Q1. Formally define when you can say *f(n)* =Theta*(g(n))*. Put the following functions in order from lowest to highest in terms of their Theta classes. (Some of the functions may be in the same class. Indicate that in your list as well.) [1 + 3]

(a) $f_1(n) = nlogn$

(b) $f_2(n) = n^{3/2}$

(c) $f_3(n) = 1000$

(d) $f_4(n) = \sqrt{n}(n + logn)$

(e) $f_5(n) = 3^n$

(f) $f_6(n) = 2^{n+2}$

(g) $f_7(n) = 0.00001$

Solution: Theta notation: standard definition:

$$\{f_7, f_3\} < f_1 < \{f_2, f_4\} < f_6 < f_5$$

Q2. Suppose that you are given *n* red and *n* blue water jugs, all of different shapes and sizes. All red jugs hold different amounts of water, as do the blue ones. Moreover, for every red jug, there is a blue jug that holds the same amount of water, and vice Versa. Your task is to find a grouping of the jugs into pairs of red and blue jugs that hold the same amount of water. To do so, you may perform the following operation: pick a pair of jugs in which one is red and one is blue, fill the red jug with water, and then pour the water into the blue jug. This operation will tell you whether the red or the blue jug can hold more water, or that they have the same volume. Assume that such a comparison takes one time unit. While a brute force algorithm would require $O(n^2)$ comparisons, your goal is to use a divide and conquer algorithm that reduces the number of comparisons to determine the grouping. Remember that you may not directly compare two red jugs or two blue jugs. Also, comment

on the complexity of your algorithm.
[6]

Basic Idea: Choose a jug r randomly from R (set of red jugs), and compare it with every jug of B (set of blue jugs). This will help you partition set B around pivot r (as there is exactly one jug matching this size), others would be smaller or larger. Now, partition the set R by taking jug 'b' having the same size as 'r'. Recursively call this for the partitions.

Complexity: same as Quick Sort.

Pseudo Code below (Not required)

```
MATCH-JUGS(R, B)
if |R| = 0                          ▷ Sets are empty
  then return
if |R| = 1                          ▷ Sets contain just one jug each
  then let R = {r} and B = {b}
       output "(r, b)"
       return
  else r ← a randomly chosen jug in R
       compare r to every jug of B
       B_< ← the set of jugs in B that are smaller than r
       B_> ← the set of jugs in B that are larger than r
       b ← the one jug in B with the same size as r
       compare b to every jug of R − {r}
       R_< ← the set of jugs in R that are smaller than b
       R_> ← the set of jugs in R that are larger than b
       output "(r, b)"
       MATCH-JUGS(R_<, B_<)
       MATCH-JUGS(R_>, B_>)
```

Question 3: **[3]**
Preorder and in order of a tree is give

Preorder ----- A B D H E C F I G J K
Inorder -------- D H B E A I F C J G K

What is the postorder?

H D E B I F J K G C A

Question 4: [2]

T is a binary tree of height 3. What is the largest number of nodes that T can have? What is the smallest number?

If every internal node of T has 2 children then there is one node at the root, 2 at depth 1, 4 at depth 2, 8 at depth 3, for a total of 15 nodes. If every internal node of T has 1 child, then there is a total of 4 nodes.

Question 5: [5]

If $E(G)$ denotes the edges in a graph G, $E(T)$ denotes the edges in the BFS tree T of G. The set $E_n = E(G) \backslash E(T)$ denotes the set of non-tree edges, i.e., the edges which are in the graph G but not in the tree T.

Cross edge: A non-tree (missing) edge, {u, v} $\in E(G) \backslash E(T)$ is said to be a cross edge if $u \in L_i$ and $v \in L_i$ (i.e., both the vertices are in same level). Let Ec denote the set of all cross edges in T.

Slant edge: Slanting edge if $u \in L_i$ and v $\in L_j$, j = i +1 or j = i - 1 (i.e., both the vertices are in adjacent levels). Let E_s denotes the set of all cross edges in T.

Given a graph G, give an algorithm to find if G contains an odd cycle, a cycle of length $2k + 1$, $k \geq 1$, or not? (Hint: Use the edges classes defined above).

Solution: The existence of cross edges in T (Ec <> ∅) implies the existence of odd cycles in G. Let e = {u, v} be a cross edge in T and let x be the common parent of u and v. It is clear that, the length of Pxu (Path from x to u in T) is equal to the length of Pxv. Thus, Pxu and Pxv forms an odd cycle together with e.