CS367 Homework 3

Lecture 1, Spring 2018

Due by 11:59 pm on Friday, March 23, 2018 (not accepted late)

In this page: Questions | Handing in Related pages: Assignments

Questions

Homework assignments must be done individually. Collaboration on homework assignments is not allowed.

1. Assume that **general trees** are implemented using a `Treenode` class that includes the following fields and methods:

```
// fields
private T data;
private List<Treenode<T>> children;

// methods
public T getData() { return data; }
public List<Treenode<T>> getChildren() { return children; }
```

   For efficiency, use an iterator to access the children in the list returned by the method `getChildren` (as we've done in lecture). You may assume that `getChildren` never returns `null`: if a node is a leaf, then `getChildren` will return a non-`null` list containing zero elements.

   Write an `isBinary` method whose header is given below.

```
public boolean isBinary( Treenode<T> n )
```

   The method should determine if the general tree rooted by n is also a binary tree. A binary tree is recursively defined as:

   - An empty tree is binary.
   - A leaf node is a binary tree.
   - A node with 1 or 2 children is binary if each child is itself a binary tree.

2. Assume that **binary trees** are implemented using a `BinaryTreenode` class that includes the following fields and methods:

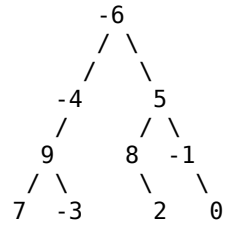
```
// fields
private T data;
private BinaryTreenode<T> left, right;

// methods
public T getData()                 { return data; }
public BinaryTreenode<T> getLeft()        { return left; }
public BinaryTreenode<T> getRight()       { return right; }
public void setLeft(BinaryTreenode<T> newL)  { left  = newL; }
public void setRight(BinaryTreenode<T> newR) { right = newR; }
```

   Write the `findNegatives` method whose header is given below.

```
public static List<Integer> findNegatives( BinaryTreenode<Integer> n)
```

   The method should return a list containing all the negative values in a binary tree containing `Integer` data. For example, if the tree pointed to by n looks like this:



then `findNegatives(n)` should return a list containing -6, -4, -1, and -3 (not necessarily in this order). If the same value appears more than once in the tree, it should also appear more than once in the result list.

Part A: First, complete the English descriptions of the base and recursive cases, like what was given above for Question 1.

- The list of negative values in an empty tree is the empty list.
- The list of negative values in a tree with one node is *(fill in your answer here)*
- The list of negative values in a tree with more than one node is *(fill in your answer here)*

Part B: Now write the `findNegatives` method. You may assume that the `List` used to hold negative values is implemented as an `ArrayList`.

3. Assume that **binary search trees** are implemented using a `BSTnode` class that includes the following fields and methods:

```
// fields
private K key;
private BSTnode<K> left, right;

// methods
public K getKey() { return key; }
public BSTnode<K> getLeft()  { return left; }
public BSTnode<K> getRight() { return right; }
public void setLeft(BSTnode<K> newL)  { left  = newL; }
public void setRight(BSTnode<K> newR) { right = newR; }
```

where `K` is a class that implements the `Comparable` interface. For this question you will **write the `secondSmallest` method** whose header is given below.

```
public K secondSmallest(BSTnode<K> n)
```

The method should return the second smallest item in the tree or null if the tree is empty or only has one item. (Note: your method is not required to be recursive.)

Handing in

Please include your name at the top your file.

Put your answers to the questions into one file named `Homework3` with the appropriate file extension, e.g., `Homework3.pdf` (see File Format for acceptable file formats).

Electronically submit your work to the Homework 3 tab on Canvas.