Nabeel Hussain

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.

Converting an integer into a binary representation by hand can be done through repeatedly dividing the number by 2, while keeping track of its remainder on the right side. The remainder will always either be a 0 or 1.

Once you have divided enough times and reached a quotient of zero, then starting with the bottom remainder, read the sequence of remainders upwards to the top. That will represent the integers binary representation form.

The recursive method to this operation is:

**public** **static** **void** binaryRepresentation(**int** number)

{

**if**(number == 0) // Base case

{

System.***out***.print("");

}

**if**(number > 0) // Recursive case

{

*binaryRepresentation*(number/2);

**int** binaryRemainder = number % 2;

System.***out***.print(binaryRemainder + "");

}

}

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.

**public** **static** **void** baseConversion(**int** a, **int** b)

{

**if**(a == 0) // Base case

{

System.***out***.print("");

}

**if**(a > 0) // Recursive case

{

*baseConversion*(a/b, b);

**int** number = a % b;

//if a % b is between 0 through 9, then print the digit.

**if**(number >= 0 && number <= 9)

{

System.***out***.print(number + "");

}

//Otherwise, use characters A through Z for any digit after 9.

**else**

{

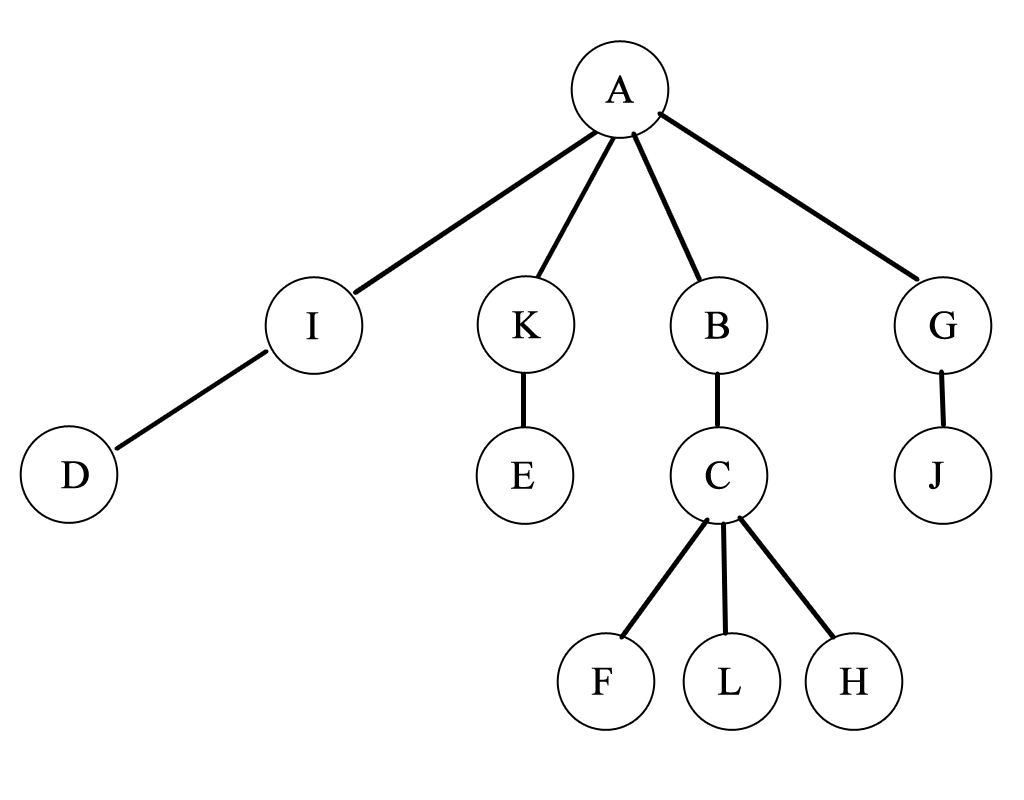
System.***out***.print(Character.*toChars*(65 + number-10));

}

}

}

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.



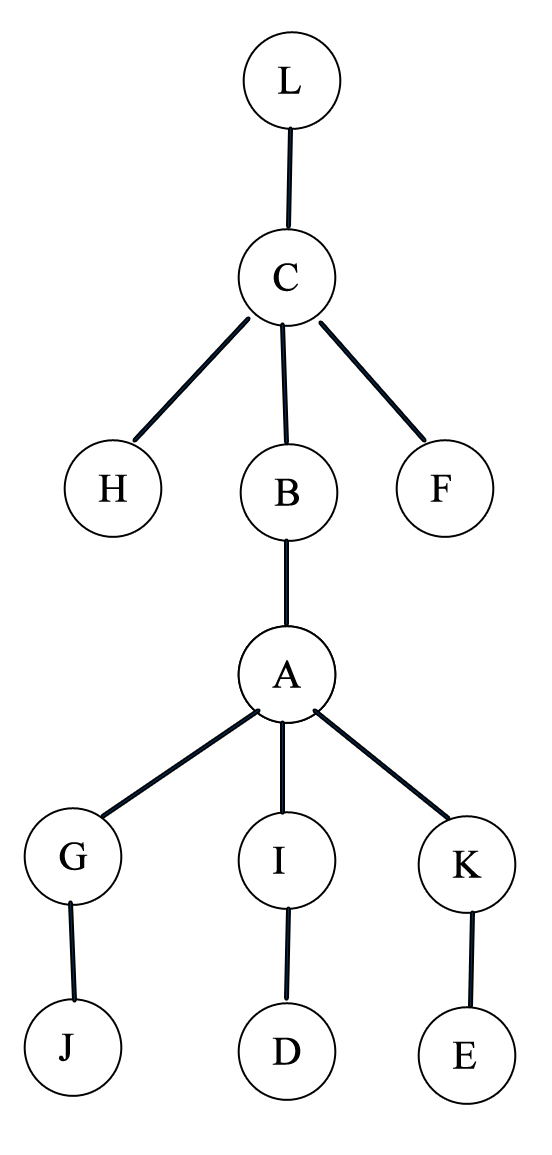


a) pre-order: **A, I, D, K, E, B, C, F, L, H, G, J**

b) post-order: **D, I, E, K, F, L, H, C, B, J, G, A**

c) level-order: **A, I, K, B, G, D, E, C, J, F, L, H**

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



a) pre-order: **L, C, H, B, A, G, J, I, D, K, E, F**

b) post-order: **H, J, G, D, I, E, K, A, B, F, C, L**

c) level-order: **L, C, H, B, F, A, G, I, K, J, D, E**

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.

A tree containing a single root node, will have height of 0.

Therefore,

50 = 1 node

If the root node has a child, then the tree will have a height of 1. The root node can have a max of 5 children.

Therefore,

51 = 5 nodes

Then, if each of those children have children of their own, then the tree will have a height of 2. Also, each of those children can have 5 children of their own.

Therefore,

52 = 25 nodes

If you keep following this geometric pattern, then for a tree of height H, the max number of nodes it can have is:

Total nodes = 50 + 51 + 52 + 53 + 54 +……+ 5H = 5H+1 - 1

Total nodes **=**

**Total nodes = =**

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?

**Maximum Height:**

If each node has only 1 child, then:

Root node = max height is 0

Root node with 1 child = max height is1

Root node with one grandchild = max height is 2

Root node with one great grandchild = max height is 3

Etc…

This will turn into a linked list, and as a result:

**The maximum height of a tree with N nodes is N-1.**

**Minimum Height:**

Given the number of nodes, each level must be completely full before the next level can be reached, in order to find the minimum height of the tree.

Nodes **=**

(N-1) \* (Nodes) =

(5-1) \* (Nodes) =

Log5(4 \* (Nodes)) = H + 1

Log5(4 \* (Nodes)) – 1 = H

**H = Log5(4) + Log5(Nodes) – 1**

Ex.

The height for a 5-ary tree containing 156 nodes is:

Proof:

H = Log5(4) + Log5(156) – 1

H = Log5(624) – 1

H = 3.999 – 1

H = 2.999 = 3 (approx.)

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 ()

**public** **void** listLink()

{

// The first token is the representation of the node

System.***out***.print(**this** + " ");

// The getChildren() method of the Node class that will return a List <Node> // of the child nodes of that node

List<Node> childNodeList = getChildren();

**for**( Node node: childNodeList)

{

//A space-separated list of the children of the node

System.***out***.print(node + " ");

node.listLinks();

}

}