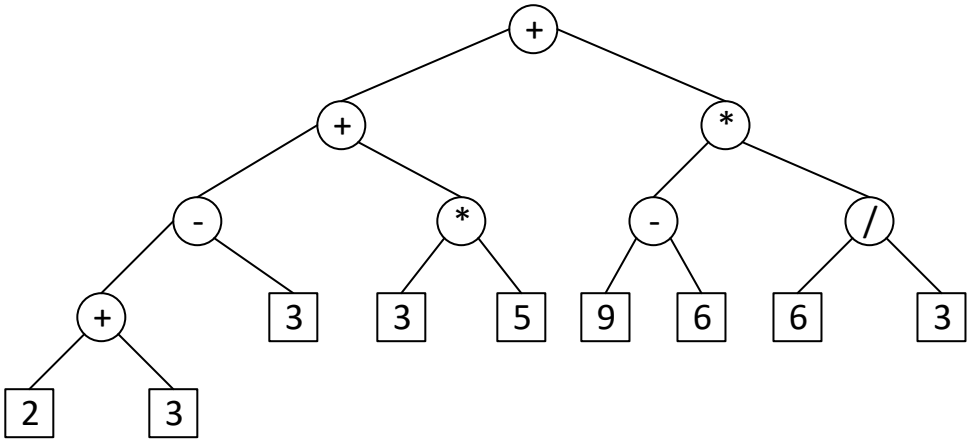
CS526 O2
Homework Assignment 3

Problem 1 (10 points). Consider the following binary tree that stores an arithmetic expression.

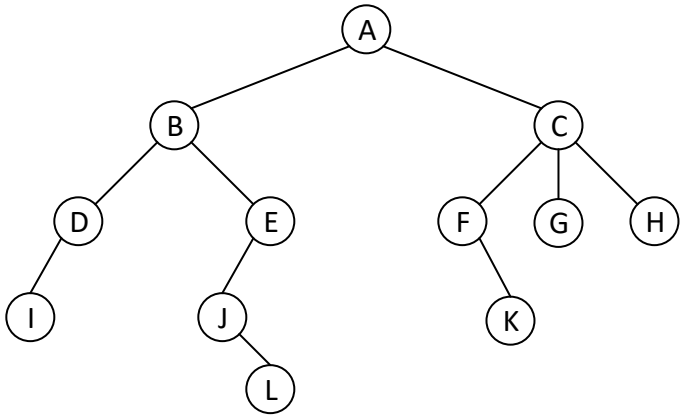


Write the arithmetic expression of the tree and show the value of the expression. When you write the expression, make sure that you use parentheses in your expression correctly.

Problem 2 (10 points). Draw the arithmetic expression tree that stores the following expression:

((16 / (10 - (2 * 3))) - (2 * (7 - 2)))

For Problem 3, Problem 4, and Problem 5, use the following tree, which stores characters:

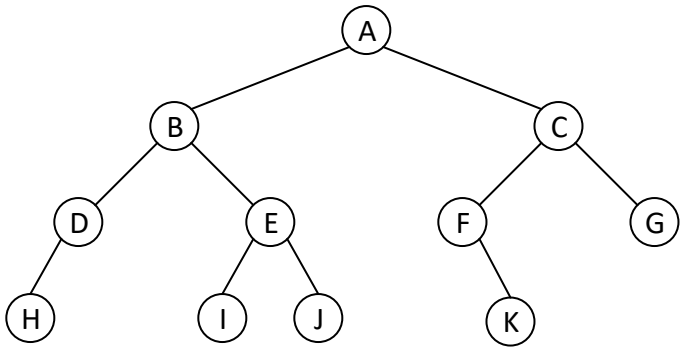


Problem 3 (10 points). Show the sequence of nodes (characters) generated by preorder tree traversal.

Problem 4 (10 points). Show the sequence of nodes (characters) generated by postorder tree traversal.

Problem 5 (10 points). Show the sequence of nodes (characters) generated by breadth-first tree traversal.

Problem 6 (10 points). Consider the following binary tree:



Show the sequence of nodes (characters) generated by inorder tree traversal.

Problem 7 (40 points). This problem is a programming problem.

Write a ***recursive*** method named *makeBinaryTree* within the *IntBST* class, which is posted on Blackboard.

The *IntBST* class implements a binary search tree that stores integers. The *IntBST* class inherits from many other classes and interfaces. All these classes and interfaces, including the *IntBST* class itself, are included in the package *nodeTrees*. You need all files in the package for this problem.

The *makeBinaryTree* method must satisfy the following requirements:

- The signature must be

  ```
  public static IntBST makeBinaryTree(int[] a)
  ```

  You must not change the signature. But, you may write additional method(s) within *IntBST* class if needed.

- This method receives an array of integers with size $n = 2^k - 1$, $k >= 1$. So, $n = 1, 3, 7, 15$, and so on. The integers in the array are sorted in non-decreasing order.
- This method then builds a "complete" binary search tree that contains all integers in the array, and print it on the screen (see an example below). You must also print the size and the height of the tree. Here, a *complete binary tree* is a binary tree where all leaf nodes have the same depth and all internal nodes have 2 children. (this definition is slightly different from the definition in the book).
- A tree must be built **recursively**.

After implementing the method within *IntBST*, use the provided *Hw3_p7.java* program to test your method.

If you run the program with the following array (*a*4 in the main method):

```
{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150}
```

Your output must be:

```
Tree size: 15
Tree height: 3

                    |-- 150
               |-- 140
                    |-- 130
       |-- 120
                    |-- 110
               |-- 100
                    |-- 90
|-- 80
                    |-- 70
               |-- 60
                    |-- 50
       |-- 40
                    |-- 30
               |-- 20
                    |-- 10
```

Note that the tree is printed horizontally.

Do not change any other methods in the *IntBST* class and any other files in the *nodeTrees* package. However, you may override a method in a superclass of *IntBST* and redefine the method within *IntBST*, if needed.

Hint: High-level description of one possible implementation:

- Split the given array into three parts: left subarray, a root node *r*, right subarray. Suppose for example, that the given array is [10, 20, 30, 40, 50, 60, 70]. Then, the left subarray is [10, 20, 30], *r* = 40, and the right subarray is [50, 60, 70].
- Build a binary tree *lt* from the left subarray and build a binary tree *rt* from the right subarray. When you build each subtree, you must do it recursively.
- Make *lt* as the left subtree of *r* and make *rt* as the right subtree of *r*. You need to figure out how to implement this. I suggest that you study other programs in the *nodeTrees* package.

Deliverable

No separate documentation is needed. However, you must include the following in your source code:

- Include the following comments above each method:
  - Brief description of the method
  - Input arguments
  - Output arguments
- Include inline comments within your source code to increase readability of your code and to help readers better understand your code.

You must submit the following files:

- *Hw3_p1_p6.pdf*: This file must include answers to Problem 1 – Problem 6.
- Completed *IntBST.java* file, which includes the *makeBinaryTree* method.
- If you modify *Hw3_p7.java* file, then you must submit the modified file too.

Combine all files into a single archive file and name it *LastName_FirstName_hw3.EXT*, where *EXT* is an appropriate archive file extension, such as *zip* or *rar*.

Grading

Problem 1 through Problem 6

- For each problem, up to 8 points will be deducted if your answer is wrong.

Problem 7

- If your program does not compile, 32 points will be deducted.
- If your program compiles but causes a runtime error, 24 points will be deducted.
- If your method is not a recursive method, 32 points will be deducted.
- If there is no output or output is completely wrong, 24 points will be deducted.
- If your output is partially wrong, up to 24 points will be deducted.