Part 2: Amortized Analysis of the dynamic Array

1. How many cost units are spent in the entire process of performing 32 consecutive push operations on an empty array which starts out at capacity 8, assuming that the array will double in capacity each time a new item is added to an already full dynamic array? As N (ie. the number of pushes) grows large, under this strategy for resizing, what is the big-oh complexity for a push?
   1. The total cost units for making it is 58 steps.
   2. The big-oh complexity O(1)+
2. How many cost units are spent in the entire process of performing 32 consecutive push operations on an empty array which starts out at capacity 8, assuming that the array will grow by a constant 2 spaces each time a new item is added to an already full dynamic array? As N (ie. the number of pushes) grows large, under this strategy for resizing, what is the big-oh complexity for a push?
   1. The total cost units for making it is 260 steps.
   2. The big-oh complexity O(n).
3. Suppose that a dynamic array stack doubles its capacity when it is full, and shrinks (on Pop only) its capacity by half when the array is half full or less. Can you devise a sequence of N push() and pop() operations which will result in poor performance (O(N^2) total cost)? How might you adjust the array's shrinking policy to avoid this? (Hint: You may assume that the initial capacity of the array is N/2.)
   1. One method to get (O(N^2) would be: Push() values to the stack until you reach capacity, that causes the capacity to double. Then, pop() a value from the stack, to cause the capacity to be cut in half.
   2. You can adjust the array’s shrinking cost to make it only shrink when the array is at one-fourth capacity.