1. I claim that in *any* intermixed sequence of Inserts and Deletes, starting with a New array-list, the amortized cost of each Insert and Delete is $O(1)$. What this claim *means* is that the total time required to execute *any* intermixed sequence of N_i Inserts and N_d Deletes is at most $O(N_i + N_d)$.

   One way to prove this claim is partition the overall running time of the data structure into *phases* immediately after each Resize, as indicated in the pseudocode in the homework handout, and analyze each phase separately.

   There are two cases to consider, depending on whether the phase ends by doubling or halving the data array. Let n_0 denote the value of AL.num at the start of a phase; to avoid trivial boundary cases, assume $n_0 \geq 4$.

   (a) Suppose the phase ends by doubling the data array. What is the *exact* minimum number of Insert and Delete operations that phase can contain? Your answer should be a function of n_0.

   Solution: At the start of each phase, we have $n_0 = AL.\text{num} = AL.\text{cap}/2$. The capacity AL.cap does not change until Resize doubles the array and ends the phase. Thus, just before Resize doubles the array, we have $AL.\text{num} = AL.\text{cap} = 2n_0$. The number of items in the array-list has increased by n_0, so we must have performed at least n_0 Inserts. (There are also trivially at least 0 Deletes, and therefore at least n_0 operations overall.) ■

   (b) Suppose the phase ends by halving the data array. What is the *exact* minimum number of Insert and Delete operations that phase can contain? Your answer should be a function of n_0.

   Solution: At the start of each phase, we have $n_0 = AL.\text{num} = AL.\text{cap}/2$. Just before Resize halves the array, we have $AL.\text{num} = AL.\text{cap}/4 = n_0/2$. The number of items in the array-list has decreased by $n_0/2$, so we must have performed at least $n_0/2$ Deletes. (There are also trivially at least 0 Inserts, and therefore at least n_0 operations overall.) ■

   (c) Complete the amortized analysis: Prove that the total time to execute any phase containing N_i Inserts and N_d Deletes is at most $O(N_i + N_d)$.

   Solution: Ignoring calls to Resize, each Insert and Delete requires $O(1)$ time, and the final call to Resize requires $O(n_0)$ time. So the total running time for the entire phase is $O(N_i + N_d + n_0)$.

   - If the phase ends by doubling the array, we have $n_0 \leq N_i$, so the total time for the phase is at most $O(2N_i + N_d) = O(N_i + N_d)$.
   - If the phase ends by halving the array, we have $n_0 \leq 2N_d$, so the total time for the phase is at most $O(N_i + 3N_d) = O(N_i + N_d)$.

   We conclude that each Insert and Delete *within any particular phase* runs in $O(1)$ amortized time. But every Insert and Delete is in *some* phase! ■

2. **Think about this one on your own; do not submit solutions.**

   Suppose we are allowed to maintain a tighter invariant $AL.cap \leq (1+\varepsilon)n$, for some fixed constant $\varepsilon > 0$. We still want to keep the amortized time for each INSERT and DELETE as small as possible.

   (a) How would you change the INSERT, DELETE, and RESIZE algorithms?

   Solution:

   - INSERT still triggers a RESIZE when $\mathsf{num} \geq \mathsf{cap}$.
   - DELETE triggers a RESIZE when $\mathsf{num} \leq (1-\varepsilon)\mathsf{cap}$.
   - RESIZE copies the data into an array of size $\lceil \mathsf{num} \cdot (1+\varepsilon/2) \rceil$.

   ■

   (b) What is the amortized time for each INSERT and DELETE, as a function of ε? (You should see a tradeoff between space and time; the amortized time for each operation should increase as ε approaches zero.)

   Solution (sketch): Following the analysis in problem 1, we partition the sequence of operations into phases, each ending immediately after a RESIZE.

   Let n_0 denote the value of $AL.\mathsf{num}$ at the start of a phase. Ignoring floors and ceilings, the phase contains at least $n_0\varepsilon/2$ operations, and the overall time for the phase is $O(n_0)$. So each operation takes $\boldsymbol{O(1/\varepsilon)}$ amortized time. ■