

Part I (Written component):Due: **Friday, September 8 at 2:15 pm.**

Submission: Please hand in a paper copy of Part I at the beginning of class.

1. [4 points]:

Prove that $\log_2(n+2) \in \Theta(\log_2(n))$.

2. [8 points]:

Consider the following algorithm that accepts an $n \times n$ matrix M and returns the index of the first row of M that has a median value greater than 0 if one exists, or -1 otherwise.

```

FindPositiveMedian(M, n) {
    i=0
    do
        M[i] = SelectionSort(M[i])    //sort values in row i of matrix M
        median = M[i, n/2]
        i = i + 1
    while (median <= 0 && i < n)

    if (median <= 0)
        return i-1
    else
        return -1
}

```

Assume that the worst-case running time $S(n)$ of SelectionSort is $\Theta(n^2)$.Let $T(n)$ be the worst-case running time of the algorithm.

- (i) Give a function $f(n)$ such that $T(n) = O(f(n))$ and prove this result.
- (ii) Give a function $g(n)$ such that $T(n) = \Omega(g(n))$ and prove this result.
- (iii) Is there a function $h(n)$ such that $T(n) = \Theta(h(n))$? If so, give $h(n)$ and prove this result. If not, explain why such a function does not exist.

Note that you should give functions in parts (i) and (ii) which are good bounds for $T(n)$. For example, you should not say that $T(n) = O(n^{10})$ if it is also true that $T(n) = O(n^9)$.

3. [4 points]:

(i) Give an AVL Tree that contains the keys $\{2,4,6,8,10,12,14\}$ and has the minimum possible height of any AVL Tree that contains these keys.(ii) Give an AVL Tree that contains the keys $\{2,4,6,8,10,12,14\}$ (and no others) and has the maximum possible height of such a tree.

(iii) Give a sequence in which the keys can be inserted into an empty AVL Tree to produce the tree you gave in part (ii).

Part II (Programming component):Due: **Friday, September 15 at 2:15 pm.**

Submission: Please submit the file TwoFourTree.java, and any other files you may have written that are necessary to build and run your code, by sending them as email attachments to **submit301@cs.grinnell.edu**. Your submission should arrive by the due date and time. *Be aware that email sent from within the MathLAN*

network will generally arrive within a minute or so, but email from hotmail, gmail, and similar accounts can sometimes be delayed significantly prior to delivery. Your code should compile and run on MathLAN.

4. [24 points]:

Write a public class named `TwoFourTree` that implements a (2,4)-Tree data structure according to the following specifications. Your code should be written with good style (e.g., it should be well-organized, easy to read, and well-documented). Your tree should store integer keys, and it should include the methods described below.

```
public boolean contains(int k)
```

This method returns *true* if key `k` is in the tree. Otherwise, it returns *false*.

```
public void insert(int k)
```

This method inserts key `k` into the appropriate location in the tree. If the tree already contains key `k`, it does nothing.

```
public void delete(int k)
```

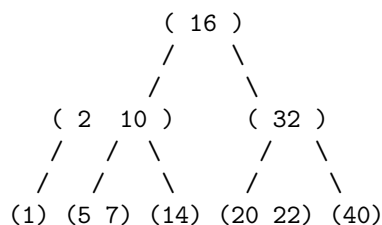
This method removes key `k` from the tree. If the tree does not contain `k`, it does nothing.

```
public void printTree()
```

This method prints a visual representation of the tree given by a pre-order traversal. An example is given below. (Note that this method will come in quite handy as you test your code.)

A pre-order traversal of a tree is one in which, for each node x in the tree, x is visited (and in this case printed), and then the children of x are visited from left to right. *If you are not familiar with pre-order traversals, please ask me about them. The problem is not meant for you to have to puzzle over this definition.*

As an example, the following AVL tree



should produce output that looks approximately as follows:

```

16
 2 10
  1
  5 7
  14
 32
 20 22
 40

```

HINT:

Your implementation may include other methods in addition to those described above. For example, you may find it helpful to implement recursive methods

```
private AVLNode insert(int k, AVLNode node), and
```

```
private AVLNode delete(int k, AVLNode node)
```

that insert key `k` into (or delete key `k` from) the subtree rooted at `node` and return a reference to the node at the root of the modified subtree. (See the reading by Weiss for a BST example that uses this idea.)