Sorting

1. **Bubble Sort**
    1. In an unsorted array of 5 elements, start with the first two elements and sort them in ascending order. (Compare the element to check which one is greater).
    2. Compare the second and third element to check which one is greater, and sort them in ascending order.
    3. Compare the third and fourth element to check which one is greater, and sort them in ascending order.
    4. Compare the fourth and fifth element to check which one is greater, and sort them in ascending order.
    5. Repeat steps 1–5 until no more swaps are required.
2. **Selection Sort**
    1. Get a list of unsorted numbers
    2. Set a marker for the unsorted section at the front of the list
    3. Repeat steps 4 6. until one number remains in the unsorted section
    4. Compare all unsorted numbers in order to select the smallest one
    5. Swap this number with the first number in the unsorted section
    6. Advance the marker to the right one position
    7. Stop
3. **Merge Sort**
    1. Divide by finding the number q q qq of the position midway between p p pp and r r rr. Do this step the same way we found the midpoint in binary search: add p p pp and r r rr, divide by 2, and round down.
    2. Conquer by recursively sorting the subarrays in each of the two subproblems created by the divide step. That is, recursively sort the subarray array[p..q] and recursively sort the subarray array[q+1..r].
    3. Combine by merging the two sorted subarrays back into the single sorted subarray array[p..r].
4. **Quick Sort**
    1. Find a “pivot” item in the array. This item is the basis for comparison for a single round.
    2. Start a pointer (the left pointer) at the first item in the array.
    3. Start a pointer (the right pointer) at the last item in the array.
    4. While the value at the left pointer in the array is less than the pivot value, move the left pointer to - the right (add 1). Continue until the value at the left pointer is greater than or equal to the pivot value.
    5. While the value at the right pointer in the array is greater than the pivot value, move the right pointer to the left (subtract 1). Continue until the value at the right pointer is less than or equal to the pivot value.
    6. If the left pointer is less than or equal to the right pointer, then swap the values at these locations in the array.
    7. Move the left pointer to the right by one and the right pointer to the left by

one.
    8. If the left pointer and right pointer don’t meet, go to step 1.
5. **Insertion Sort**

6. **Counting Sort**