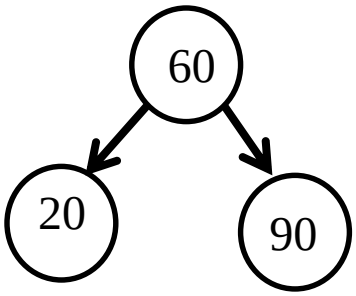
CS 314 Section Problem 8 - Trees - **There are 2 problems to solve**

1. Write an instance method for an `IntBST` class that determines the number of elements in the tree that are evenly divisible (no remainder) by a given integer. The `IntBST` only stores `int`s.



Examples based on tree to the right:

```
numDivisible(20) -> 2
numDivisible(-10) -> 3
numDivisible(1) -> 3
numDivisible(7) -> 0
```

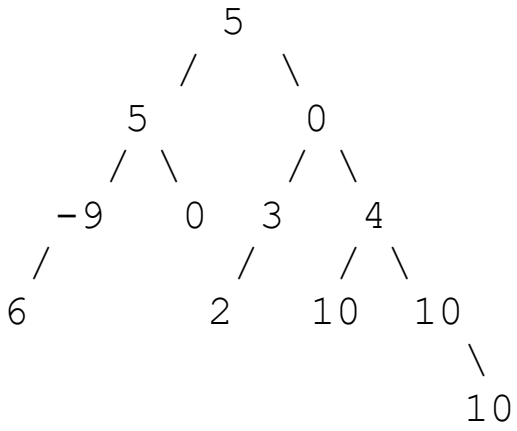
You may not use any other classes or methods except the the `BSTNode` class.
As always, you may add your own helper methods.

Your solution shall be as efficient as possible given the constraints.

```
public class IntBST {

      private BSTNode root; // root == null if size == 0
      private int size;

      // recall outer class can access private fields in nested class
      private static class BSTNode {
            private int val;
            private BSTNode left, right;
      }

      // pre: num != 0
      public int numDivisible(int num) {
```

2. Consider a binary tree that contains integers. The binary tree is ***not*** a binary search tree. Write a method that returns `true` if there is a non-empty path from the overall root of a tree to a descendant node in which the sum of the data stored in the nodes in the path equals a target value.
For this question the root is considered a descendent of itself. (A path can consist of just the root node.)

Consider the following tree and various target values



target of 10 -> true: root(5) -> left (5)
target of 5-> true: root(5)
target of 0 -> false
target of 27 -> false
target of 19 -> true: root(5) -> right(0) -> right(4)
-> left(10) = 5 + 0 + 4 + 10 = 19
target of 7 -> true: root(5) -> left(5) -> left(-9)
-> left(6) = 5 + 5 + (-9) + 6 = 7
target of 14 -> false

The path must start at the root and move to descendant nodes. The path cannot go back up the tree.

Use the following `BinaryNode` class:

```
public class BinaryNode {
      public int getData();
      public BinaryNode getLeft();
      public BinaryNode getRight();

      public void setData(int n);
      public void setLeft(BinaryNode left);
      public void setRight(BinaryNode right);
}
```

Use the following `BinaryTree` class:

```
public class BinaryTree {
      private BinaryNode root; // if tree is empty root == null
}
```

You may not use any other Java classes or methods other than the `BinaryNode` class.

You may not use any other methods from the `BinaryTree` class unless you implement them yourself as a part of your answer.