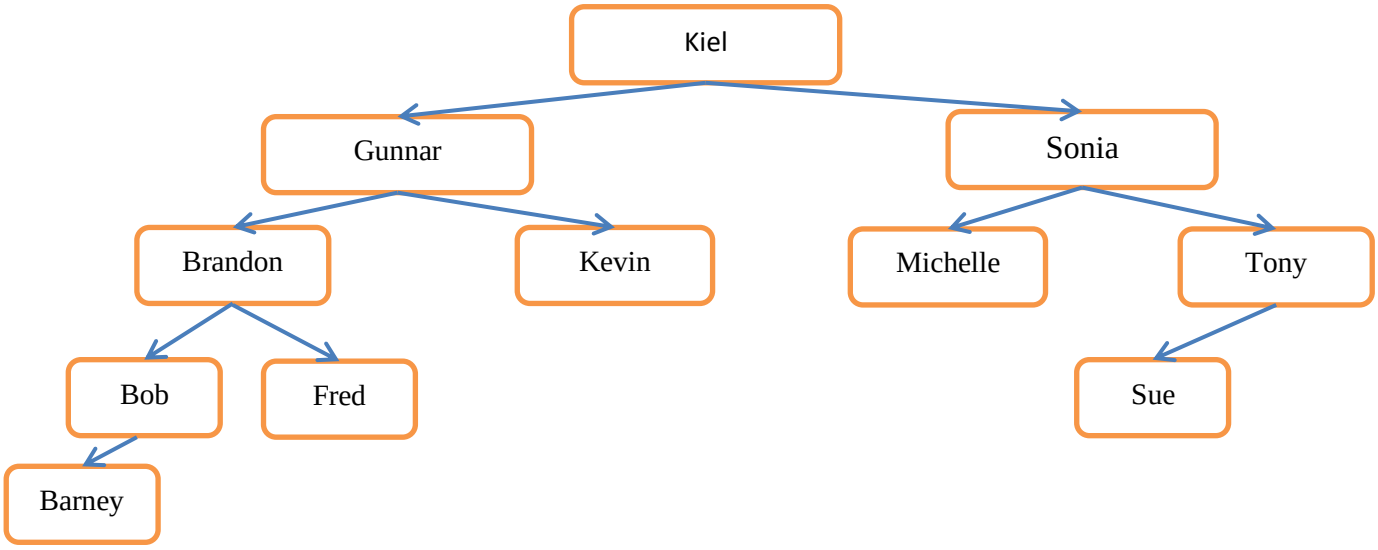
CS 145 Final Exam

Practice Problems

Binary Search Tree

Draw a picture below of the binary search tree that would result from inserting the following names into an empty binary search tree in the following order: Kiel, Gunnar, Sonia, Kevin, Tony, Brandon, Michelle, Bob, Sue, Fred, Barney, Gunnar, Tony



HashArray

Using an array of size 7, what would happen if you preformed the following additions using the normal mod 7 hash with probing.

- .add(5)
- .add(6)
- .add(11)
- .add(12)
- .add(13)
- .remove(4)
    - Does nothing, there is no 4
- .add(3)
- .add(10)

[0]	[1]	[2]	[3]	[4]	[5]	[6]
					5	
					5	6
				11	5	6
12				11	5	6
12	13			11	5	6
12	13			11	5	6
12	13		3	11	5	6
12	13	10	3	11	5	6

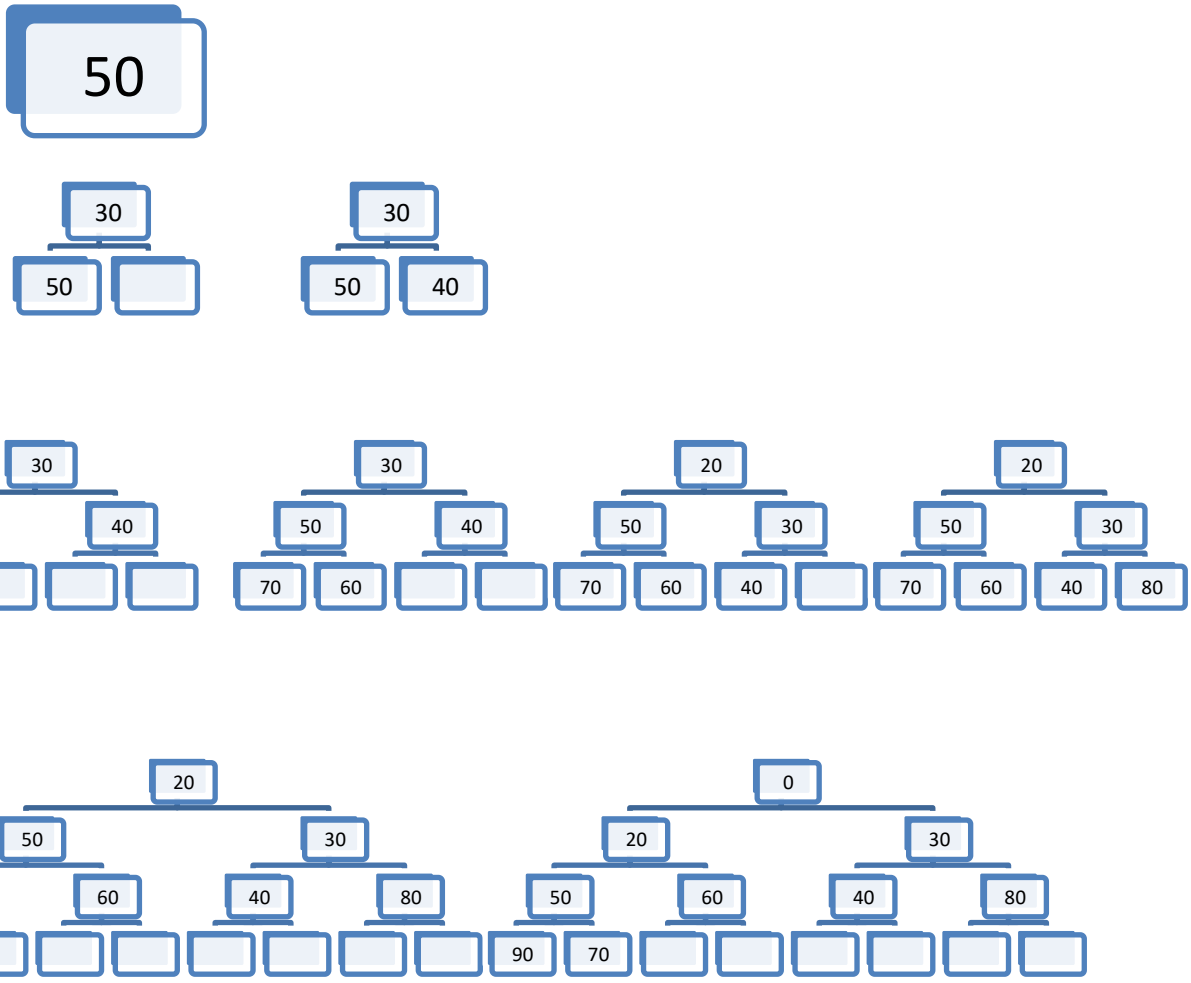
Heap

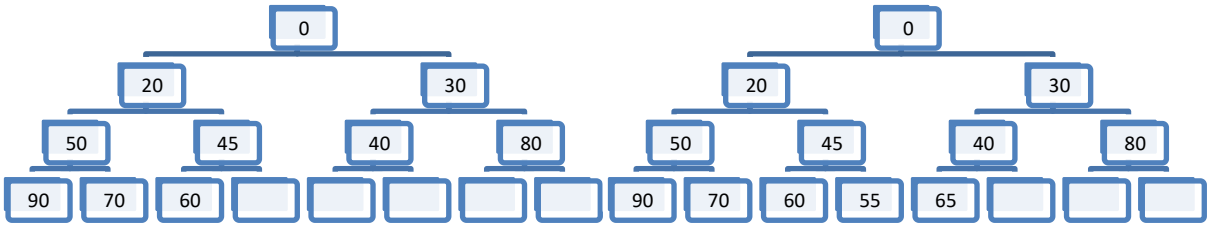
Starting with an empty min-heap, add the following values in this order.

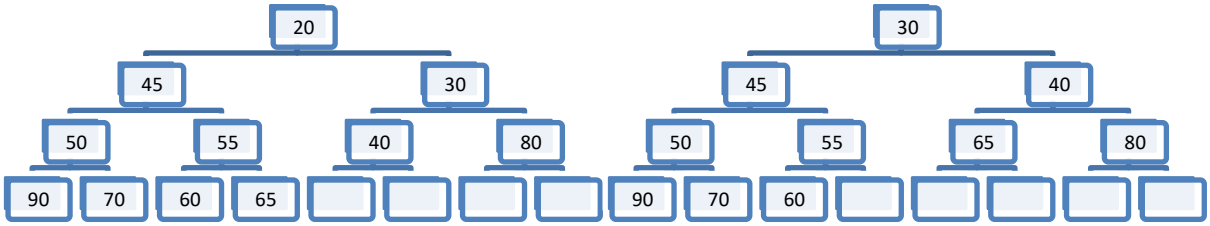
50 30 40 70 60 20 80 90 0 45 55 65

Now pop off the smallest element twice, what does the new array look like?

What would the tree look like as an array?








```
Array Form :  [30 , 45, 40, 50, 55, 65, 80, 90, 70, 60]
```

OR

```
Array Form :  [0, 30 , 45, 40, 50, 55, 65, 80, 90, 70, 60]
```

Depending on if you write the initial zero or not.

Recursion

Write a recursive method that finds the power of 2 to the x given that x is non-negative. Do not use Math.Pow() or any loops.

Examples:

- recursive2Power(2) = 4
- recursive2Power(3) = 8
- recursive2Power(0) = 1
- recursive2Power(-1) = error

```
public static int recursive2Power(int val)
{
  if (val < 0)
  {
    System.out.println("Error");
    return 0;
  }
  if (val == 0)
  {
    return 1;
  }
  else return recursive2Power(val - 1) * 2;
}
```

Tree

Write a method `trim` that could be added to an `IntTree` class. The method accepts minimum and maximum integers as parameters and removes from the tree any elements that are not within that range, inclusive. For this method you should assume that your tree is a binary search tree (BST) and that its elements are in valid BST order. Your method should maintain the BST ordering property of the tree.

Assume that you are adding this method to the IntTree class as defined below:

```
public class IntTree {
      private IntTreeNode overallRoot;
      public void trim(int min, int max)
      {
          if (overallRoot == null) return;
          overallRoot = trim(min, max, overallRoot);
       }

      private IntTreeNode trim(int min, int max, IntTreeNode cNode)
      {
                  if (cNode == null) { return null;}

                  if (cNode.data < min)
                  {
```

```
                return trim(min, max, cNode.right);
            }
            else if (cNode.data > max)
            {
                return trim(min, max, cNode.left);
            }
            else
                {
               cNode.right = trim(min, max, cNode.right);
               cNode.left = trim(min, max, cNode.left);
               return cNode;
            }

    }
```

Inheritance

Consider the following class definitions.

```
public class Pepsi extends Soda {
      public void meth2() {
            System.out.println("Pepsi2");
            meth1();
      }
}

public class Soda {
      public void meth1() {
            System.out.println("Soda1");
      }
}

public class DietCoke extends Coke {
      public void meth1() {

      System.out.println("DietCoke1");
      }
}

public class Coke extends Soda {
      public void meth2() {
            System.out.println("Coke2");
            super.meth1();
      }
      public void meth3() {
            System.out.println("Coke3");
            meth1();
      }
}
```

Now assume that the following variables have been declared and initialized.

```
Soda v1 = new Coke();

Soda v2 = new Pepsi();

Soda v3 = new DietCoke();

Object v4 = new Soda();

Coke v5 = new DietCoke();

Coke v6 = new Coke();
```

In the table below, indicate in the right-hand column the output produced by the statement in the left-hand column. If the statement produces more than one line of output, indicate the line breaks with slashes as in "a/b/c". If the statement causes an error, fill in the right-hand column with "error".

Statement	Output
`v1.meth2();`	error Soda, doesn't have meth2
`v2.meth2();`	error Soda, doesn't have meth2
`v3.meth2();`	error Soda, doesn't have meth2
`v4.meth2();`	error Object, doesn't have meth2
`v5.meth3();`	Coke3 / DietCoke1
`v6.meth3();`	Coke3 / Soda1
`((Object)v2).meth1();`	error Object, doesn't have meth1
`((Soda)v5).meth1();`	DietCoke1
`((Pepsi)v4).meth2();`	error, Cans$Soda cannot be cast to Cans$Pepsi
`((Soda)v4).meth2();`	error Soda, doesn't have meth2

Linked Lists

Write a method that you could add to a singly linked list that would remove the second to last element of the linked list.

What if the linked list was a doubly linked list?

```
      public void remove2nd()
      {
        Node temp = frontNode;
        Node beforeTemp;
        if (temp == null) return;
        if (temp.next == null) return;
         while (temp.next.next != null)
         {
           beforeTemp = temp;
          temp = temp.next;
         }
         beforeTemp.next = temp.next;
          // For DLL add the next line
         // beforetemp.next.previous = beforeTemp;

      }

Method #2
// Doesn't work with DLL
      public void remove2nd()
      {
        if (frontNode == null) return;
        if (frontNode.next == null) return;
         frontNode = remove2nd(frontNode)
      }

      private Node remove2nd(node cNode)
      {
         if (cNode.next.next == null) return cNode.next;
         else
         {
           cNode.next = remove2nd(cNode.next);
           return cNode;
         }
      }
```

Sets

Write a static method numInCommon that takes two Lists of integers as parameters and returns the number of unique integers that occur in both lists. Use one or more Sets as storage to help you solve this problem.

For example, if one list contains the values [3, 7, 3, -1, 2, 3, 7, 2, 15, 15] and the other list contains the values [-5, 15, 2, -1, 7, 15, 36], your method should return 4 (because the elements -1, 2, 7, and 15 occur in both lists).

```
// version 1
public int numInCommon(List<Integer> l1, List<Integer> l2)
{
      Set<Integer> s1 = new Set<Integer>();
      Set<Integer> s2 = new Set<Integer>();

      for(Integer x : l1) s1.add(x);
      for(integer x : l2) if(s1.contains(x)) s2.add(x);

      return s2.size();
}

// OR version 2
public int numInCommon(List<Integer> l1, List<Integer> l2)
{
      Set<Integer> s1 = new Set<Integer>();
      for(Integer x : l1) if(l2.contains(x)) s1.add(x);
      return s1.size();
}

//  OR version 3
public int numInCommon(List<Integer> l1, List<Integer> l2)
{
      Set<Integer> s1 = new Set<Integer>();
      Set<Integer> s2 = new Set<Integer>();
      for(Integer x : l1) s1.add(x);
      for(Integer x : l2) s2.add(x);
      s1.retainAll(s2);
      return s1.size();
}
```

Maps

Write a method `intersect` that takes two Maps of strings to integers as parameters and that returns a new map whose contents are the intersection of the two. The intersection of two maps is defined here as the set of keys and values that exist in both maps. So if some value K maps to value V in both the first and second map, include it in your result. If K does not exist as a key in both maps, or if K does not map to the same value V in both maps, exclude that pair from your result. For example, consider the following two maps:

{Janet=87, Logan=62, Whitaker=46, Alyssa=100, Stefanie=80, Jeff=88, Kim=52, Sylvia=95}

{Logan=62, Kim=52, Whitaker=52, Jeff=88, Stefanie=80, Brian=60, Lisa=83, Sylvia=87}

Calling your method on the preceding maps would return the following new map (the order of the key/value pairs does not matter):

{Logan=62, Stefanie=80, Jeff=88, Kim=52}

```
public static Map<String,Integer> instersect(Map<String,Integer> map1,
                                             Map<String,Integer> map2)
{
  Map<String,Integer> answer = new TreeMap<String,Integer>();


  for (String x : map1.keySet())
  {
   if (map2.containsKey(x) && map2.get(x).equals(map1.get(x)) )
   {
     answer.put(x,map1.get(x));
   }
  }
  return answer;
}
```