Consider the array doubling problem covered in class and shown in some of the references. It is shown that the amortized cost of an INSERT is 3 if the array-doubling cost charges only for the cost of copying all elements from the original array to the larger one. That is, if an INSERT is performed at location \( i \), where \( i = 2^k + 1 \) (for some \( k \)), then the cost is \( i - 1 \) for the copying and then 1 for the write at location \( i \).

On the other hand, we saw in class that the amortized cost is 5 if the array doubling is charged for (i) copying the \( i - 1 \) elements into the left side of the new array (locations 1, \ldots, \( 2^k \)) and (ii) initializing the right side (locations \( 2^k + 1, \ldots, 2^{k+1} \)) to zero. Total cost, including the write of the new element, is \((i - 1) + (i - 1) + 1 = 2i - 1 = 2^{k+1} + 1\).

For this problem, we modify the array doubling cost to charge for (i) an initialization of the whole array (set locations 1, \ldots, \( 2^{k+1} \) to zero) and then (ii) copying the smaller array into the larger one (locations 1, \ldots, \( 2^k \)).

For this problem, you should carefully derive the amortized cost of an INSERT operation using each of the given techniques.

1. aggregate (also called brute force)
2. accounting (aka banker’s, taxation)
3. potential (aka physicist’s)

Notes:

- The amortized cost should be 7 per insertion (I think!).
- The arrays here are indexed starting at 1, rather than the first location being 0. Sorry.