This quiz is closed book and only one notes sheet. There are 75 points in ten questions and you must show your work to receive full credit. Put your name, the date, and “Exam 2” on each extra sheet of paper you turn in. Papers must be stapled.

1. Give and briefly explain an example of a solution that uses recursion. (5 points)
   ANSWER: There are many possible correct answers such as The Tower of Hanoi, depth-first search of a graph, and others.

2. Define: (5 points each)
   (a) Tree
      ANSWER: A tree is a simple, connected, acyclic graph. A tree is a connected graph with the fewest possible number of edges.
   (b) Interior node
      ANSWER: Any node in a tree with more than zero children nodes is an interior node.
   (c) Binary tree
      ANSWER: Any tree where each node has no more than two children nodes or trees.

3. For a given tree, what characteristic of the tree gives the maximum number of steps required to find any particular node? (5 points)
   ANSWER: Semi-trick question. If the tree has no known structure, the number of nodes in the tree is the number which must be searched. In a BST this can be reduced because of the known structure of the BST.

4. Given 5 unique items to store in a binary tree:
   (a) What is the worst case depth of the tree? (2 points)
      ANSWER: 5
   (b) What is the best case depth of the tree? (2 points)
      ANSWER: 3
   (c) With a plain BST and input in sorted order, what is the expected depth of the tree? (1 point)
      ANSWER: 5 because sorted input will produces a tree that leans completely to either the left or right depending upon the ascending or descending order of the input.

5. What is the expected time complexity for a merge sort? (5 points)
   ANSWER: $O(n \log_2 n)$
6. Write the pseudo code to do a breadth-first traversal of a binary tree. (10 points)
Answer:

**Algorithm 1** BreadthFirstTraversal(G)

1: **procedure** BREADTHFIRSTTRAVERSAL((G))
2: Create an empty queue of binary trees named Q
3:  Q.enqueue G.root
4:  **while** Q.isNotEmpt do
5:      u = Q.dequeue
6:      visit u
7:      if u.leftChild.notEmpty() then
8:         Q.enqueue u.leftChild
9:      end if
10:     if u.rightChild.notEmpty() then
11:        Q.enqueue u.rightChild
12:    end if
13:  end while
14: **end procedure**

7. Write the pseudo code to do an pre-order walk of a binary tree. (10 points)
   Answer:

**Algorithm 2** PreOrderTraversal(G, root)

1: **procedure** PREORDERTRAVERSAL((Tree G, Node root))
2:     PreOrderTraversalHelper(G, root)
3: **end procedure**
4: 
5: **procedure** PREORDERTRAVERSALHELPER((Tree G, Node target))
6:     visit target
7:     PreOrderTraversalHelper(TreeG, Nodetarget.leftSubTree)
8:     PreOrderTraversalHelper(TreeG, Nodetarget.rightSubTree)
9: **end procedure**

8. Give an example, other than searching, of a use for a depth-first traversal of a binary tree. (5 points)
   Answer: Any of the NodeVisitors from the binary tree lab. Examples could include finding the extreme values, printing the tree in pre-order, printing the tree in-order, printing the tree in post-order, sorting, etc.

9. In class, we discussed a Red-Black tree to produce a balanced BST:
   (a) Does it always produce a perfectly balanced tree? (5 points)
       Answer: Typically, it does not produce a perfectly balanced tree, but the depth of the tree is no more than twice the best possible depth.
   (b) What did the structure add to each node to aid in the balancing? (HINT: The name of the structure gives you the answer. 5 points)
Answer: Red-Black trees add a “color” to each node and uses a set of rules about these colors to attempt to balance the tree. The legend is that the authors of the original paper only had two colors of pens to play with: red and black.

10. What would you expect for the time complexity of a pre-order walk of a BST with $n$ nodes? (5 points)

Answer: Because you must visit each node exactly once, you would expect a time complexity of $O(n)$. 