Name: Tatyana Vlaskin
Date: 10/17/2014
Amortized Analysis

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 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?

To add 1st 8 elements it costs 8 units
To add 9th element, it costs 9 units
To add next 7 elements (10 – 16), it costs 7 units
To add 17th element it costs 17 units
To add elements 18 -32, it cost 15 units
Total cost to push 32 elements = 8 + 9+ 7+ 17+ 15 = 56 units
As the number of pushes tends to infinity, the vast majority of pushes will take 1 cost unit, thus, the overall big-O complexity is 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 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?

8 items	**8**
add 2, make 10	**0**
copy 8 add 2	**10**
add 2, make 12	**0**
copy 10, add 2	**12**
add 2, make 14	**0**
copy 12, add 2	**14**
add 2, make 16	**0**
copy 14, add 2	**16**
add 2, make 18	**0**
copy 16, add 2	**18**
add 2, make 20	**0**
copy 18, add 2	**20**
add 2, make 22	**0**

For 32 total push operations, there are 260 cost units.
As the number of pushes goes to infinity, we need to keep adjusting the capacity infinitely, thus the big O complexity is 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.)

Since, we are starting with the initial capacity of N/2, the poor sequence will be:
1st push – this will double the capacity and then follow by a pop()- this will reduce the capacity. And then repeat a push() and a pop(). This will result in O{push()pop()}, and change in the capacity back to back, thus, we have (O(N^2)) total cost.

To avoid this, you can make the array to shrink its capacity by half when the array is less than half full, like when the size is ¼ or 1/6, etc.