- The instructions are the same as in Homework-0, 1, and 2.

There are 5 questions for a total of 100 points.

---

1. (15 points) Solve the following recurrence relations.
   (a) $T(n) = 3T(n/3) + cn$, T(1) = c
   (b) $T(n) = 3T(n/3) + cn^2$, T(1) = c
   (c) $T(n) = 3T(n-1) + 1$, T(1) = 1

2. (15 points) An array $A[1...n]$ containing $n$ integers is said to be a *hill-array* iff
   1. $n \geq 3$,
   2. $A[1], A[2], ..., A[n]$ are distinct, and
   3. there is an index $1 < i < n$ such that

   $$A[1] < A[2] < ... < A[i-1] < A[i] > A[i+1] > A[i+2] > ... > A[n]$$

   and such an index $i$ is called the *peak-index*.

   Design an algorithm for finding the peak-index of any given hill-array. You are given as input a hill-array $A$ and the size $n$ of the array. Give a time analysis for your algorithm and a brief explanation for correctness.

3. (20 points) Consider the following problem: You are given a pointer to the root $r$ of a binary tree, where each vertex $v$ has pointers $v.lc$ and $v.rc$ to the left and right child, and a value $Val(v) > 1$ . The value NIL represents a null pointer, showing that $v$ has no child of that type. You wish to find the path from $r$ to some leaf that minimises the product of values of vertices along that path. Give an algorithm to find the minimum product of vertices along such a path along with a proof of correctness and runtime analysis.

4. (20 points) Let $S$ and $T$ be sorted arrays containing $n$ elements. Design an algorithm to find the $n^{th}$ smallest element out of the $2n$ elements in $S$ and $T$. Discuss running time, and give proof of correctness.

5. An array $A[1...n]$ is said to have a majority element if more than half (i.e., $> n/2$) of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is $A[i] \geq A[j]$?" (Think of the array elements as GIF files, say.) However you can answer questions of the form: "is $A[i] = A[j]$?" in constant time.
   (a) (15 points) Show how to solve this problem in $O(n \log n)$ time. Provide a runtime analysis and proof of correctness.
   (*Hint: Split the array A into two arrays A1 and A2 of half the size. Does knowing the majority elements of A1 and A2 help you figure out the majority element of A? If so, you can use a divide-and-conquer approach.*)
   (b) (15 points) Design a linear time algorithm. Provide a runtime analysis and proof of correctness.
   (*Hint: Here is another divide-and-conquer approach:*

- *Pair up the elements of $A$ arbitrarily, to get $n/2$ pairs (say $n$ is even)*
- *Look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them*
- *Show that after this procedure there are at most $n/2$ elements left and that if $A$ has a majority element, then it's a majority in the remaining set as well)*