Name: Brian Sherif Nazmi

AaDS HW4

Problem 1:

a). a is an array of size n.

```
~~for (a=n;~~ while (array not sorted)
{
    for (i=0; i < n-1; i++)
    {
        if (current set "i" greater than
            next set (i+1)) ~~$~~ {
            Swap positions of sets
        }
    end for after iteration of whole array
end while after array is sorted
```

b.) While looking at the pseudocode for Bubble sort, we note 1 while loop and 1 for loop each iterating over all the elements.

This gives us 2 possible cases.

b.) case 1: Best case scenario. All items are already sorted giving only 1 iteration with the while loop.

Giving $\boxed{O(n)}$ where n is the number of inputs.

case 2: Average and worst case.

where worst case is an array sorted in reversed order. This means the algorithm will have to look at the whole array then look at n-1 elements

$O = n + n-1 + n-2 + \ldots + 1 = (n \times (n+1))/2$

$\frac{n^2+1}{2} = n^2 + \frac{1}{2}$ (constant ignored)

$\boxed{O(n^2)}$

Problem 1

c) Insertion sort : Stable algorithm

In insertion sort, we have a while loop condition. This condition prevents the algorithm from running over the start of the array. Another condition is used to insert the item in the correct spot. This two conditions enforce sort stability by checking both keys.

Merge sort: Stable algorithm

when merging the different sides; There is a condition L<=R to favor left side values over Right ones if they are equal

Heap sort: ~~Sta~~ Not-stable

The final sequence of the results from heapsort comes from removing the items from the heap in a size order.

This means ~~[illegible]~~ all order is lost in heap creation

Bubble sort: Stable

Due to if statement where condition states $L \leq R$ favoring the left over the right if items are equal.

d) Insertion sort: Adaptive

Best case senerio: $O(n)$ as it goes over all inputs

Worst & Average case: $O(n^2)$

Mergesort: Non Adaptive

All cases give: $\Theta((n \log(n))$

As the algorithm proceeds to split down the array then compare then merge.

Heap sort: Not Adaptive

The Second loop in heap sort "~~sinking~~" phase. gives a time of $O(n \log(n))$ time

Bubble sort: Adaptive: $O(n)$ when sorted as it passes thru the elements.

~~Non~~ Average time is $O(n^2)$

c) Comparing both Heap sort and variant heap sort.

~~Hea~~ Size of N	Heap Sort	V. Heap Sort
1000	0,00022	0,000187
10000	0,00197	0,001826
100 000	0,02149	0,021038
1000000	0,24785	0,274699

With smaller input V. Heap sort is faster

As N grows heap sort is faster