Quiz 1 Practice Questions

1. The input is a set S containing n real numbers, and a real number x.
a. Design an algorithm to determine whether there are two elements of S whose sum is exactly x. The algorithm should run in O(nlogn) time.
b. Suppose now that the set S is given in a sorted order. Design an algorithm to solve the above problem in time O(n).

2. You are given 9 identical looking balls and are told that one of them weighs a bit less than the rest of the eight balls. The only operation you are allowed is to compare a set of balls against another set of balls. Determine the lighter ball using 3 comparisons. Generalize your answer to more that 9 balls if possible.

3. Harder) You are given 12 balls, and are told that one of them is of a different weight from the rest – i.e., you don't know if it is heavier or lighter. Determine this ball using 4 comparisons.

4. Solve the following recurrence equations:
a. $T(n) = 2T(n/2) + n\log_2 n$, $T(2) = 4$
b. $T(n) = 3T(n/2) + n \log_2 n$, $T(1) = 1$
c. $T(n) = T(9n/10) + n$
d. $T(n) = T(n - 1) + \log_2 n$

5. Compare the following functions in terms of orders. In each case, say whether $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, and/or $f(n) = \theta(g(n))$.
1. $f(n) = \sqrt{n}$, $g(n) = (\log n)^5$
2. $f(n) = n^2/\log n$, $g(n) = n(\log n)^2$
3. $f(n) = \log n$, $g(n) = \log(n^2)$
4. $f(n) = n2^n$, $g(n) = 3^n$

6. The majority of a set of numbers is defined as a number that repeats at least n/2 times in the set. Design a linear time algorithm to find the majority, if one exists.

7. Show exactly why if we grouped elements into groups of 3 each, the median finding algorithm that we discussed in class will not work in linear time. What would be the running time of the algorithm in this case?

8. Which is better in the median-finding algorithm, grouping into groups of 5, or into groups of 7? Explain your answer.

9. Let X and Y be two arrays of n numbers each, both already sorted. Give a O(logn) algorithm to compute the median of XUY.

10. Given any node u in a binary tree, let the number of nodes of the subtree tree rooted at u be nodes(u), the number of nodes of the left subtree at u be leftnodes(u), and the number of nodes of the right subtree be righnodes(u). Clearly we have nodes(u) = leftnodes(u) + rightnodes(u) + 1.

We now introduce a new definition of approximately balanced trees. A tree is approximately balanced if leftnodes(u) <= 2*rightnodes(u), for all nodes u in the tree. Given a tree with n nodes, determine an upper bound for height of the tree, i.e. determine if in the worst case the height = O(n) or O(√n), or O(logn), etc.
Hint: use recurrence relations to express the height.

11. Suppose you are given a completely balanced binary search tree (a completely balanced tree means that each path from root to leaf is exactly the same). What is the time required to find the median of the elements of such a tree?

12. Can you use a binary search tree to simulate heap operations? What are the advantages/disadvantages of doing so?

13. You are given n positive integers. Can you find the product of k minimum numbers in the array? (Hint: use **max-heap**, A **max-heap** is a complete binary tree in which the value in each internal node is greater than or equal to the values in the children of that node.)

14. Given a heap and a number k, design an efficient algorithm that outputs the top-k largest element from the heap. What is the running time of the algorithm? Can you design an algorithm that runs faster than O(klogn)?