Important: Be clear and precise.

We recommend that you do these problems on your own. Do not look for answers elsewhere and do not discuss these with anybody. In case you already know the answer mention this in your submission. If you do discuss the problem with somebody by accident or otherwise mention this too in your submission. You get half the marks for attempting most questions, so attempt each one.

You get points for what your TA makes of what you have written. Make reasonable assumptions when necessary, though we reserve the right to determine if they were indeed reasonable. If you write many answers to the same question, only the first will be graded. You can use any theorem/algorithm done in class. You have to describe the algorithm in English. You may then give pseudocode. However there must be enough detail in the description so that if we wish to write code we can. Writing code is optional, but writing only code will not fetch you any marks. The first 3 problems are to be submitted separately, stapled A4 sized sheets: HW1-A. The rest form HW1-B and are to be submitted separately. Write your roll number on both. In case running times are not explicitly mentioned in the question, any polynomial in the input size is all right.

1. You are given as input a sorted array A of n elements. You are also given a number M which exists in A. Your objective is to find the index of M in A. However, the array has the property that as soon as you probe three array elements with value at least M the array resets to all zeroes and you will not be able to finish your task. However you may probe as many elements of value strictly less than M.

   Find the index of M using the minimum number of probes. Full marks if you make $o(n)$ probes. Note that $g(n)$ is $o(n)$ if $\lim g(n)/n$ goes to zero as n goes to infinity.

2. Given an array with n elements. Determine if there is an element in the array which occurs greater than $n/2$ times using only equality tests between elements. That is, you are only allowed to test if two elements of the array are equal or unequal.

   For full credit $O(n)$ tests. Half credit $O(n \lg n)$ tests.

3. Given two sorted arrays of size n each, and a number k, show how to determine an element of rank k among the elements of the two arrays combined together using $O(\log k)$ comparisons.

   The rank of an element is its position in the sorted array.

4. Given an array A of size n, design an $O(n)$ time algorithm to find, for each entry of the array, the first number to its right which is less than it.

   That is, for each entry $A[i]$, the smallest index $j > i$ such that $A[j] < A[i]$.

   Note that in some cases it may not exist.

5. Your instructor, had a list of your roll numbers (you may assume they were 1 through n) sorted based on your rank in class (assume that ranks were distinct.) This was stored in an array B. For keeda-value your instructor further computed an array A, where $A[i]$ stored the number of students whose rank was greater than i and whose roll numbers

were less than the student who got the ith rank. As fate would have it, the array B was deleted accidentally. Give an $O(n \log n)$ algorithm to compute the array B given A. First give an $O(n^2)$ algorithm.

Note that the array B is essentially a permutation of the numbers $1, \ldots, n$.