You will submit your solution to this assignment to the Curator System (as `HW1`). Your solution must be either a plain text file (e.g., NotePad) or a typed MS Word document; submissions in other formats will not be graded.

Partial credit will only be given if you show relevant work.

---

1. [50 points] Write an implementation of an algorithm to perform a range search in a BST. Base your solution on the BST interface given for Minor Project 2, and assume that the following public method has been added to the interface:

```
 public Vector<T> rangeSearch(T lower, T upper) {

    Vector<T> matches = new Vector<T>();
    rangeSearchHelper(lower, upper, root, matches);
    return matches;
}
```

Complete the implementation by writing the body of the private recursive helper function shown below.

```
private void rangeSearchHelper(T lower, T upper, BinaryNode sroot,
                                                        Vector<T> matches) {

    if ( sroot == null ) return;   // nothing of interest here

    if ( sroot.element.compareTo(lower) >= 0 &&
         sroot.element.compareTo(upper) <= 0 ) { // current elem is in range
         matches.add(sroot.element);
    }
    if ( sroot.element.compareTo(lower) > 0 ) {  // may be matches in left
                                                 //   subtree
       rangeSearchHelper(lower, upper, sroot.left, matches);
    }
    if ( sroot.element.compareTo(upper) <= 0 ) {  // may be matches in right
                                                  //   subtree
       rangeSearchHelper(lower, upper, sroot.right, matches);
    }
}
```

Your implementation should operate as efficiently as possible. It should put references to all the matching data objects, if any, into the `Vector` object that is returned by the public function.

2. [25 points] Suppose you have a collection of *N* different integer values, where *N* is at least 2. If you insert the values into a BST in one order, and then insert the same values into a second BST in a different order, is it possible that the two resulting BSTs will have the same structure? If yes, give an example. If no, explain clearly why not.

Yes, it is possible. For example, consider inserting the values {50, 25, 75} in that order and then in the order {50, 75, 25}. You get precisely the same tree in both cases.

3. [25 points] Suppose you have a collection of *N* different points in the xy-plane, where *N* is large. If you insert the points into a PR quadtree that does not use buckets, then some of the branches may be very "stalky". If you insert the same points into a PR quadtree that does use buckets, you would expect that there would be fewer "stalky" branches.

If a bucket size of 4 is used, what would you expect the effect to be on a typical PR quadtree branch? That is, if you compared branches the two trees that contained the same subsets of points, what difference would you expect to see? Quantify your answer if possible; that is, what is the minimum difference you would expect and what is the maximum difference you would expect, and why?

Crudely put, we'd expect that branches in the bucketed tree would be shorter.

More precisely, we'd expect that <u>some</u> branches would be shorter, but not all. In particular, whenever the data set contains four data points that lie within the same quadrant of a region, the bucketed tree will store all four in a single leaf while the unbucketed version will have at least one additional level due to partitioning the quadrant that holds the four points at least once.

As for an upper bound on the shortening effect, that would depend entirely on how tightly bunched the data elements are. Following the result given in the Samet paper and cited in the course notes, if no buckets are used (or bucket size 1 if you prefer), the upper bound on the height of the tree would depend upon the minimum diagonal of a square that contained the four points in question. So, it's not possible to state a precise upper bound on how many levels would be saved by a bucket size of 4 without having specific information regarding the actual data points to be stored.