P6.1

a) BubbleSort (A)

```
swapped = true
// itterate while swap is available
while true do
// set false as there is no swap
    swapped = false
    // itterate through the array
    for i = 1 to A.lenght do
        if A[i] > A[i+1] then    // check if first is bigger than second
            Swap [A[i], A[i+1]]  // swap them
            swapped = true
        end if
    end for
  end while.
```

b) Worst-case

Worst-case of this algorithm happens when the array is sorted in reverse order, which makes while loop run $n-1$ times and the inner loop n times which makes $\Theta(n \cdot (n-1))$ which is $\Theta(n^2)$.

Average-case

For Average-case half of the elements needs to be swapped, so $\Theta\left(\frac{n(n-1)}{2} \times \frac{1}{2}\right)$ which is $\Theta(n^2)$

P.6.1 (b) Best Case

Best Case would happen where array is sorted already and loops only one time which makes our time complexity $\theta(n)$.

c) Insertion Sort

Stable sorting algorithm is element are in order they will not be swapped.

Merge Sort

When merging the two sorted subarrays, the elements of left subarray will be placed first in the merged array if they are smaller to the elements they are being compared to in the right subarray, which makes this algorithm stable.

Heap Sort

This is an unstable algorithm as even if the elements are in order they might be swapped.

Bubble Sort

This is a stable sorting algorithm, because if [illegible] elements are in order they will not be swapped.

P6. 1. ~~a~~

d) Insertion sort

It is adoptive as best case is $\Theta(n)$ and worst case is $\Theta(n^2)$

Merge Sort

It is not adoptive as best and worst case come out to be $\Theta(n \log n)$

Heap Sort

It is not adoptive as best and worst case is $\Theta(n \log n)$.

Bubble Sort

It is adoptive as best case has $\Theta(n)$ complexity, but worst case has $\Theta(n^2)$.