1. Difference Between Array Size and Capacity (0.1 pts)

The size of an array refers to the number of elements currently stored in it, while the capacity is the total number of elements the array can hold before needing to resize.

2. What Happens When an Array Needs to Grow Beyond Its Capacity?

When an array reaches its maximum capacity, it must be resized to accommodate more elements. The way this happens depends on the memory availability.

Case 1: Space is Available After the Array (0.1 pts)

If the memory right after the array is free, the array can simply expand in place without moving.

Before Expansion:

```
Memory Layout:
[ 10 ] [ 20 ] [ 30 ] [ 40 ] [ 50 ] [ ?? ] [ ?? ] [ ?? ] ...
```

After Expansion (Adding More Elements):

```
[ 10 ] [ 20 ] [ 30 ] [ 40 ] [ 50 ] [ 60 ] [ 70 ] [ 80 ] ...
```

- The memory location remains the same.
- Only the size increases.

Case 2: Memory After the Array is Occupied (0.2 pts)

If another variable occupies the space after the array, a new memory block must be allocated, and the data must be copied over.

Before Expansion (Occupied Memory):

```
Memory Layout:
[ 10 ] [ 20 ] [ 30 ] [ 40 ] [ 50 ] [ Other Data ] [ Other Data ] ...
```

- The array cannot expand in place.

After Expansion (Relocated):

A new, larger array is allocated in a different memory location, and the contents are copied.

```
New Memory Block:
[ 10 ] [ 20 ] [ 30 ] [ 40 ] [ 50 ] [ 60 ] [ 70 ] [ 80 ] ...
```

- The old array is deleted, and the pointer is updated.
- This process is costly due to copying operations.

3. Discuss one or more techniques real-world array implementations use to amortize the cost of array expansion

Since copying data to a new memory block is expensive, real-world implementations use techniques to reduce the cost of expansion:

1. Doubling Strategy: Instead of increasing the array size by one element, it doubles the capacity each time it grows.
    - Example: If an array of capacity 4 is full, it resizes to 8 instead of 5.
2. Geometric Growth: Some implementations increase size by a factor (e.g., 1.5× instead of 2×) to balance memory use and performance.
3. Preallocation: Some programming languages (like Java's `ArrayList`) allow setting an initial capacity to avoid frequent resizing.