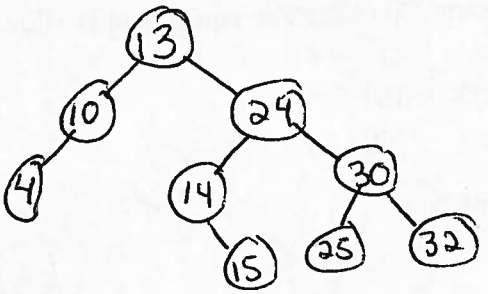
7 6 8 7 2 9 3 5 5 7 6 8 7

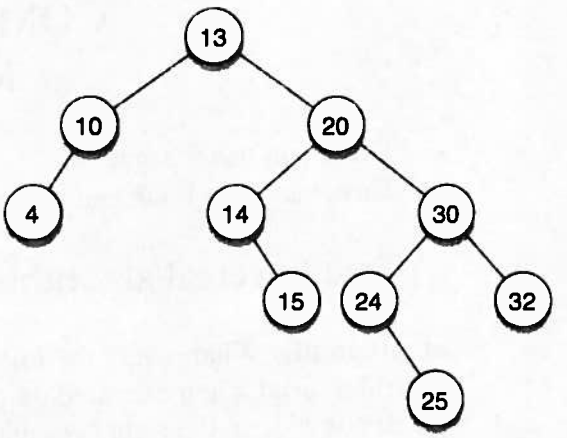
b) (10 points) Let $P(n)$ be the total number of times the "Print" statement is going to be executed when the WeirdTraversal algorithm is called on the root of a *full* heap of n nodes (a full heap is one where the last level is full, as shown above). Write a recurrence for $P(n)$. You do **not** need to solve the recurrence.

$$P(n) = \begin{cases} 3P\left(\frac{n-1}{2}\right) + 1 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

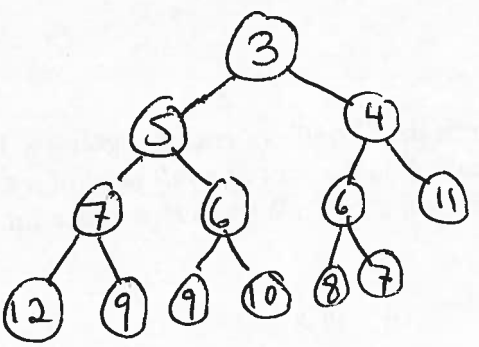
Question 2. Binary Search Trees and Heaps (16 points)

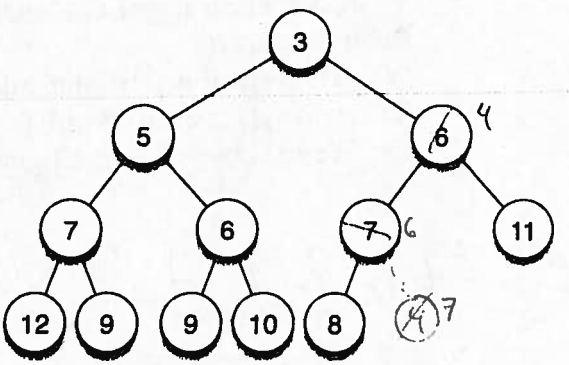
a) (8 points)
Consider the following Binary Search Tree.
Draw the Binary Search Tree after the remove(20) operation has been performed, as seen in class.





b) (8 points)
Consider the following heap.
Draw the Heap after the insert(4) operation has been performed, as seen in class.





3) Questions with short answers (28 points; 4 points each)

a) True or False? *Justify your answer briefly!* When designing a hash function for a hash table implementing a dictionary, one has to make sure that two different keys are never mapped to the same bucket.

False. It is ok for many keys to be mapped to the same bucket as each bucket is a dictionary that can store any number of elements.
However, for efficiency reasons, it is good to avoid having too many key in the same bucket.

b) Consider a sorted linked list containing n nodes, each storing an integer. On such a list, why is it not possible to do a binary search that runs in worst-case time $O(\log n)$?

Because we can't directly access the k-th element in time $O(1)$, as to find it we need to hop k times starting from the head. So simply finding the $\frac{n}{2}$-th element takes time $O(n)$ $\not\in$ $O(\log n)$

c) A ternary tree is a tree where each node has up to three children. What is the maximum number of nodes in a ternary tree of height h? *Justify your answer.*

$$\sum_{i=0}^{h} 3^h = \frac{3^{h+1}-1}{2}$$

d) Name the type of tree traversal algorithm that will print keys in increasing order when executed on a Binary Search Tree? No justifications are needed.

In-order traversal.

e) When using a linked list (not doubly-linked) to implement a queue ADT, is it better to implement the enqueue operation with the addLast() method and the dequeue operation with the removeFirst() method, or to implement enqueue with addFirst() and dequeue with removeLast()? Why?

Better is: enqueue: addLast
dequeue: removeFirst
because both run in time $O(1)$

whereas removeLast takes time $O(n)$

f) (2 points each) What Abstract Data Type would be the most appropriate to represent each of the following situations? No justifications are needed.

1) When a student is busy taking several courses, he works on his assignments in the order he receives them, completing first the assignment that was assigned first.

Queue

2) When another (wiser) student is busy taking several courses, she works on her assignments in the order of their due dates, completing first on the assignment that is due first.

Priority queue

3) When you go to the doctor, he/she can access your medical record by simply typing in your Health Insurance number.

Dictionnary

4) Mathieu receives a lot of e-mails. He always replies to the most recently received un-answered e-mail, deletes it, and then repeats the process.

Stack

Question 4. (18 Points)

Consider the Java implementation of a linked list given below. Implement in Java the method insertAfter, which takes as argument two integers called *marker* and *newElement*, and inserts immediately after the node with value *marker* a new node with value *newElement*. See below for an example. If no node with value *marker* exists in the list, then no new node is inserted.

Example:
Calling insertAfter(12,7) on the linked list: 5 → 2 → 6 → 12 → 9

Results in : 5 → 2 → 6 → 12 → 7 → 9

```
class node {
        public int value;
        public node next;
};

class linkedList {
        public node head;
        public node tail;

        public void insertAfter(int marker, int newElement) {
```

```
node cur = head;
while (cur != null && cur.value != marker) {
        cur = cur.next;
}
if (cur != null) {
    cur.setNext(new node(newElement, cur.getNext());
    if (cur == tail)  tail = cur.getNext();
}
```

```
        }
};
```

Question 5 (18 points)

Consider a *Binary Search Tree* that contains a set of distinct integer keys.

Problem: Write an algorithm that determines the number of keys between *Min* and *Max* (inclusively) contained in the tree, where *Min* and *Max* are two given integers.
For example, when executed on the root of the tree shown in Question 4 (a), nbKeysInRange(root, *Min*=12, *Max*=24) should return 5, because there are 5 keys with values between 12 and 24. *Your algorithm must avoid visiting parts of the trees that are not necessary*.

Algorithm nbKeysInRange(treeNode *n*, int *Min*, int *Max*)
Input: A treeNode *n* and two integers *Min* and *Max*
Output: Number of nodes with keys in the range [*Min*..*Max*] in the subtree rooted at *n*.
/* WRITE YOUR PSEUDOCODE HERE */

```
if(n=null) return 0;
count ← 0;
if( n.value ≥ Min && n.value ≤ Max) count ← count+1
if(n.value ≥ Min) count = count + nbKeysInRange(n.getLeftChild(), Min, Max)
if (n.value ≤ Max) count = count + nbKeysInRange (n.getRightChild(), Min, Max)
return count
```