Contents

Trees

Solutions for those exercises.

Exercises

1. Consider the following tree:
   (a) Explain why it is **not** a binary search tree.
   (b) Pick one among *inorder*, *preorder* and *postorder* traversal, and give
      i. A brief description of how it proceeds,
      ii. What it would produce for the given tree.
2. Consider the following implementation of "random" binary trees:

```
public class RBTree<T>
{

private class Node
    {
    public T Data { get; set; }
    public Node left;
    public Node right;
    public Node(
        T dataP = default(T),
        Node leftP = null,
        Node rightP = null
        )
        {
            Data = dataP;
            left = leftP;
            right = rightP;
        }
    }

private Node root;

public RBTree()
```

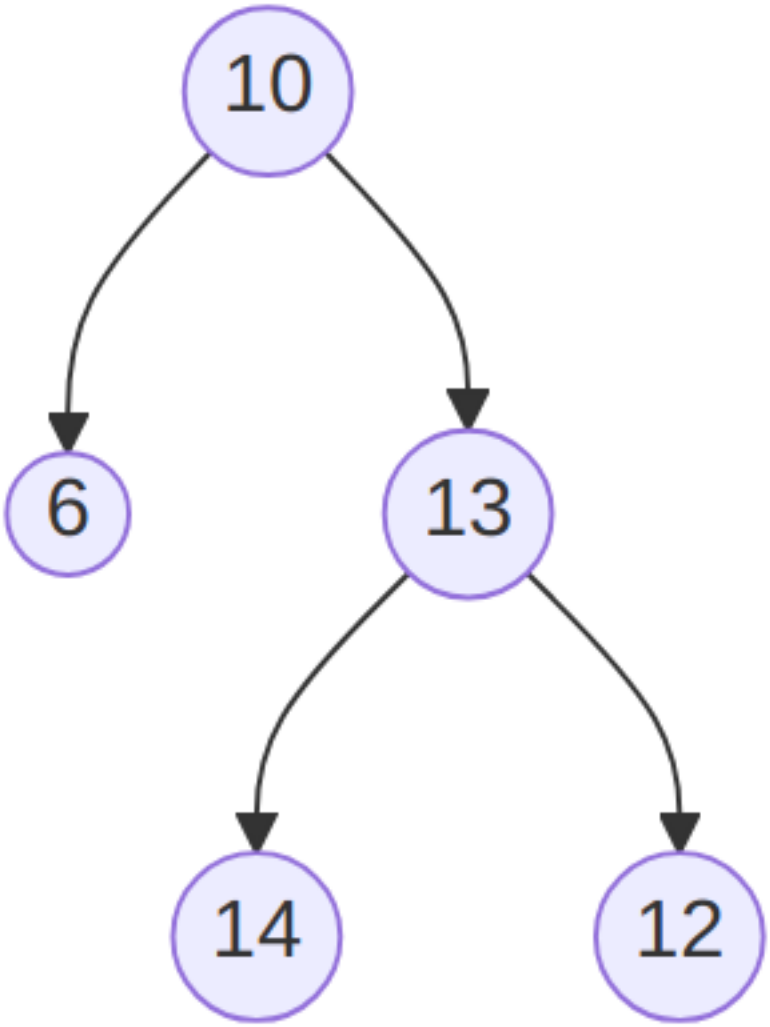

Figure 1: A binary tree that is not a binary search tree. (text version, image version, svg version)

```
    {
        root = null;
    }

public void Insert(T dataP)
    {
        root = Insert(dataP, root);
    }

private Node Insert(T dataP, Node nodeP)
    {
        if (nodeP == null)
        {
            return new Node(dataP, null, null);
        }
        else
        {
            Random gen = new Random();
            if(gen.NextDouble() > 0.5)
            {
               nodeP.left = Insert(dataP, nodeP.left);
            }
            else
            {
                nodeP.right = Insert(dataP,
 ↪  nodeP.right);
            }
        }
        return nodeP;
    }
}
```

Note that the `Insert(T dataP, Node nodeP)` method uses the `gen.NextDouble() > 0.5` test that will be randomly **`true`** half of the time, and **`false`** the other half.

(a) Explain the `T dataP = default(T)` part of the `Node` constructor.

(b) Write a `ToString` method for the `Node` class, remembering that only a node `Data` needs to be part of the `string` returned.

(c) Write a series of statements that would

   i. create a `RBTree` object,
   ii. insert the values 1, 2, 3, and 4 in it (in this order).

(d) Make a drawing of a possible `RBTree` obtained by executing your code.

(e) Write a `Find` method that takes one argument `dataP` of type `T` and returns **`true`** if `dataP` is in the `RBtree` calling object, **`false`** otherwise.