The purpose of this lab is to gain experience with binary search trees, implementing various tree operations and comparing the performance of balanced and unbalanced binary search trees.

1. Use the provided code to generate a sequence of integers S of length n, where n is a user-provided parameter (the provided program generates a special type of random sequence called a random walk).

2. Insert the elements of S into a binary search tree T1.

3. Use the following procedure to generate a balanced tree T2
   a. Extract the keys in T1 into a sorted array
   b. Build balanced binary search tree T2 with the elements of T1, as discussed in class.

4. Compute and display the heights of T1 and T2.

5. Display the contents and structure of T1 and T2 as follows: elements are displayed in a reverse inorder traversal, one per line, with an indentation that is proportional to the node’s depth.

   Thus the following tree

   ![Tree Diagram]

   Would be represented as:

   14
   13
   10
   8
   7
   6
   4
   3
   1

   Notice that the tree structure can be easily inferred from this representation – just lean your head to the left.

6. Generate m integers in the range \([\min(T1),\max(T1)]\) and search for each of them in T1, computing the running time of the sequence of searches

7. Repeat the previous step using tree T2

8. Write a report showing the results of your methods and describing your experiments, in particular the running times of steps 6 and 7 for different values of m and n. For large values of n, show the trees’ heights but don’t display them.
You must implement your own tree methods, using the java built-in libraries is not allowed. In order to complete the assignment, you will need to implement (at least) methods to do the following:
1. Insert a new element to a BST – notice that duplicates will be allowed.
2. Extract the elements of a BST and store them in a sorted array.
3. Print the contents and structure of a BST.
4. Find the minimum element in a BST
5. Find the maximum element in a BST
6. Search for a given value in a BST
7. Compute the height of a BST.

Appendix: Random walk code

```java
import java.util.*;

public class randomWalkArrayGenerator{
    // CS2302 - Fall 2010
    // Generates a random walk sequence and stores it in an array
    // Programmed by Olac Fuentes

    public static int[] randomWalkArray(int n){
        Random generator = new Random();
        int []randArray = new int[n];
        randArray[0]=0;
        for(int i=1;i<n;i++)
            randArray[i] = randArray[i-1]+ generator.nextInt(201)-100;
        return randArray;
    }

    public static void main(String[] args) {
        int n=10000;

        int [] myArray;
        myArray = randomWalkArray(n);
        for(int i=0;i<n;i++)
            System.out.println(myArray[i]+" ");
    }
}
```