**Question 1. Aim of this question is to understand amortized cost analysis.**

**Sample Instance 4 of Clearable Table**

add, add, add, add, clear, add, clear add, clear, add, add, add, clear, add, add, clear, add, add, clear, add, add, clear.

1. Show all the calculations:
2. add 1 element -> cost 1
3. add 2 element -> cost 1
4. add 3 element -> cost 1
5. add 4 element -> cost 1
6. clear 4 element -> cost 4
7. add 1 element -> cost 1
8. clear 1 element -> cost 1
9. add 1 element -> cost 1
10. clear 1 element -> cost 1
11. add 1 element -> cost 1
12. add 2 element -> cost 1
13. add 3 element -> cost 1
14. clear 3 element -> cost 3
15. add 1 element -> cost 1
16. add 2 element -> cost 1
17. clear 2 element -> cost 2
18. add 1 element -> cost 1
19. add 2 element -> cost 1
20. clear 2 element -> cost 2
21. add 1 element -> cost 1
22. add 2 element -> cost 1
23. clear 2 element -> cost 2

Total costs = 1 + 1 + 1 + 1 + 4 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 3 + 1 + 1 + 2 + 1 + 1 + 2 + 1 + 1 + 2 = **30**

Number of operations = **22**

Average cost per operation = 30/22 = 1.36 < 2.

**Sample Instance 3 of ArrayList with size doubling strategy**

A resize just happened from size 16 to size 32.

1. Show all the calculations:

That means, we have 16 free spaces in your newly created ArrayList.

add add add add add add add add add add add add add add add add rezise

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3\*size = 3\*32 = 96.

Total cost = 16 + 96 = 112.

We want to distribute the total cost of 96 among previous 16 adds.

Amortized\_Cost(add) = 112/16 = 7.

1. Amortized\_Cost(resize) = 0.

**Question 2. Aim of this question is to better understand amortized cost analysis.**

**Data structure: ArrayList with size tripling strategy.**

1. We assume that there are **K** and resize from **K** to size **3K** (size tripling strategy)
2. **a. What is the actual cost of add?**
   * + 1. The cost of add are **2K**
3. **b. What is the actual cost of resize?**
4. Cost of resize = **3 \* 3K** to create a new array and **+ 3K** (copy **3K** items from the old array to new array) = **12K**
5. Thus, the total cost = **2K** (cost of adds) + **12K** (cost of resize) = **14K**

**3. Using traditional worst-case analysis, show that the average cost of an operation is NOT constant time.**

The average cost of an operation = **14K/2K**=7

* 1. **d. Consider a sample instance (hint : resize just happened and current size of the array is 9. (You should never consider current size = 1 for this type of calculation). You are adding. Then you resized again)** 
     + 1. **- What is the Amortized\_Cost(add)?**
       2. Cost of add = 6
  2. **- What is the Amortized\_Cost(resize)?**
  3. Cost of resize = 6 + 2 \* 9 + 2\*9 (copy item from old array to new array) = 42
  4. **- Through amortized cost analysis show if there is sequence of n operations (some add, some resize) the average cost of an operation is constant time.**
  5. Average cost of operation = 42/6 = 7