CSC 212 Homework # 4 Solution to Selected Problems
BT & BST

Problem 1

1.
2.
```
private void swap(BTNode<T> t) {
        if (t == null)
                return;
        if (t.left != null) {
                T tmp = t.data;
                t.data = t.left.data;
                t.left.data = tmp;
        } else if (t.right != null) {
                T tmp = t.data;
                t.data = t.right.data;
                t.right.data = tmp;
        }
        swap(t.left);
        swap(t.right);
}
```

Problem 2

1.
```
public static <T> LinkedList<T> collectLeaves(BT<T> bt) {
        LinkedList<T> l = new LinkedList<T>();
        if(!bt.empty()){
                bt.find(Relative.Root);
                recCollectLeaves(bt, l);
        }
        return l;
}
private static <T> void recCollectLeaves(BT<T> bt, LinkedList<T>
    l) {
        boolean leaf = true;
        if(bt.find(Relative.LeftChild)) {
                leaf = false;
                recCollectLeaves(bt, l);
```

```
                bt.find(Relative.Parent);
        }
        if(bt.find(Relative.RightChild)) {
                leaf = false;
                recCollectLeaves(bt, l);
                bt.find(Relative.Parent);
        }
        if(leaf) {
                l.insert(bt.retrieve());
        }
}
// This method is traversing the tree post-order. But notice
   that because we are collecting only leaf nodes, all three
   orders of traversal (pre-order, in-order and post-order)
   give the same result.
```

2.

Problem 3

1.
2.

```
public static boolean find(BT<Integer> bt, int k) {
        if(bt.empty())
                return false;
        Relative dir;
        bt.find(Relative.Root);
        do {
                if (k == bt.retrieve())
                        return true;
                if (k < bt.retrieve())
                        dir = Relative.LeftChild;
                else
                        dir = Relative.RightChild;
        } while(bt.find(dir));
        return false;
}
```

Problem 4

1.
2.

```
public void printReverse() {
        recPrintReverse(root);
}
private void recPrintReverse(BSTNode<T> t) {
        if(t == null)
                return;
        recPrintReverse(t.right);
        System.out.println(t.key);
```

```
        recPrintReverse(t.left);
}
```

Problem 5