0. **What must be true of an array before it can be sorted?**
No requirements. Well, it has to be non-null, but that's it.

1. **What built-in Java command can sort an array?**
   Arrays.sort()

2. **Tracing Algorithms**
   a. For each array below, show what the **BubbleSort** algorithm would make it look like.

   Array at the beginning:
   
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>22</td>
<td>15</td>
<td>-19</td>
<td>31</td>
<td>10</td>
<td>-4</td>
<td>53</td>
<td>67</td>
<td>18</td>
<td>19</td>
</tr>
</tbody>
</table>

   After 1 iteration of BubbleSort:
   
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>-19</td>
<td>22</td>
<td>10</td>
<td>-4</td>
<td>31</td>
<td>53</td>
<td>18</td>
<td>19</td>
<td>67</td>
</tr>
</tbody>
</table>

   After 2 iterations of BubbleSort:
   
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-19</td>
<td>15</td>
<td>10</td>
<td>-4</td>
<td>22</td>
<td>31</td>
<td>18</td>
<td>19</td>
<td>53</td>
<td>67</td>
</tr>
</tbody>
</table>

   Array at the beginning:
   
   "mogwai" | "elvis" | "ccr" | "rem" | "abba" | "juno" | "who" | "u2" | "prince" | "gomez"

   After 1 iteration of BubbleSort:
   
   "elvis" | "ccr" | "mogwai" | "abba" | "juno" | "rem" | "u2" | "prince" | "gomez" | "who"

   After 2 iterations of BubbleSort:
   
   "ccr" | "elvis" | "abba" | "juno" | "mogwai" | "rem" | "prince" | "gomez" | "u2" | "who"

   b. For each array below, show what the **MergeSort** algorithm would make it look like.

   Array at the beginning:
   
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>22</td>
<td>15</td>
<td>-19</td>
<td>31</td>
<td>10</td>
<td>-4</td>
<td>53</td>
<td>67</td>
<td>18</td>
<td>19</td>
</tr>
</tbody>
</table>

   After 1 iteration of Mergesort:
   
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>20</td>
<td>-19</td>
<td>31</td>
<td>-4</td>
<td>10</td>
<td>53</td>
<td>67</td>
<td>18</td>
<td>19</td>
</tr>
</tbody>
</table>

   After 2 iterations of MergeSort:
   
   |-19| 15| 20| 31|-4 | 10| 53| 67| 18| 19|

   Array at the beginning:
   
   "mogwai" | "elvis" | "ccr" | "rem" | "abba" | "juno" | "who" | "u2" | "prince" | "gomez"

   After 1 iteration of MergeSort:
   
   "elvis" | "mogwai" | "ccr" | "rem" | "abba" | "juno" | "u2" | "who" | "gomez" | "prince"

   After 2 iterations of MergeSort:
   
   "ccr" | "elvis" | "mogwai" | "rem" | "abba" | "juno" | "u2" | "who" | "gomez" | "prince"
3. Writing short methods involving search and sort
   a. Write a method that takes an array of ints as an argument, plus two additional int arguments called pos1 and pos2. The method should swap the elements of the array at positions pos1 and pos2.

   ```java
   public static void swap(int[] arr, int pos1, int pos2) {
       int temp = arr[pos1];
       arr[pos1] = arr[pos2];
       arr[pos2] = temp;
   }
   ```

   b. Write a method that does one iteration of BubbleSort. It should use the swap method you defined above. It should return true if it performed any swaps during the iteration, and false otherwise.

   ```java
   public static boolean bubbleSortIteration(int[] arr) {
       boolean ret = false;
       for(int i=0; i<arr.length-1; i++) {
           if(arr[i] > arr[i+1]) {
               swap(arr, i, i+1);
               ret = true;
           }
       }
       return ret;
   }
   ```

   c. Implement the BubbleSort algorithm. (That means, write a Java method that takes an array of ints as an argument, and does BubbleSort to the array.) Use the method for one iteration of BubbleSort, which you defined above.

   ```java
   public static void bubbleSort(int[] arr) {
       boolean didSwap = true;
       while(didSwap) {
           didSwap = bubbleSortIteration(arr);
       }
   }
   ```

   // for d and f, see my implementation listed on the course Website, under Sorter.java
   // I'll leave part e up to you.

   d. Implement the Merge algorithm. It should take four int arguments (left1, right1, left2, right2), plus an array of ints as a fifth argument. It should merge the part of the array between left1 and right1 with the part of the array between left2 and right2.

   e. Write a method that does one iteration of MergeSort. It should take two arguments, an array of ints, and an int that size how big of a portion of the array to merge.

   f. Implement the MergeSort algorithm.
4. How long do these algorithms take to run?
   a. MergeSort speed: $O(n \log(n))$
   b. Merge speed: $O(n)$
   c. BubbleSort speed: $O(n^2)$
   d. Binary Search speed: $O(\log(n))$
   e. Linear Search speed: $O(n)$

   f. If it takes 10 seconds to run BubbleSort on an array of 10,000 elements, how long will it take to run BubbleSort on an array of 20,000 elements?

   Since BubbleSort takes time proportional to $n^2$, doubling the number of elements will take \((2n)^2 / n^2 = 4n^2 / n^2 = 4 \text{ times as long. So 20,000 elements should take 40 seconds.}\)

   g. If it takes 10 seconds to run Binary Search on an array of 100,000 elements, how long will it take to run Binary Search on an array of 200,000 elements?

   Since Binary Search takes time proportional to $\log(n)$, doubling the number of elements will take $\log(2n)/\log(n) = (\log(2) + \log(n)) / \log(n) = (1 + \log(n)) / \log(n)$, which is approximately equal to 1, times as long. So 200,000 elements should also take 10 seconds (very slightly more, but not noticeably).

   h. If it takes 10 seconds to run MergeSort on an array of 20,000 elements, how long will it take to run MergeSort on an array of 40,000 elements?

   Since MergeSort takes time proportional to $\log(n)$, doubling the number of elements will take $2n \log(2n) / (n \log(n))$, which is approximately $2n \log(n) / (n \log(n)) = 2$, times as long. So 40,000 elements will take approximately 20 seconds (slightly more).