Exercise 3 MA407

This set of exercises will introduce you to a particularly efficient method to search for an entry in a sorted array, which is called *binary search*. The basic idea of binary search is that the section of the array where the entry is searched is halved in each iteration.

More precisely, suppose you have a sorted array `x[0],...,x[n-1]` of integers. We call these integers *keys* and allow repetitions in the array. When searching for an entry with value `key` in `x` binary search now first looks at the middle element `x[`$\lfloor \frac{n}{2} \rfloor$`]` and compares it to `key`. If `key`$< x[\lfloor \frac{n}{2} \rfloor]$ then we can forget about the upper half of the array and continue searching for `key` in the lower half `x[0],...,x[`$\lfloor \frac{n}{2} \rfloor - 1$`]`. Else we search in the upper half `x[`$\lfloor \frac{n}{2} \rfloor$`],...,x[n-1]`. Assume the former happens (the case that the later happens is analogous). Then we proceed similarly and compare `key` to the middle element `x[`$\lfloor \frac{n}{4} \rfloor$`]` of this part of the array, and again, either continue with the upper half or lower half and so on.

Clearly, one advantage of this algorithm is that it does not need to compare `key` to every element in the array. The first exercise now asks you to say more about the efficiency of this algorithm by finding an upper bound on the number of comparisons needed.

Exercise 3.1.

Determine how many comparisons the above described binary search algorithm has to perform on an array with n entries (in the worst case).

The remainder of this exercise set asks you to implement binary search and some test cases. For this purpose you should write a Java class `BinarySearch` containing the methods described in the exercises below. Please include in this file your answer to Exercise 3.1 as a comment.

Firstly, the class `BinarySearch` should contain a static method `binsearch` which implements binary search by using recursion.

Exercise 3.2.

Write a recursive static method `binsearch` *which, when called with* `binsearch(key,x,i,j)`*, searches for the integer* `key` *among the array entries* `x[i],...,x[j-1]` *using the binary search strategy described above. The entries of* `x` *are assumed to be in sorted order but can have repetitions. The return value of this method should be an integer* `k` *that is the smallest array index with the property* `x[k]==key`*, if there is any such index* `k`*. If not (so* `key` *is not among the entries* `x[i],...,x[j-1]`*) then the return value should be the index* `k` *so that the assignment* `x[k]=key` *would still keep the array sorted (if the array elements* `x[k],...,x[j-1]` *were shifted up to* `x[k+1],...,x[j]`*).*

Here is an example: Suppose the array `x` of integers contains as entries the numbers `x[0]=1`, `x[1]=4`, `x[2]=4`, `x[3]=4`, `x[4]=8`, `x[5]=9`. Then `binsearch(8,x,0,6)` should return (the array index) 4 since `x[4]=8`. Similarly `binsearch(8,x,3,5)` should return 4. Moreover `binsearch(4,x,0,6)` should return 1 since `x[1]=4`, and since 1 is the *first* index i so that `x[i]=4` (of course this is an arbitrary rule, but here we ask for this particular behaviour). `binsearch(7,x,0,6)` should return 4 since the "correct" place for key 7 is at index 4, because in `x[4]` we find the first value in the array that is bigger than 7. Test your `binsearch` method with a small example in the `main` method of your class (see also Exercise 3.4 below).

It would be useful to get from the `binsearch` method the information whether the `key` we were looking for was actually found. However, this method returns an integer, and can therefore not return an additional boolean value to indicate if the `key` was found or not. The next exercise will show you a way around this obstacle by using *static variables*. Static variables (also called class variables) are variables that are declared outside methods in a class, for example as `static boolean found;`. Such a variable can then be used by any method in the class and it keeps its value after a method that used it terminates. Hence such variables can be used to "communicate" information between different methods other than the return values. You can find out more on class variables, e.g., on
`http://www.tutorialspoint.com/java/java_variable_types.htm`

Exercise 3.3.

Add a static variable `found` *of type* `boolean` *to your class. Set this variable to* `false` *before starting a binary search in the* `main` *method, and set it to* `true` *when the key is found during the binary search.*

Finally, we want to include some test cases in the `main` method of `BinarySearch`.

Exercise 3.4.

In the `main` *method include several tests (including elements that are present and not present) of binary search with each of the following three arrays:*

- *the array* $\{1, 3, 6, 7, 7, 7, 9\}$,
- *the array* $\{0, 2, 4, 6, \ldots, 19999998\}$, *of the* 10^7 *even integers from* 0 *up to* $2 \cdot 10^7 - 2$
- *the array* $\{10, \ldots, 10\}$ *of* 10^7 *identical entries of value* 10.

For each test you should print information on whether the `key` *was found, and where. (See the example below.) Moreover you should also document the number of search iterations (that is, the number of recursive calls the method* `binsearch` *performs) in each test. For this purpose, also include a static variable* `counter` *to your class.*

The output of your program should look similar to the following.

```
Testing a small array: 1 3 6 7 7 7 9
Key 0 not found, should be at index 0, after 4 binary search iterations.
Key 1 found at index 0, after 4 binary search iterations.
Key 7 found at index 3, after 5 binary search iterations.
Key 10 not found, should be at index 7, after 5 binary search iterations.

Testing a large array with 10000000 entries: 0 2 4 6 ... 19999998
Key 5000000 found at index 2500000, after 25 binary search iterations.
Key 5000001 not found, should be at index 2500001, after 25 binary search iterations.

Testing a large array with 10000000 identical entries: 10 10 10 ... 10
Key 10 found at index 0, after 25 binary search iterations.
Key 9 not found, should be at index 0, after 25 binary search iterations.
Key 11 not found, should be at index 10000000, after 26 binary search iterations.
```

Optional exercise: Write a method `insert(key,x)` which uses `binsearch` to insert the value `key` at the correct position into the ordered array `x`. Also test this method in `main`.

Please submit your solutions on moodle before the lecture on **24 October 2018**.
Include your **name** and **student number** in the java file you submit. Good luck!