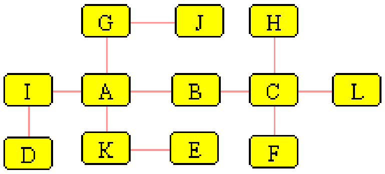
CMSC 350 Homework 3

1. (15 points) Write a static recursive Java method that will accept an int and write the base 2 (binary) representation of that int, one bit per recursive call, to System.out.

2. (15 points) Write a static recursive Java method that will accept two int's (A and B) and will write the representation of A base B, one character per call, to System.out. Assume that B is less than 36, and use digits 0 through 9, followed by characters A through Z to represent the characters of B.
   For example, 15 base 10 would be printed as F base 16.

3. (10 points) For the following tree, what is the order if the root is taken to be node A and the tree is traversed
   a) in pre-order
   b) in post-order
   c) in level-order



4. (10 points) For the tree in part 1, same questions but take the root as node L.

5. (10 points) In a 5-ary tree, that is a tree with at most 5 children at each node, how many nodes can a tree of height H have. Recall that the height of a tree is a length of the longest path from the root to the leaf nodes of the tree.

6. (10 points) In a 5-ary tree, what is the maximum height of a tree with N nodes. Since that was easy, let's try one a little harder: what is the minimum height of a tree with N nodes?

7. (30 points) Write a recursive method listLinks that will list all the links in a tree or subtree of a node.
   The context should be a Node (a class with a getChildren method that will return a List <Node> of the child nodes of that node) and use the toString method of the Node class. The output for each call of the listLinks method should be to System.out, with the first token the representation of the node followed by a space-separated list of the children of the node.
   Calling this method in the context of the root node of a tree will result in a listing of all the links in the tree. The call would be something like:
   ```
   tree.root.listLinks ();
   ```
   The signature of this method should be:
   ```
   void listLinks ()
   ```
   And you will use a method with the following signature:
   ```
   List <Node> getChildren ()
   ```