Add	1	2	3	Avg
ArrayList	4974300	5389800	6450700	5604933.33
LinkedList	76200	282600	242900	200566.667
Sort	1	2	3	Avg
ArrayList	4166300	3756700	3889500	3937500
LinkedList	2144300	2141700	3360900	2548966.67
Shuffle	1	2	3	Avg
ArrayList	934700	1150900	2193400	1426333.33
LinkedList	346000	2468300	685500	1166600
get Random	1	2	3	Avg
ArrayList	29305200	26553200	25788500	27215633.3
LinkedList	596546300	619173200	609426200	608381900
get Sequential	1	2	3	Avg
ArrayList	76200	282600	242900	200566.667
LinkedList	694700	1409700	673400	925933.333

a) summarize which implementation appears to be better for each kind of operation & discusses how those results are consistent with how the list is implemented.

I highlighted in light green the significant numbers in LinkedList. I found that when it came to adding LinkedList it was faster and more efficient. This makes sense as ArrayList does have to create a new array every time an index or piece is added because it is not mutable. LinkedList is far more efficient at adding and removing items. However, the Add operation nanoseconds from LinkedList matched the Arraylist get Sequential operation nanoseconds. This exact time matching tells me they are equally efficient if the correct algorithm is used.

Further, LinkedList performed faster in Sequential because LinkedList is faster and knows its next item in order. This was a quick process to achieve. However, the Random operation for LinkedList performs poorly because they are stored separated in memory and so retrieving this information in memory takes slower.