*All your algorithms must be written in pseudo code, and justified.  
A comparison will be considered as an elementary operation in O(1)*

1. Let a[] be a vector of size n and i an index between 0 and n-1. Propose an algorithm in O(n) time in order to count the number of elements greater than a[i] in a[]. **/1**
2. Let a[] be a vector. Propose an algorithm in order to compute the upper median of a[] (i.e., the element a[i] such that exactly half of the elements of the vector are bigger than it). The algorithm needs not be deterministic (i.e., it can be randomized). **/1**
3. Recall the definition of a balanced binary search tree. **/1**
4. Let a[] be a vector of size n. Propose an algorithm in O(nlog(n)) time in order to insert all the elements of a[] in a balanced binary search tree. **/1**
5. Show that any algorithm in order to insert the elements of a vector in a (not necessarily balanced) binary search tree requires at least O(nlog(n)) time. **/1**
6. Let a[] be a vector of size n. Propose an algorithm in expected O(n) time in order to decide the maximum number of occurences (= repetitions) of an element in the vector (if the algorithm proposed is correct, but slower, you get 1pt). **/2**
7. Let a[] be a vector of size n. Propose an algorithm in O(nlog(n)) time in order to decide, for each index i between 0 and n-1, the number of indices j < i such that a[i] > a[j]. (if the algorithm proposed is correct, but slower, you get 1pt). **/2**
8. Is this optimal? **/1**