Homework 6

a. Can be seen in bubblesort.cpp.

b. Best case

Best case is in case of a sorted array which would mean that the loop would run once making the complexity $\theta(n)$.

Worst case

worst case would be a reverse order array. this means that both loops would have to run making the complexity $\theta(n \cdot (n-1)) = \theta(n^2)$

average case

=> for the average case, the effort would be to swap half of the elements and so $\theta\left(n \cdot \frac{(n-1)}{2}\right) = \theta(n^2)$ complexity.

c. insertion sort

=> stable because if elements are ordered, they won't be swapped.

Merge sort

In the case of merging the arrays at the end we ensure that $L \leq = R$ and if this is the case, the algorithm is stable if not it will be unstable.

Bubble sort

It is stable because if the elements are already ordered the swapping does not take place.

Heap sort

It is unstable because even if the elements are ordered they might be swapped.

Insertion sort

d) It is adaptive as best case $\theta(n)$ and worst case is $\theta(n^2)$

Merge sort

It is non-adaptive since the worst case and the best case are the same $\theta(n \lg n)$ complexity.

Heapsort

non-adaptive because best case and worst case yields $\theta(n \lg n)$ complexities.

Bubble sort

Adaptive because worst case is $\theta(n^2)$ while best case is $\theta(n)$

6.2

a. Can be seen as "Heapsort6.cpp"

b. Can be seen as "bottomup heapsort.cpp".

c. After a certain high number of elements n, the variant (bottom up heapsort) takes less time.