Department of Electrical and Computer Engineering
University of Puerto Rico
Mayagüez Campus

ICOM 4035 – Data Structures
Spring 2012
Midterm Exam # 3

Name: _____

Student Number: _____

Section: _____

Instructions:

1. Write your name on all pages of this exam now!

2. You have two hours to complete this exam. Use your time wisely. Do not spend too much time on a problem, when you can work on others.

3. There are 3 problems for a maximum score of 70 points. Complete as many problems as you can, and earn as many points as possible

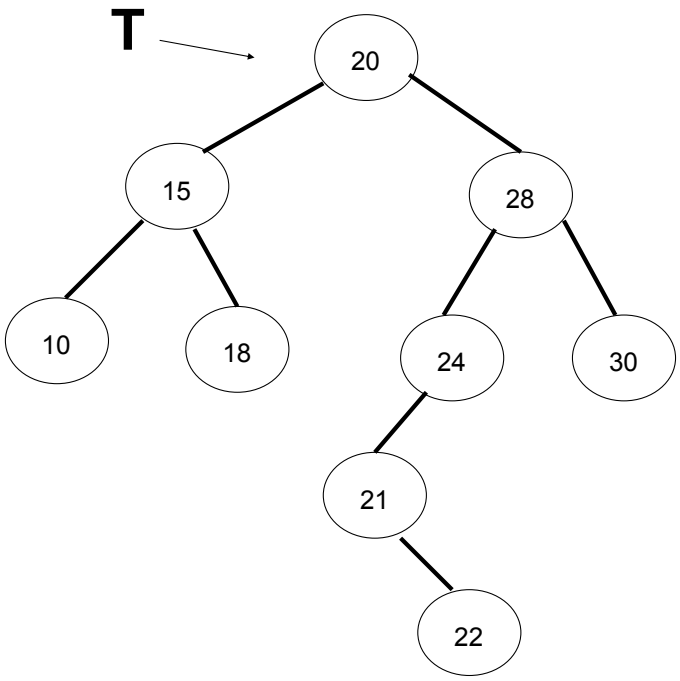
GOOD LUCK!

Scores

1	/20
2	/25
3	/25
Total	/70

Problem 1. (20 points) General Course Questions

Use the following Binary Tree T, storing integers, to answer the following questions:



a) (5 pts) What is the height of the tree T?

b) (5 pts) What is the depth of node 28 in tree T?

Problem 1. (Continuation)

c) (5 pts) Write a listing of the nodes in the tree T as visited in **post-order**.

For this problem, indicate the complexity (Big-O bound) for the function or code fragment. JUSTIFY YOUR ANSWER.

d) (**5 pts**)

```
// Print the contents of a hashtable that uses open addressing
public void print(PrintStream out){
     for (int i=0; i < this.buckets.length; ++i){
          out.print(i + ": ");
          if (this.buckets[i] != null){
               out.print(this.buckets[i]);
          }
          out.println();
     }

}
```

Problem 2. (25 pts) Understanding and Using Hash tables

Use the material discussed in class about doubly linked lists and hash tables to answer the following questions:

a) (**5 pts**) Consider a hash table for storing integers that has 10 buckets and uses separate chaining. Suppose that the table uses a hash function h(x) = x % N, where N is the number of buckets. Suppose that the hash code for an integer k is the integer k itself. Draw the resulting hash table after adding the following numbers to the hash table: 21, 9, 11, 17, 22, 3, 78, 94, 25, 1, 56.

Problem 2 (Continuation)

b) (10 pts) Consider a hash table that uses separate chaining. Write a **member** method named `bucketPals()`, which returns an array of Objects with all the elements currently located in the same bucket as an element `obj`. If the element `obj` is not found in the hash table, the method returns `null`.

```
public Object[] bucketPals (E obj){
```

Problem 2 (Continuation)

c) (**10 pts**) Consider the hash table implemented with open addressing. Remember that a collision will cause an element `obj` to be stored in a different bucket from the one that the hash function indicates. The bucket that the hash function indicates is called the "proper bucket". Write a **member** method `collisionDistance`() which returns the number of positions away from its proper bucket for a given value `obj`. The function returns -1 if `obj` is not in the hash table, or 0 if the element `obj` is in the bucket that the hash function specifies.

```
public int collisionDistance (E obj){
```

Problem 3. (25 pts) Understanding and Implementing Trees

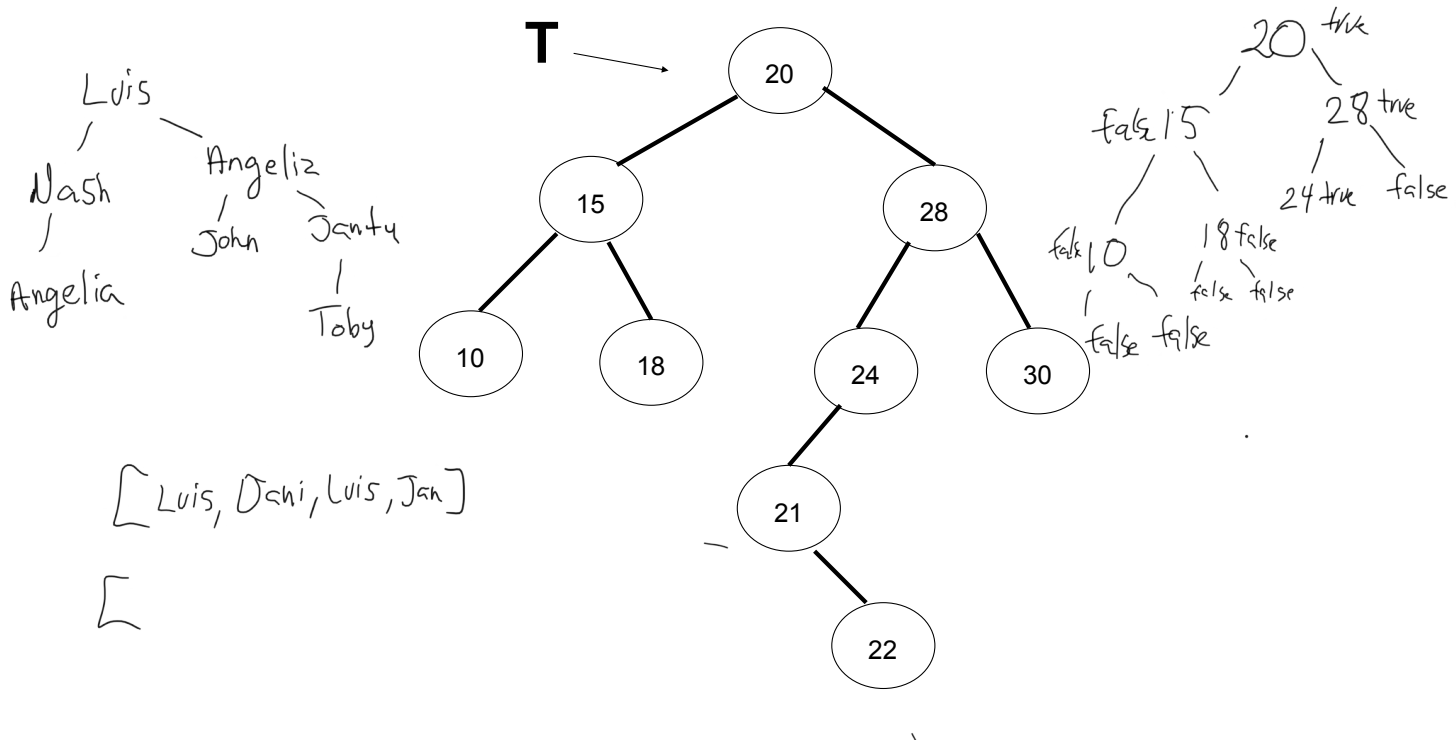


Figure 1: Example Binary Tree

Extend the functionality of the `BinaryTreeImp` by completing the following tasks:

a) **(5 pts)** Implement a **member** method named `getPath()` that returns a SinglyLinkedList with all the values located in the path from the root of a tree to the a value E. If the value is not found, the function returns `null`. For example, in the tree T of figure 1, a call to `T.getPath(24)`, returns L = {20, 28, 24}. Likewise, a call to `T.getPath(20)` returns L = {20}. ***Hint: You need to use a recursive auxiliary function*.** Provide your answer on the next page:

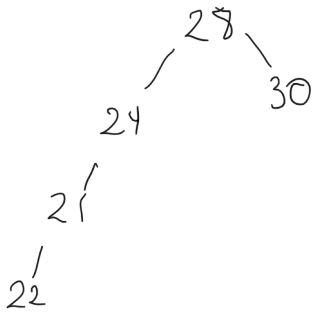
Problem 3. (Continuation)

```
public List<E> getPath(E obj){
```

Problem 3. (Continuation)

b) (**10 pts**) Write a **member** method named `descendants()` which returns a SinglyLinkedList with all the descendants of a given node, including itself. If the value is not found, it returns null. For example, in the tree in Figure 1, a call to `T.descendants(28)` returns L = {28, 24, 21, 22, 30}. ***Hint: You need to use a recursive auxiliary function***

```
public List<E> descendants(E obj){
```



Problem 3. (Continuation)

c) (**10 pts**) Write a **member** method named `internalNodes()`, which returns a SinglyLinkedList with all the internal nodes in a binary tree. Remember that an internal node is a node that has at lest one child. For example, in tree T of figure 1, a call to `T.internalNodes()` returns L = {20, 15, 28, 24, 21}. The function returns an **empty list** if there are no internal nodes. ***Hint: You need to use a recursive auxiliary function***

```
public List<E> internalNodes(){
```