1.)

a.) public static void sort (double [] A)

```
while (A not sorted) {
    for (int i=0; i< A.length-1; i++){
    for(int j=0; j<A.length-1; j++) {
    if (A[i] > A[i+1]) {        //not swapped
        //swap A[i] and A[i+1]
    } }
    } }
}
```

b.) time complexity $= \theta(n^2)$

There are $\forall$ inputs of length n with $\forall$ lines of code $(n \geq 1)$ and each line runs at maximum $n^2$

i and j are variables in the loop and takes n values and each line runs once for every value.

A line is executed once for a value of i and for a value of j so it's total runs $n^2$ times.

$cn^2 = o(n^2)$

1.) c.) The inner loop starts with $j=0$ and checks if $A[j] < A[j+1]$. After each iteration, the $i-k+1$ elements are in the correct position if A were sorted nondescendingly. Now, prove loop invariant for $(k+1)$

if $A[j-1] \geq A[j]$ then $A[j-1]$ and $A[j]$ swap

→ inner loop starts with $j=k$, the smallest element and goes to $A[k]$.
A is an array that has a maximum value of $(k+1)$.
When the outer loop runs $(k+1)$ times, the last $(k+1)$ elements are sorted, so the loop invariant is true for $k+1$, and the induction is complete.

2.) The number of subsets of $\{1, 2, \ldots, n\}$ having an odd number of elements is $2^{n-1}$.

Base case: $n=1$

when $n=1$.

$2^{n-1} = 1$
$2^{1-1} = 1$
$2^{0} = 1$
$1 = 1$ ✓

Induction hypothesis: If $P(n)$ is true, then $P(n+1)$ is true

$2^{n-1}$ prove
$2^{(n+1)-1} = 2^{n}$

Induction step: Let A be a set with $n+1$ elements

$\underbrace{\text{set } A = A - \{a\}}$

has two groups: 1) subset containing a
2) subset not containing a

A' has n elements so $2^{n-1}$ even subsets and $2^{n-1}$ odd subsets are there in group 1. Group 2 is in the form $B = B' \cup \{a\}$ where $B'$ is a subset of $A'$ and is odd. With the induction hypothesis there is $2^{n-1}$ even subsets of A and $2^{n-1}$ odd subsets of A'.

(cont.)

(cont.)

$2^{n-1}$ odd subsets of group 2 and $2^{n-1}$ even subsets of group 2.

Because group 1 and group 2 have $2^{n-1}$ even subsets, A has $2^{n-1} + 2^{n-1}$ even subsets which equals $2^n$.

For odd subsets, A also has $2^{n-1} + 2^{n-1} = 2^n$.

A was a set with n+1 elements. Thus, a $(n+1)$ element set has $2^n$ even subsets and $2^n$ odd subsets.

(n-element set has $2^{n-1}$ even subsets and $2^{n-1}$ odd subsets
(n+1)-element set has $2^n$ even subsets and $2^n$ odd subsets)

Thus, $P(n+1)$ is true for all $n \in \mathbb{N}$.

3.) $f(n) = a_0 + a_1 n + a_2 n^2 + \dots a_k n^k$

Show that $f(n) \in O(n^k)$

$$\lim_{n \to \infty} \frac{(a_0 + a_1 n + a_2 n^2 + \dots a_k n^k)}{n^k} = \lim_{n \to \infty} \frac{a_k n^k}{n^k}$$

$$= a^k$$

Rule: if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = c$ and $c \neq 0$, then

$f(n) \in \Theta\, g(n)$ : $f(n) = g(n)$

$a^k$ is a constant which doesn't equal 0 so $f(n) \in O(n^k)$. ✓

Show that $f(n) \notin o(n^{k'})$ for all $k' < k$

$$\lim_{n \to \infty} \frac{(a_0 + a_1 n + a_2 n^2 + \dots a_k n^k)}{n^{k'}} = \infty$$

because the numerator $(n^k)$ is of higher degree than denominator $(n^{k'})$

Rule: if $\lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty$, then $f(n) = O(g(n))$ : $f \leq g$

$\infty$ is not less than $\infty$ so $f(n) \notin o(n^{k'})$.

4.) $\log_2 n = O(n^{1/10})$

$$\lim_{n\to\infty} \frac{\log_2 n}{n^{1/10}} = \frac{\frac{1}{n\ln 2}}{\frac{1}{10}n^{-\frac{9}{10}}} = \frac{\frac{1}{n\ln 2}}{\frac{1}{10n^{\frac{9}{10}}}} = \frac{10n^{\frac{9}{10}}}{n\ln 2}$$

$$= \frac{10}{n^{\frac{1}{10}}\ln 2}$$

$$\left(\lim_{n\to\infty} \frac{1}{n^{\frac{1}{10}}}\right) \cdot \frac{10}{\ln 2}$$

$$0 \cdot \frac{10}{\ln 2} = 0 \checkmark$$

because $\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ iff $f(n) = O(g(n))$.

$\log_2 n = \Omega(n^{1/10})$

$$\lim_{n\to\infty} \frac{\log_2 n}{n^{1/10}} = 0$$

but $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ has to be greater than 0 if $f(n) = \Omega(g(n))$, so this is false.

$\log_2 n = \Theta(n^{1/10})$

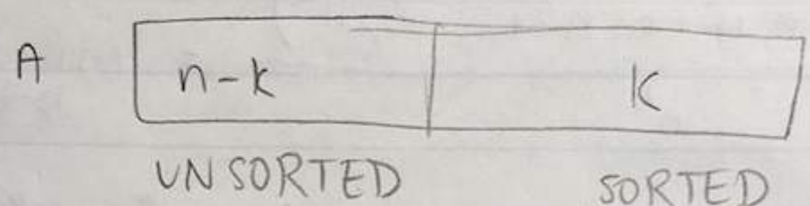
$$\lim_{n\to\infty} \frac{\log_2 n}{n^{1/10}} = 0$$

This is not true because the limit is 0 but the value cannot equal 0 if $f(n) = \Theta(g(n))$.

5.) array A = sorted except for k elements

↓

n-k elements are sorted, k are unsorted




Insertion sort only needs 1 comparison to check that an element is in the correct location for n-k sorted elements. The k elements could be in the sorted section and in the worst case they would be in the beginning of the sorted section. If this is the case, there would be $O(n)$ comparisons for the k elements, which leads to an overall runtime of $O(nk+n)$ which simplifies to $O(nk)$.