1. Suppose we have an implementation of ArrayList that doubles the capacity when the array fills. Suppose further that, when initialized, the array starts with capacity 1. Show the sizes and growth of the array when the following steps are implemented.

```
ArrayList<Integer> a = new ArrayList<>();
    a.add(3);
    a.add(-1);
    a.add(1, 2);
    a.add(3, 0);
    a.add(4);
    a.add(1, 5);
```

Answer:

When initialized, the array's size is 0, the array's capacity is 1.
After a.add(3), the array's size is 1, the array's capacity is 1.
After a.add(-1), the array's size is 2, the array's capacity is 2.
After a.add(1, 2), the array's size is 2, the array's capacity is 2.
After a.add(3, 0), the array's size is 4, the array's capacity is 4.
After a.add(4), the array's size is 5, the array's capacity is 8.
After a.add(1, 5), the array's size is 5, the array's capacity is 8.

2. Suppose you have data that you would like to sort using bubblesort. In terms of running time, does it make any difference whether that data is stored in an ArrayList or in a doubly LinkedList? Explain.

Answer:

In terms of running time for the bubblesort, it does not make any difference whether that data is stored in an ArrayList or in a doubly LinkedList. Because for each iterator during the bubblesort, both the ArrayList and the doubly LinkedList can access the current data, the previous data and the next data with the O(1) time complexity, so that the time complexity of the bubblesort for the data store in an ArrayList and a doubly Linked List is both $O(n^2)$.

3. Suppose that you have data that is stored in a List and that the data is already sorted. You would like to search the array to see if it contains an element. In terms of running time, does it make any difference whether the data is stored in an ArrayList or in a doubly Linked List? Explain.

Answer:

In term of running time for search the sorted data array to see if it contains an element, it will make difference between the ArrayList and the doubly Linked List. Because the doubly Linked List can not access any data in the list with O(1) time complexity, so that we can only use the linear search algorithm to search the data, and the time complexity when search the data in the doubly LinkedList is O(n). But the ArrayList can access any data in the list with O(1) time complexity, so that we

can use the binary search algorithm to search the data, and the time complexity when search the data in the ArrayLinkedList is $O(\log_2 n)$, which is difference from the doubly LinkedList's search time complexity.

4. Show what happens when the following operations are performed on a Queue. Don't worry about the specific implementation of the Queue, but do show which elements are present (and in which order) and which elements are removed.

```
Queue q = …
q.enqueue(0);
q.enqueue(4);
q.enqueue(-1);
q.dequeue();
q.enqueue(0);
q.dequeue();
q.enqueue(7);
q.dequeue();
q.dequeue();
```

Answer:

When initialized, the queue is empty.
After q.enqueue(0), the elements in the queue is: 0.
After q.enqueue(4), the elements in the queue is: 0, 4.
After q.enqueue(-1), the elements in the queue is: 0, 4, -1.
After q.dequeue(), the elements in the queue is: 4, -1.
After q.enqueue(0), the elements in the queue is: 4, -1, 0.
After q.dequeue(), the elements in the queue is: -1, 0.
After q.enqueue(7), the elements in the queue is: -1, 0, 7.
After q.dequeue(), the elements in the queue is: 0, 7.
After q.dequeue(), the elements in the queue is: 7.