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: 4

Key: -25 Positions visited during binary search: 4, 1, 0

Key: 101 Positions visited during binary search: 4, 7, 8, 9

Key: 45 Positions visited during binary search: 4, 7, 6

Key: -9 Positions visited during binary search: 4, 1, 2

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);
```

`4, return 4`

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

`4 -> 7, return 7`

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

`4 → 1 → 2 → 3, return 3`

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

`4 → 1 → 0, return -1 (because it can't find -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");
```
`4, return 4`

```
int pos = Arrays.binarySearch(y, "prince");
```
`4 → 7 → 5 → 6, return 6`

```
int pos = Arrays.binarySearch(y, "who");
```
`4 → 7 → 8 → 9, return 9`

```
int pos = Arrays.binarySearch(y, "beirut");
```
`4 → 1 → 0, return -1 (can't find "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.

```
public static int median(int [] X)
{
  Arrays.sort(X);  // first, arrange the elements of X in ascending order
  int mid = X.length / 2;
  return X[mid];  // return the number in the middle position

  // technically, if there are an even number of elements in X,
  // the median should be an average between the two middle elements.
  // can you figure out how to modify this method to make that happen?
}
```

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.

```
public static boolean containsZero(int [] X)
{
  // need to sort before searching!
  Arrays.sort(X);

  int pos = Arrays.binarySearch(X, 0);
  return (pos >= 0);
}
```

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 at the beginning:

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

After 1 iteration of BubbleSort:

15	-19	22	10	-4	31	53	18	19	67

After 2 iterations of BubbleSort:

-19	15	10	-4	22	31	18	19	53	67