NAME: Solution STUDENT ID: _____

COMP 250 – Midterm #2 - 1
March 11th 2013

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

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

a) (No justification needed) If a dictionary containing N keys it implemented using a hash table with K buckets, where each bucket is implemented using a linked-list, what is the

i) best-case running time?

O(1), if the element sought is the first in its bucket

ii) worst-case running time?

O(N), if all elements are in the same bucket and the one we are looking for is the last one of its 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)?

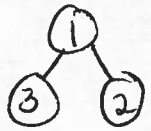
Because with a linked list, we can't access the middle element in constant time.

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.*

$\#nodes = 1+3+9+27+\ldots = \sum_{i=0}^{h} 3^i = \frac{3^{h+1}-1}{3-1} = \frac{3^{h+1}-1}{2}$

d) True or False? Justify your answer. A pre-order traversal executed on a heap will print the keys in increasing order.

False. On the following heap



pre-order traversal prints 1, 3, 2.

e) When using a singly-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() → O(1)
dequeue : removeFirst() → O(1)

The other way around would lead to O(n) running time for dequeue
enqueue: addFirst() → O(1)
dequeue: removeLast() → 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 2. (20 Points)

Consider the Java implementation of a linked list given below. Implement in Java the method removeElements(int *marker*), which remove <u>all</u> nodes with value equal to *marker* from the link list, leaving the rest intact. If no node with value *marker* exists in the list, then the list is left unchanged. For example:
calling removeElements(12) on the linked list: 5 → 12 → 6 → 3 → 12 → 12 → 9
results in : 5 → 6 → 3 → 9

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

class linkedList {
        public node head;
        public node tail;

        public void removeElements(int marker) {

                while (head.value == marker) head = head.next;
                node cur = head;
                while (cur != null) {
                        node n = cur.next;
                        while (n != null && n.value == marker) n = n.next;
                        cur.next = n;
                        if (cur.next == null) tail = cur.next;
                        cur = cur.next;
                }


        }
};
```

Question 3 (20 points)

Let T be a Binary Search Tree that contains a set of keys with *distinct* integers. Assume that you have a method subtreeSize(treeNode n) that returns the number of nodes in the subtree rooted at n, including n itself. Assume at any call to subtreeSize(treeNode n) takes time O(1).

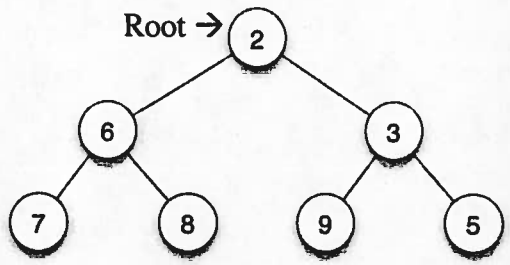
Problem: Write an algorithm that computes the number of nodes with a key greater or equal to a given integer k. Your algorithm should run in worst-case time O(h), where h is the height of the binary search tree (but you don't need to prove it).

Algorithm nbGreaterEqual(treeNode n, int k)
Input: A treeNode n and an integer k
Output: The number of nodes with key greater or equal to k in the subtree rooted at n.
/* WRITE YOUR PSEUDOCODE HERE */

```
if (n≠null) then
    if(n.getValue() < k) return nbGreaterEqual(n.getRightChild(), k);
    else return 1 + nbGreaterEqual(n.getLeftChild(), k)
                  + subtreeSize(n.getRightChild(), k)
else return 0;
```

4) Tree traversal algorithms (12 points)

Consider the following binary tree traversal algorithm.




```
Algorithm WeirdTraversal(treeNode n, int depth)
if (n!= null) then {
        if (depth is even) then {
                WeirdTraversal( n.getLeftChild() , depth+1 )
                WeirdTraversal( n.getRightChild() , depth+1 )
                Print n.getKey()
        }
        else {
                Print n.getKey()
                WeirdTraversal( n.getRightChild(), depth+1 )
                WeirdTraversal( n.getLeftChild(), depth+1 )
        }
}
```

Note: Not all versions of the exams had this version of the question

What would be printed when executing WeirdTraversal(root, 0)? No justification is needed.