1. Add Method:

Analysis:

This method is supposed to make the array bigger depending on the fact weather the current array is filled (if statement) . How? It multiplies the number of elements of the array times 2 and adds one; So: ( O(2\*n+1) but since they are constants O(n) ).

After assigning the new size of the array it the runs a for loop that copies the elements in data[] to biggerArray[]. After all of this it will just increase manyIteams once.

Answer: Well I am not sure if this is correct but the big O complexity of the size on the array by which it is beaing increased by multiplying it by two and since two is a constant. But if we are just talking about by how much the array is being added then it would just be one which is also a constant.

n = manyIteams.

Worst Case: Array is filled therefore array gets multiplied by 2 and allocates more space.

Big O: O(n)

1. Counting Occurrences:

Analysis:

This method is supposed to return the number of times which a particular element is fount in the array of data[]. How? With a for loop it goes through the entire array as long as its less than manyItems witch is in a way our “true size of the array”;

After that, every time it sees that that element is there then the counter increases.

Answer: Since the only thing that is beaing increase in this method is really our counter (answer) every time that it find that element it would just be increasing by one, and one is a constant.

n = manyIteams

Worst Case: All the elements in data[] are the target and since the for loop is runing for n.

Big O: O(n)

1. List Position:

Analysis:

This method is supposed to return where exacly the position in which the linked list has that specific position. How? This is method first ensures that what ever input is entered it is not less than equal to 0 (if so it throws an exception). Then it will run a for loop which runs as long as our cursors is less than the position we are looking for, the other instance is that cursors is not equal to null.

After that for every iteration our counter (Answer) increases once, and then it returns answer.

n = position

Worst Case: The method runs until cursor reaches null.

Not only that but then it would return an inaccurate counter (answer) maybe placing an if statement that if its equal to null then it sure return - 1. (this is because it would still return the number of iterations the for loop went through)

Big O: O(n)

1. List Length:

Analysis:

This method is supposed to calculate the number of nodes in the linked list. How? This is done with a node that starts from the head which is one of the parameters provided when calling the method. From that head the for loop will be running until it is no longer pointing to null. Every iteration the counter (Answer) is increased.

After that the method returns the counter “answer”.

n = cursor

Worst Case: The linked list has a lot of elements?

Big O: O(n)

* I apologize I was really unsure how to do this lab I simply tried my best to describe what the method did and what the Big O notation was.