Q1)

	A	B	O	o	Ω	ω	Θ
a.	$\lg^k n$	n^ϵ	yes	yes	No	No	No
b.	n^k	c^n	yes	yes	No	No	No
c.	$\sqrt{n}$	$n^{\sin n}$	No	No	No	No	No
d.	2^n	$2^{n/2}$	No	No	yes	yes	No
e.	$n^{\lg c}$	$c^{\lg n}$	yes	No	yes	No	yes
f.	$\lg(n!)$	$\lg(n^n)$	yes	No	yes	No	yes

Q2)

a)

We know that insertion sort worst case runtime is $\theta(k^2)$ for k elements, and if we open this equation we get:

$$ak^2 + bk + c$$

and we know that there are n/k sub lists so we will iterate for

$$\frac{n}{k}(ak^2 + bk + c)$$

If we open it

$$a\text{kn} + bn + cn/k$$

from that equation we can easily state that runtime is $\theta(nk)$.

b)

To solve this question, first we need to come up with a recurrence relation for x sub lists of size k and which is:

$$T(x) = \begin{cases} 0, & x = 1 \\ 2T(x/2) + xk, & x = 2^p, for\ 0 < p \end{cases}$$

We can apply Master-Theorem to find runtime:

a = 2,

b = 2,

f(n)=k*n,

$n^{\log_b a} = n^{\log_2 2} = n$

From the book:

2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.

So, in that case, case 2 of master theorem works in our example because

$$\mathrm{f(n)} = \mathrm{kn}\ and\ f(n) = \theta(n) = \theta(n^{\log_b a}) = \theta(n^{\log_2 2}) = \theta(\mathrm{n})$$

so, as a result, master theorem states that if above equation holds, following is true:

$$\mathrm{T(n)} = \theta(n^{\log_b a} lgn) = \theta(\mathrm{n}lgn)$$

If we put the actual size of the sub list which is equal to $\frac{n}{k}$ to T(n) function it becomes:

$T(\frac{n}{k}) = \frac{n}{k} * \log\frac{n}{k}$ and it equals to $T(\frac{n}{k}) = \frac{n}{k} * \log\frac{n}{k}$ due to coefficient is not important in big theta notation, result becomes as:

$$T(\frac{n}{k}) = \theta(n * \log(n/k))$$

And the asked runtime $\boldsymbol{\theta(n * \log(n/k))}$ has been found with Master Theorem.

c)

question states that modified merge-sort works in $\theta(nk\ + \mathrm{nlog}(n/k))$ worst case. If we look at the equation, we can arrive that k = θ(lgn) because if it is not true, if k>lgn then nk > $\mathrm{nlog}(n/k)$ becomes true and it is unwanted case because this case means that we exceeded merge-sort runtime which is nlogn.

So, after explanation why k = logn, we can use it in the equation and equation becomes:

$\theta(nlogn\ + \mathrm{nlog}(n/logn))$ which is equal to $\boldsymbol{\theta(nlogn)}$ **and we can say that this running time is same with merge-sort.**

To keep it in that way, **k should be less than or equal to logn.**

if k>logn, k*n > n*logn and we can see that we exceeded runtime of merge-sort.

d)

To determine the value k, we have two options. First one is obtaining the runtime of merge-sort and insertion sort with their respective coefficients. In this manner, we can create an inequality and we can solve it for k. Second way is to test these two algorithms with different values of k. After enough experiment has done, we can find the value of k via comparing the runtime values of algorithm by k values.

As a result, we can find the k value by doing experiments.

Q3)

If we know that f(x) and g(x) are monotonically increasing functions, we can state that:

$$m \leq n \Rightarrow f(m) \leq f(n)$$

$$m \leq n \Rightarrow g(m) \leq g(n)$$

we know these two lines are correct. If we $g(m) \leq g(n)$ is correct, $f(g(m)) \leq f(g(n))$ must be monotonically increasing from these two equations because inside of f is also monotonically increasing.

So, **f(g(n)) is monotonically increasing** function.

For the second equation, we know that if a number is nonnegative, if we multiply an equation with that number, equation won't change its direction so:

$$f(m) * g(m) \leq f(n) * g(n)$$

is also correct and in that case **f(n) * g(n) is monotonically** increasing.