Week 6:

1a) Bubble sort:

```
n ← array.length
do
    swapped ← false
    for i ← 1 to n-1 (inclusive)
        if array[i-1] ~~> array[i] then
            temp = array[i-1]
            array[i-1] = array[i]
            array[i] = temp
            swapped ← true
        endif
    endfor
    n = n-1
while (swapped == false)
```

b) Best case is when ~~they are~~ array is sorted, meaning there are n comparisons: $\Omega(n)$.

Worst case is when array is not sorted ~~and~~ so, there are n comparision for n ~~positi~~ positions, hence $O(n^2)$.

Average case is if half the elements need to be swapped, so; $\Theta\left(\frac{n}{2}(n-1) \cdot \frac{1}{2}\right) = \Theta(n^2)$

c) Insertion sort is stable, as elements are in order & will not be swapped.

Bubble sort is
Stable because if they are ordered, they will not be swapped.

Heap sort is unstable as even if elements are in order, they might be swapped.

Merge sort is stable when we use Left ≤ Right.

d) Insertion sort is adaptive as best case $\Theta(n)$.
Bubble sort is adaptive as best case $\Theta(n)$
Merge & Heap Sort are not adaptive as ~~both~~ worst best is $\Theta(n \lg n)$

2.

a) & b) In Heap q2.cpp.

c) As seen in graph time.png, Bottomup takes less time, but they have same complexity.