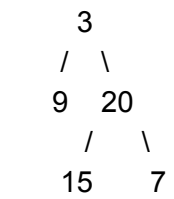
1. Average of Levels in Binary Tree

Given a non-empty binary tree, return the average value of the nodes on each level in the form of an array. Example:

Input:



Output: [3, 14.5, 11]

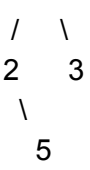
Explanation: The average value of nodes on level 0 is 3, on level 1 is 14.5, and on level 2 is 11. Hence return [3, 14.5, 11].

Note: The range of node's value is in the range of 32-bit signed integer.

2. Binary Tree Paths

Given a binary tree, return all root-to-leaf paths. Note: A leaf is a node with no children. Example:

Input:



Output: ["1->2->5", "1->3"]

Explanation: All root-to-leaf paths are: 1->2->5, 1->3

Implement and test your code in the BinaryTree.java as flowing:

```
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class BinaryTree {
      public Node root;
```

```
	public static class Node {
		public int key;
		public Node left, right;

		public Node(int item) {
			key = item;
			left = right = null;
		}
	}
	public List<String> binaryTreePaths(Node root) {
		// place your code here
		return null;
	}
	public List<String> binaryTreePaths() {
		return binaryTreePaths(root);
	}

	public List<Double> averageOfLevels(Node root) {
		// place your code here
		return null;
	}
	public List<Double> averageOfLevels() {
		return averageOfLevels(root);
	}
	public static void main(String args[]) {
		BinaryTree tree1 = new BinaryTree();
		tree1.root = new Node(3);
		tree1.root.left = new Node(9);
		tree1.root.right = new Node(20);
		tree1.root.right.left = new Node(15);
		tree1.root.right.right = new Node(7);

		System.out.println(tree1.binaryTreePaths());
		System.out.println(tree1.averageOfLevels());

	}
}
```

The expected output of the code is as follows:

```
[3->9, 3->20->15, 3->20->7]
[3.0, 14.5, 11.0]
```