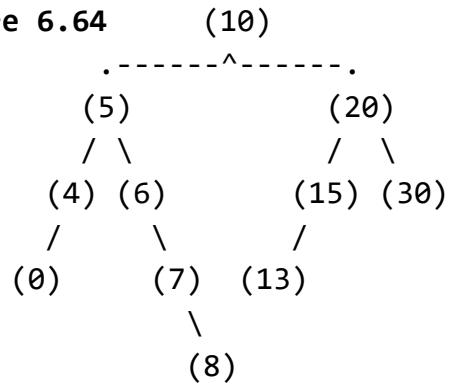
Kochepasova Olga
COSC 311 Homework hw1015
Distributed: 10/15/2019 Due: 10/22/2019

1. Drozdek Chpt 6 Exercise: 5 (6:5)

Figure 6.64



Apply `preorder()`, `inorder()`, and `postorder()` to the tree in Figure 6.64 if `visit(p)` in `IntBSTNode` is defined as:

a.
```
if (p.left != null && p.key - p.left.key < 2)
     p.left.key += 2;
```

```
preorder:  10 5 6 0 6 7 8 20 15 13 30
inorder:   0 4 5 6 7 8 10 13 15 20 30
postorder: 0 4 8 7 6 5 13 15 30 20 10
```

b.
```
if (p.left == null)
     p.right = null;
```

```
preorder:  10 5 4 0 6 20 15 13 30
inorder:   0 4 5 6 10 13 15 20 30
postorder: 0 4 8 7 6 5 13 15 30 20 10
```

c.
```
if (p.left == null)
     p.left = new IntBSTNode(p.key-1);
```

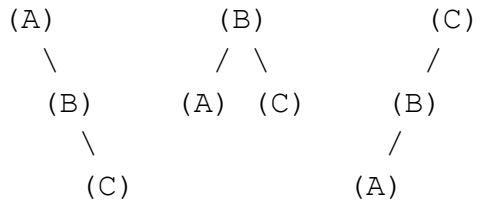
```
preorder:  10 5 4 0 -1 -2 -3 -4 ... ERROR: INFINITE LOOP
inorder:   0 4 5 6 7 8 10 13 15 20 30
postorder: 0 4 8 7 6 5 13 15 30 20 10
```

```
d. {  IntBSTNode tmp = p.right;
      p.right = p.left;
      p.left = temp;
}
```

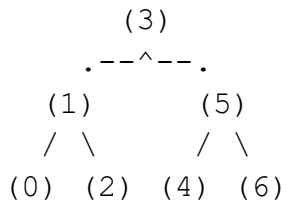
```
preorder:  10 20 30 15 13 5 6 7 8 4 0
inorder:   0 4 0 5 4 10 6 5 7 6 8 7 8
postorder: 0 4 8 7 6 5 13 15 30 20 10
```

2. Drozdek 6:8 (A < B < C)

Draw all possible binary search trees for three elements: A, B, C.



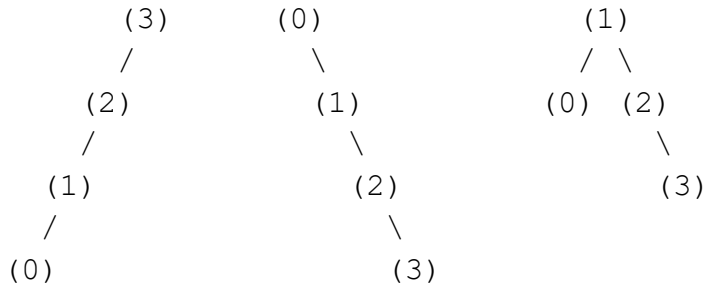
3. Consider data {0, 1, ... 6} Give the binary search tree with the best balance.



4. Give a sequence of data that will create the balanced binary search tree of #3.

```
levelorder: {3, 1, 5, 0, 2, 4, 6}
```

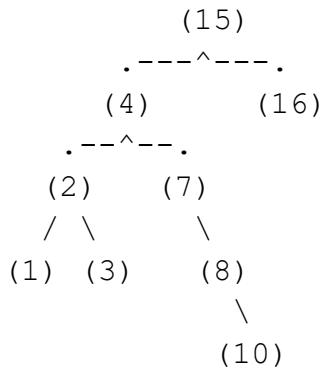
5. Consider the data {0, 1, 2, 3} . Draw three degenerate binary search trees (i.e. height of tree is 3).



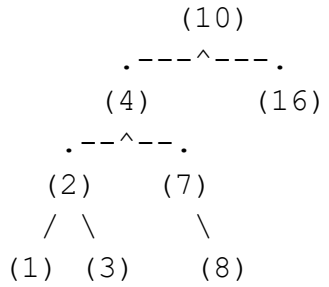
6. Consider the binary search tree created by inserting to empty tree the data in the following order:

15, 16, 4, 7, 2, 1, 3, 8, 10

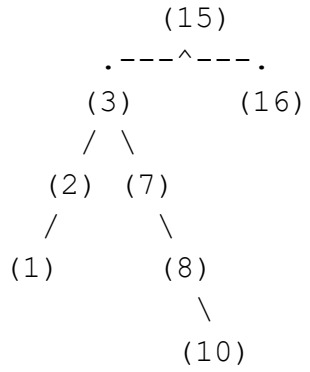
Show the tree.



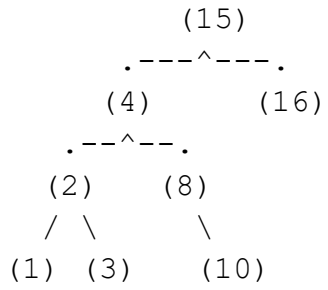
7. For the tree you created in #6, show the tree that results from deleting value 15.



8. For the tree you created in #6, show a tree that results from deleting value 4 (there are two possibilities).



9. For the tree you created in #6, show the tree that results from deleting value 7.



10. For the tree you created in #6, give the following traversals:

infix: 1, 2, 3, 4, 7, 8, 10, 15, 16

postfix: 1, 3, 2, 10, 8, 7, 4, 16, 15

prefix: 15, 4, 2, 1, 3, 7, 8, 10, 16

BREADTH-first: 15, 4, 16, 2, 7, 1, 3, 8, 10