Question 1: Ignore

Question 2. Use 5 operations to sort 4 elements a, b, c, d

```
Compare a > b {
      First_max = a
      First_min = b
} else {
      First_max_ b
      First_min = a
}
Compare c > d{
      second_max = c;
      second_min = d
} else {
      Second_max = d;
      Second_min = c;
}
Compare first_max > second_max {
      Maximum = first_max;
      First_mid = second_max;
} else {
      Maximum = second_max;
      First_mid = first_max;
}
First_min < second_min {
      Minimum = first_min;
      Second_mid = second_min
} else {
      Minimum = second_min;
      Second_mid = first_min;
}
if(first_mid > second_mid) swap(first_mid, second_mid)
The order is increasing order: minimum, first_mid, second_mid, maximum
```

It doesn't violate the lower bound because in the theorem we ignore a lot of constants. We can't compare complexity with number like that

Question 3

```
Algorithm FBSArray(arr: Array){
      Sort the input array
      i <- 0
      N <- arr.length
      answer = [] * N
```

```
      J <- 0
      For i in 0..N - 1 in every even position
            answer[i] = arr[j]
            j <- j + 1
      i <-  last not set position
      Assert (i % 2 == 1)
      While i >= 0
            answer[i] = arr[j]
            j--
            i -= 2
      Return answer
}
```

Because every odd position is filled later than positions at even positions so every number in odd positions is bigger than every position in even positions. And thanks to the fact that we filling the answer array like that all number in even positions are increasing and all number in odd positions are decreasing.

The total time complexity is O(NLogN) because we have to sort the array.