**Problem 1**

Consider a $k$-digit decimal counter, with initial value 0. Suppose that $n$ additions to the counter of numbers of the form $10^i$ are performed (i.e., each of the $n$ addition operations specifies an exponent $i$ between 0 and $k - 1$). Results are to be computed modulo $10^k$, in case of overflow. Using a credit or potential function argument, show that the total number of digits that change is at most $(10/9)n$.

**Problem 2**

As in class, let's consider a binary counter, now however it costs $2^k$ to flip the bit $A[k]$. In a sequence of $n$ increments, what is the worst case cost of a single increment? Show that the amortizing cost of a single increment is $O(\log n)$.

**Problem 3**

You are given an array $A[1, \ldots, n]$ of items. An item $x$ is called a *majority element* if it appears more than $n/2$ times in $A$. Clearly, a majority element exists, it must be unique. You are to design an algorithm that determines if $A$ has a majority element and if so, determines its value. Your algorithm should observe the following restrictions:

- The array $A$ is stored in *read-only memory*.
- Your algorithm may use at most $O(\log n)$ words of additional memory (each work can hold numbers bounded by $O(n)$).
- Your algorithm should be an index into $A$ that indicates a position where the majority element is located
- Your algorithm should be deterministic
- Your algorithm should run in time $O(n \log n)$.

**Study material**

Study, study study. Do problems in CLRS. Check out the old core exams and old PhD qualifiers for more problems.