Practice Problems: Search and Sorting

1. Tracing Algorithms

   a. Look at the example array below. For each key, indicate the *positions* in the array (the indexes, not the values) that a binary search would visit if it was searching for that key.

-20	-12	-9	1	4	16	21	67	75	101
0	1	2	3	4	5	6	7	8	9

Key: 16   Positions visited during binary search: 4, 7, 5

Key: 4   Positions visited during binary search: ?

Key: -25   Positions visited during binary search: ?

Key: 101   Positions visited during binary search: ?

Key: 45   Positions visited during binary search: ?

Key: -9   Positions visited during binary search: ?

b. For each call to the binarySearch method below, write which elements the search procedure visits.

array *x*:

0	1	2	3	4	5	6	7	8	9
-19	-12	4	9	21	22	45	51	99	103

```
int pos = Arrays.binarySearch(x, 21);
```

    1) it first searches the middle point between positions 0 and 9, or position 4 (21).
    2) It sees that the element at position 4 is exactly equal to the key, so it stops and returns that position (4).

    You can write these steps as a sequence of positions, like this:
    4, return 4

```
int pos = Arrays.binarySearch(x, 51);
```

    1) it first searches the middle point between positions 0 and 9, or position 4 (21).
    2) It sees that the element at position 4 is less than the key (21<51), so it rules out all the elements in positions 0 to 4.
    3) Next, it checks the midway point between position 5 and 9, or position 7.
    4) It sees that the element at position 7 is equal to the key, so it returns the position, 7.

    You can write these steps as a sequence of positions, like this:
    4 -> 7, return 7

```
int pos = Arrays.binarySearch(x, 9);
?
int pos = Arrays.binarySearch(x, -15);
?
```

array *y*:

0	1	2	3	4	5	6	7	8	9
"abba"	"ccr"	"elvis"	"gomez"	"juno"	"mogwai"	"prince"	"rem"	"u2"	"who"

```
int pos = Arrays.binarySearch(y, "juno");
?
int pos = Arrays.binarySearch(y, "prince");
?
int pos = Arrays.binarySearch(y, "who");
?
int pos = Arrays.binarySearch(y, "beirut");
?
```

2. Writing short methods involving search

    a. Write a method that takes an int array X as an argument. It should return the *median* value of the array. The median of a set of numbers is defined as the number in the middle position, when the numbers are arranged from smallest to largest.

    b. Write a method that takes an int array X as an argument. It should return true if 0 is in the array, and false otherwise.

3. Given the following method BubbleSort, show the result of the first 2 rounds of iterations (after calling bubbleSortIteration):

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


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

Array **arr** at the beginning:

0	1	2	3	4	5	6	7	8	9
22	15	-19	31	10	-4	53	67	18	19

Array **arr** after 1st iteration of (`bubbleSortIteration`):

Array **arr** after 2nd iteration: