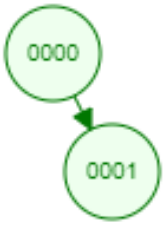
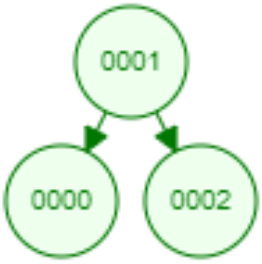
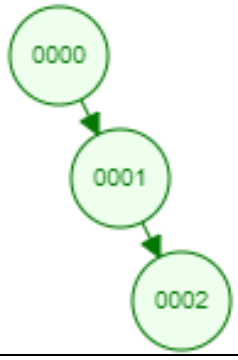
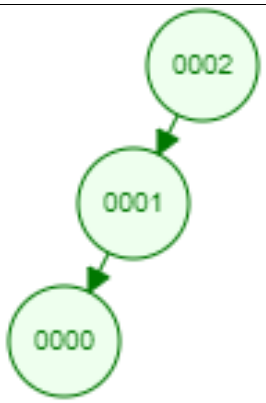
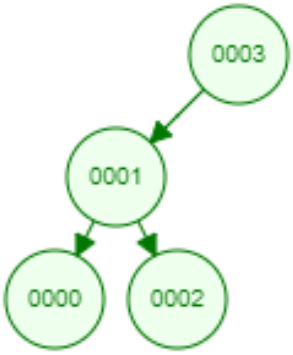
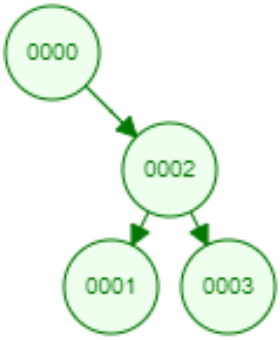
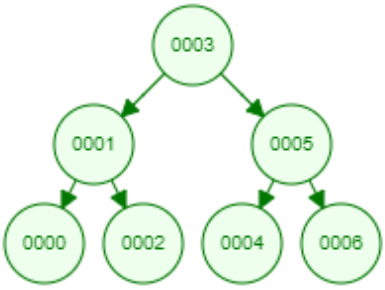
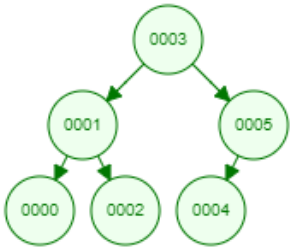
I ran all of my tests on these trees, using arrays populated with the necessary information.

Tree #	Pre-Order	Post-Order	In-Order	Symmetric	Max Imbalance	Visual Representation
0	0	0	0	true	0	
1	01	10	01	false	1	
2	10	01	01	false	1	
3	102	021	012	true	0	
4	012	210	012	false	2	

5	210	012	012	false	2	
6	3102	0213	0123	false	2	
7	0213	1320	0123	false	2	
8	3102546	0214653	0123456	true	0	
9	310254	021453	012345	false	1	

Runs: 1/1 Errors: 0 Failures: 0

BSTTester [Runner: JUnit 4] (0.000 s)

```java
    @Test
    public void test()
    {
        testPreOrder();
        testPostOrder();
        testInOrder();
        testIsSymmetric();
        testMaxImbalance();
        testReverse();
    }
```

```java
    private void checkPre(int i)
    {
        Iterable<Integer> list = trees[i].preOrder();
        int count = 0;
        for (Integer j : list)
        {
            assertTrue(j.equals(pre[i][count++]));
        }
    }

    private void checkPost(int i)
    {
        Iterable<Integer> list = trees[i].postOrder();
        int count = 0;
        for (Integer j : list)
        {
            assertTrue(j.equals(post[i][count++]));
        }
    }

    private void checkIn(int i)
    {
        Iterable<Integer> list = trees[i].inOrder();
        int count = 0;
        for (Integer j : list)
        {
            assertTrue(j.equals(in[i][count++]));
        }
    }

    private void testPreOrder()
    {
        for (int i = 0; i < 8; i++)
        {
            checkPre(i);
        }
    }

    private void testPostOrder()
    {
        for (int i = 0; i < 8; i++)
        {
            checkPost(i);
        }
    }

    private void testInOrder()
    {
        for (int i = 0; i < 8; i++)
        {
            checkIn(i);
        }
    }
```

```java
    private void testIsSymmetric()
    {
        for (int i = 0; i < 8; i++)
        {
            assertEquals(trees[i].isSymmetric(), symmetric[i]);
        }
    }


    private void testMaxImbalance()
    {
        for (int i = 0; i < 8; i++)
        {
            assertEquals(trees[i].maxImbalance(), imbalance[i]);
        }
    }

    private void testReverse()
    {
        for (int i = 0; i < 8; i++)
        {
            testReverse(trees[i]);
        }
    }

    private void testReverse(BST tree)
    {
        Iterable<Integer> inOrder = tree.inOrder();

        BST reversedTree = tree.reverse();

        Iterable<Integer> reversed = reversedTree.inOrder();

        Stack<Integer> reverser = new Stack<Integer>();
        Queue<Integer> actualReversed = new Queue<Integer>();

        for (Integer i : inOrder)
        {
            reverser.push(i);
        }
        while (!reverser.isEmpty())
        {
            actualReversed.enqueue(reverser.pop());
        }

        assertEquals(reversed.toString(), actualReversed.toString());
    }

}
```