Homework 10

1 Chapter 25: Trees

3. Consider a binary tree that has three levels.

   (a) What is the maximum number of nodes in this tree?

   If level 1 is the root of the tree, then level 1 has a maximum of 1 node, level 2 has a maximum of 2 nodes, and level 3 has a maximum of 4 nodes. That is a total of 7 nodes in a three level tree. A three level tree also has a height of 3. That means there are $2^3 - 1$ nodes, which is equivalent to 7 nodes.

   (b) What is the maximum number of leaves in this tree?

   To have the maximum number of leaves, the tree must be complete. This is because the number of nodes on some level is twice the number of nodes on a previous level. In other words, $N_l = 2 * N_{l-1}$, where N is the number of nodes, l is the level number, and $N_1 = 1$. Therefore, the maximum number of leaves in three level binary tree is 4. We can do this calculation by subtracting the maximum number of nodes in a two level binary tree from the maximum number of nodes in a three level binary tree. $(2^3 - 1) - (2^2 - 1) = 7 - 3 = 4$

   (c) Answer the previous two questions for a binary tree that has ten levels.

   i. The maximum number of nodes in a ten level binary tree is $2^{10} - 1 = 1023$.

   ii. The maximum number of leaves in a ten level binary tree is $(2^{10} - 1) - (2^9 - 1) = 1023 - 511 = 512$

7. Consider a traversal of a binary tree. Suppose that visiting a node means to simply display the data in the node. What are the results of each of the following traversals of the tree in Figure 25-23b

   (a) Preorder: 11 8 3 2 1 5 4 6 10 9 7

   (b) Postorder: 2 1 3 4 6 5 8 9 7 10 11

   (c) Inorder: 2 3 1 8 4 5 6 11 9 10 7

8. The two trees in Figure 25-23 contain integer data.

   (a) Is the tree in Part *a* a binary search tree? Why or why not?

   25-23a is not a binary search tree because the node that contains the key, 8, violates the definition of a binary search tree. It is not the case that the node that contains 8 as its key is greater than all the data in the node's left subtree, in particular the node that contains 9 as its key.

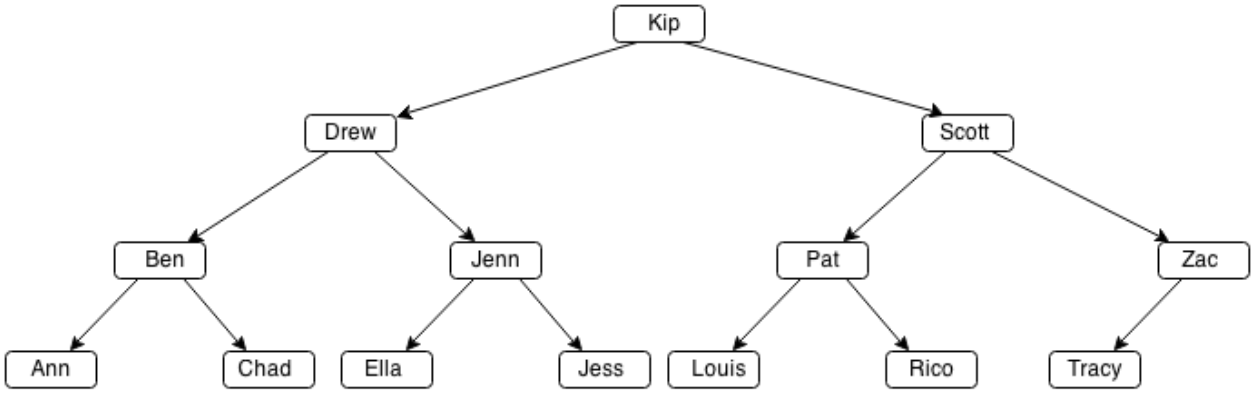
   (b) Is the tree in Part *b* a maxheap? Why or why not?

   25-23b is not a maxheap because the node that contains the key, 5, violates the definition of a maxheap. It is not the case that the node that contains 5, is greater than or equal to its descendant objects, in particular the node that contains 6.

9. Draw the shortest possible binary search tree from the following strings: *Ann, Ben, Chad, Drew, Ella, Jenn, Jess, Kip, Luis, Pat, Rico, Scott, Tracy, Zak*. Is your tree unique?

   - The question lists the elements that will go in the tree, **not** the insertion order.

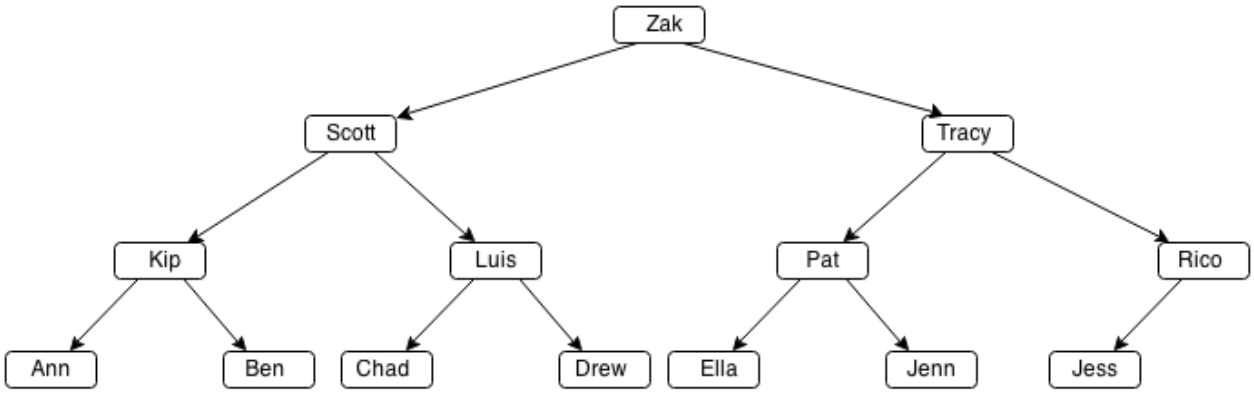
A level three tree can contain a maximum of $2^3 - 1 = 7$ nodes, and a level four tree can contain a maximum of $2^4 - 1 = 15$ nodes. Since our tree will have 14 nodes, the shortest tree we can make will be a height of 4. Also, since we will not have a full level four tree, the tree will not be unique.



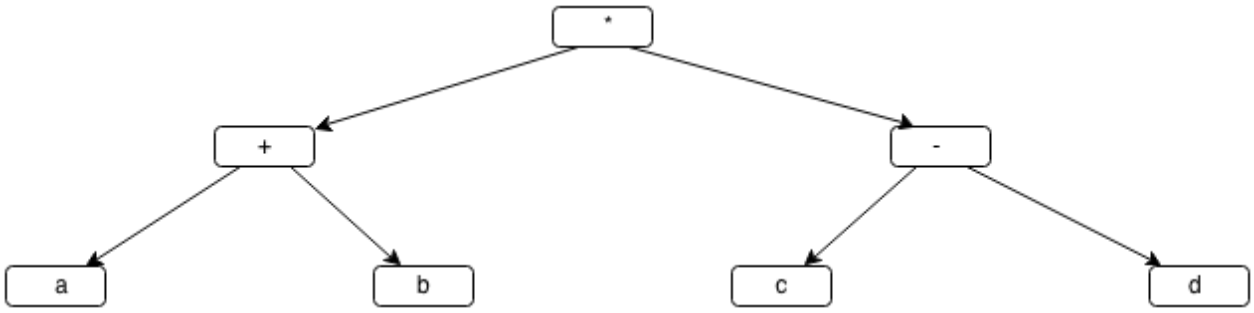
11. Draw a maxheap from the strings given in Exercise 9. Is your maxheap unique?

    - The question lists the elements that will go in the tree, **not** the insertion order.

    By the definition of a maxheap, there is no rule that differentiates the left subtree of a node from its right subtree. The data that will be inserted are not all identical to each other, the shortest binary tree will not be full, and the tree will be larger than a level one binary tree. Therefore, the maxheap will not be unique.



19. Draw an expression tree for the algebraic expression $(a + b) * (c - d)$.



EC Suppose we know that the preorder traversal of a binary search tree is 15 10 8 12 11 19 23 21 24. What is the postorder traversal of the tree?

8 11 12 10 21 24 23 19 15

2 Chapter 26: Tree Implementations

7. Suppose we want to create a method for the class `BinaryTree` that counts the number of times an object occurs in the tree. The header of the method could be as follows:

   ```
   public int count(T anObject)
   ```

   (a) Write this method using a private recursive method of the same name.

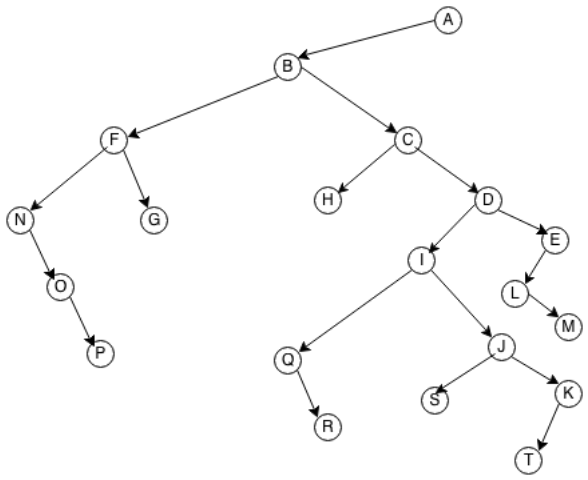
   - This method would go inside the `BinaryTree` class. The recursive method should take only two parameters: a `BinaryNodeInterface<T>` object and an object of type `T`. The code in segment 26.12 should be helpful.

   Refer to the bottom of BinaryTree.java

10. What binary tree represents the general tree in each of the following figures from the previous chapter?

    (a) Figure 25-5

    - Use the algorithm from the textbook, which is also contained in the online notes. For your answer, draw the binary tree.

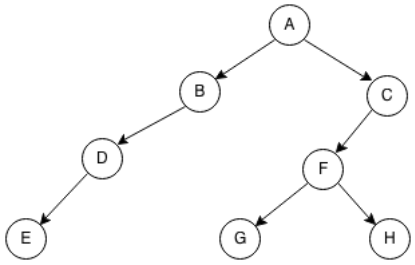


12. Knowing the preorder and inorder traversals of a binary tree will enable you to uniquely define the tree. The same is true for the postorder and inorder traversals.

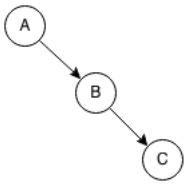
    (a) Draw the unique binary tree that has the following preorder and inorder traversals:

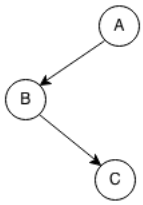
    Preorder: A, B, D, E, C, F, G, H

    Inorder: E, D, B, A, G, F, H, C



13. Although you can uniquely construct a binary tree from either its preorder and inorder traversals or its postorder and inorder traversals, more than one binary tree can have the same preorder traversal and same postorder traversal. Give an example of two different binary trees with the same preorder and postorder traversals.





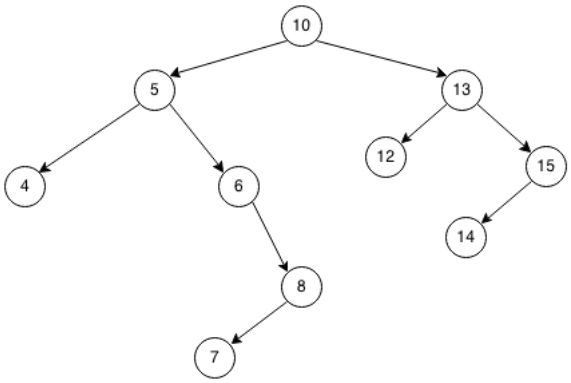
preorder: A B C

postorder: C B A

3 Chapter 27: Binary Search Tree Implementation

1. Show the results of adding the following search keys to an initially empty binary search tree: 10, 5, 6, 13, 15, 8, 14, 7, 12, 4.

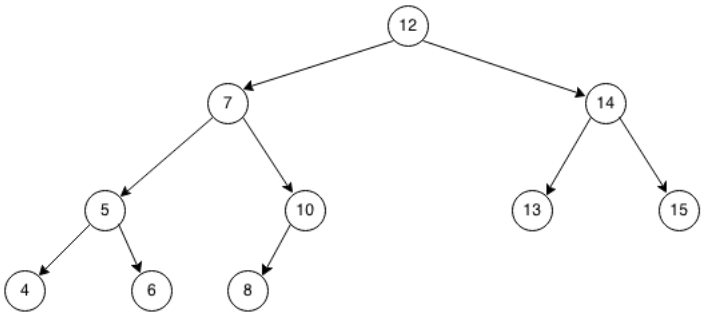
   - The question specifies the insertion order.



2. What ordering of the search keys 10, 5, 6, 13, 15, 8, 14, 7, 12, 4 would result in the most balanced tree if they were added to an initially empty binary search tree?

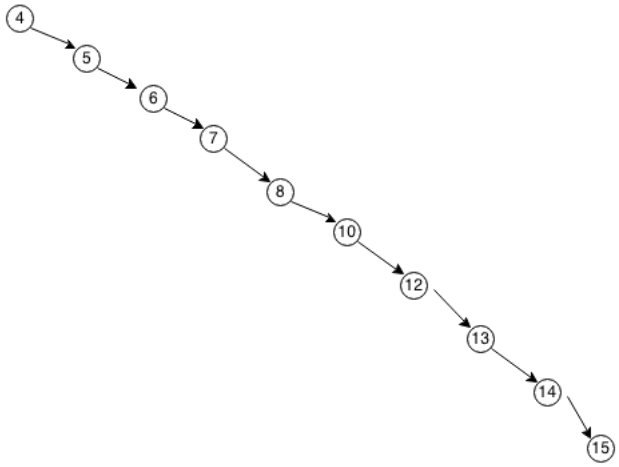
   - For your answer, submit the order **and** a drawing of the tree. Note that there is more than one right answer.

Insert order: 12, 7, 5, 4, 6, 10, 8, 14, 13, 15

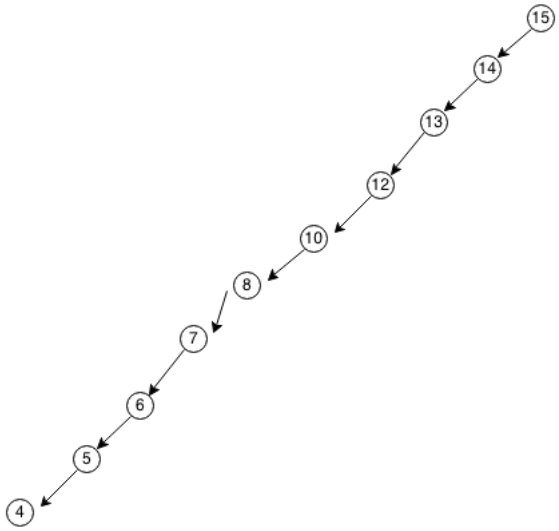


3. Give four different orderings of the search keys 10, 5, 6, 13, 15, 8, 14, 7, 12, 4 that would result in the least balanced tree if they were added to an initially empty binary search tree.
   - Provide **three** orderings. For each ordering, submit the order **and** a drawing of the tree.

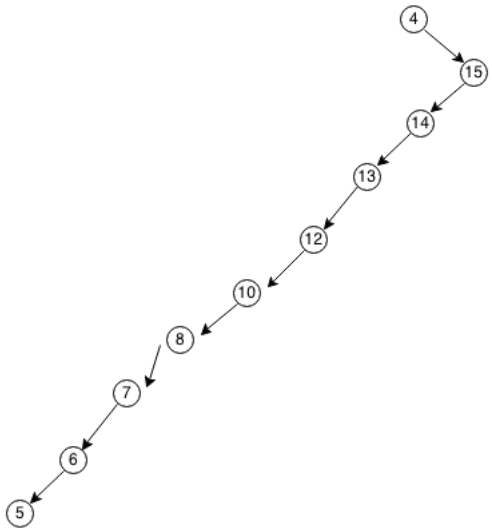
Insert order: 4, 5, 6, 7, 8, 10, 12, 13, 14, 15



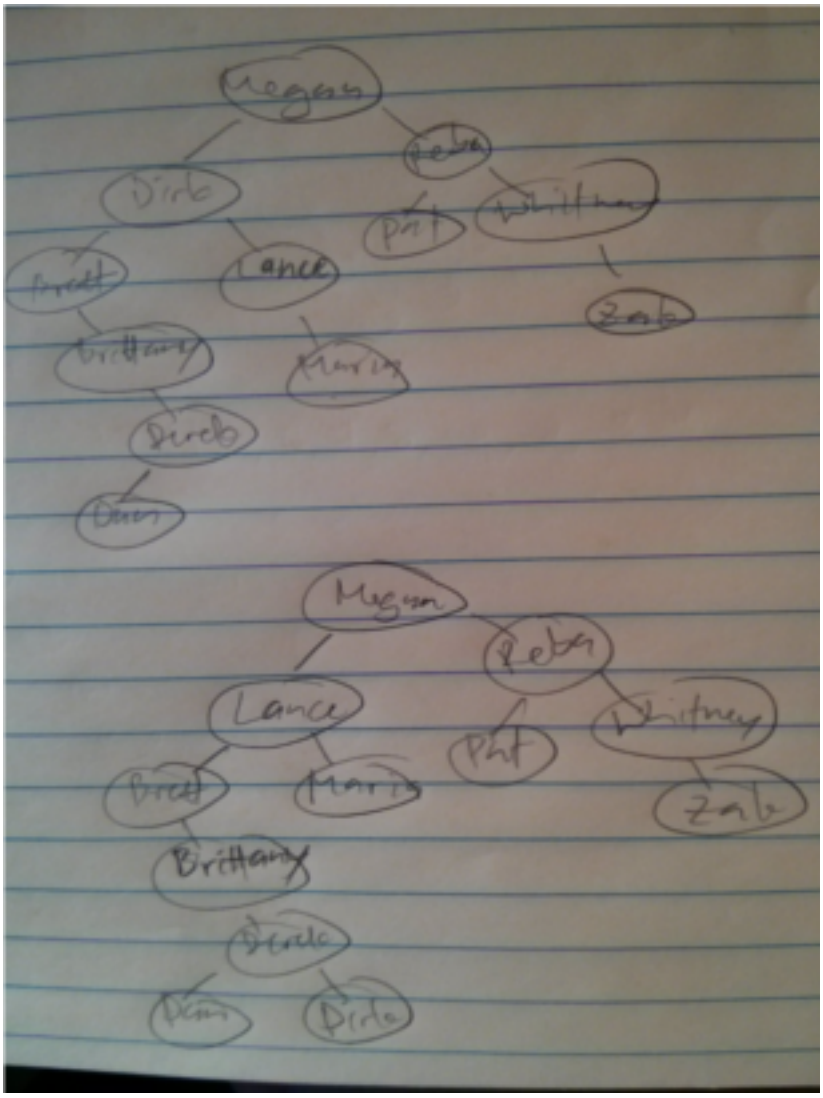
Insert order: 15, 14, 13, 12, 10, 8, 7, 6, 5, 4



Insert order: 4, 15, 14, 13, 12, 10, 8, 7, 6, 5



7. Remove *Doug* from the binary search tree pictured in Figure 27-11d in two different ways.



15. Write an algorithm that returns the smallest search key in a binary search tree.

    - Pseudocode is fine.

Note: This code does not use the ADT Dictionary, so the data member in BinaryNode is the key.

```
public T smallestNode()
{
    BinaryNodeInterface<T> currentNode = getRootNode();
    BinaryNodeInterface<T> previousNode = null;

    while(currentNode != null)
    {
        previousNode = currentNode;
        currentNode = currentNode.getLeftChild();
    }

        if(previousNode != null)
```

```
            return previousNode.getData();

        return null;
    }
```

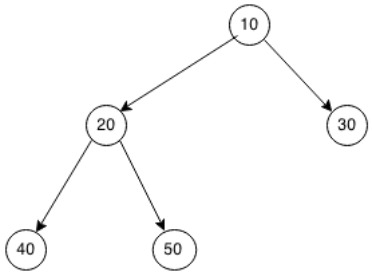
4 Chapter 28: Heap Implementation

1. Trace the formation of a maxheap by the constructor given in Segment 28.16 for each of the following arrays:

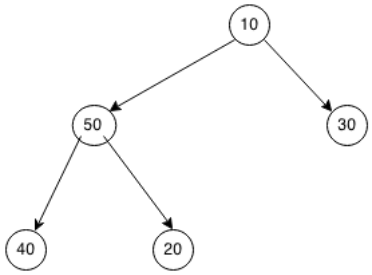
   (a) 10 20 30 40 50

   - This question asks you to trace the creation of the heap by creating a complete tree and then iteratively using the reheap method - this is the second approach described in the online notes. Show the array **and** the tree for each step. Show the initial formation of the array/tree and show what the array/tree look like after *each* call to reheap. Remember that reheap is not called for each node when you use this approach to create a heap.

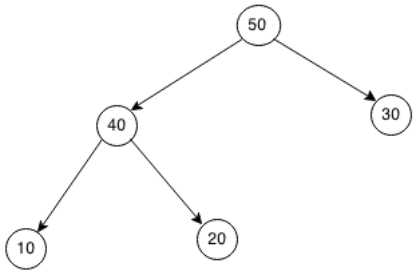
   Original Array: null 10 20 30 40 50



   After reheap(2): null 10 50 30 40 20



   After repeap(1): null 50 40 30 10 20



2. Trace the addition of each of the following values to an initially empty maxheap:

   10 20 30 40 50

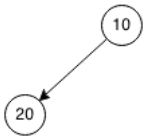
   Compare your trace with the results of Exercise 1a.

   - This question asks you to trace the creation of the heap if each node is added one at a time (instead of adding all of the nodes and restructuring - like you did in #1). Show the tree only for each step (you do not need to show the array). You will show what the tree looks like when a new node is added and you will show what the tree looks like when reheap is called as a result of each addition.

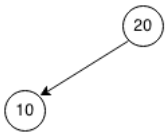
   add 10

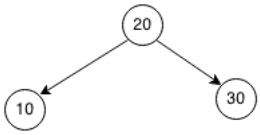
   add 20

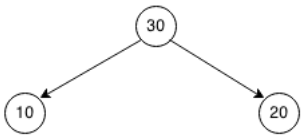
   reheap(1)

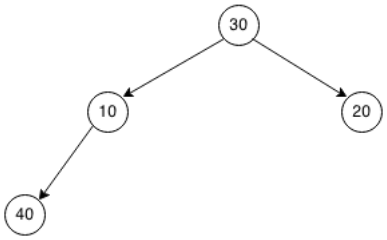
   add 30

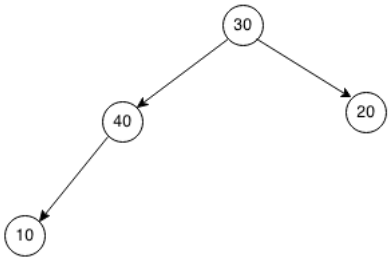
   reheap(1)

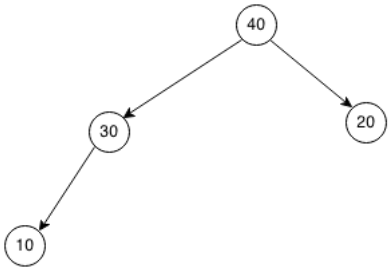
   add 40



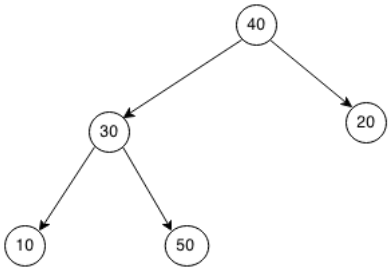
reheap(2)



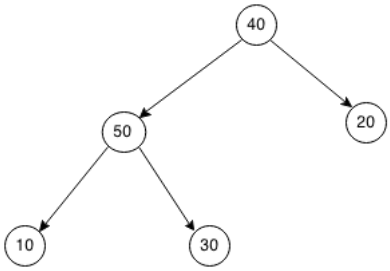
reheap(1)



add 50



reheap(2)



reheap(1)