Question 1:

(a)In class, we done Selection(A,n,k) using groups of 5. And now, we want to know what would be the run time if we use group of size g instead.

Assumption:1) If $n \leq g, k = 1$ or $k = n$ we solve it by brute force, and it takes $O(g^2)$ time (we could imagine we are using a bubble sort on this case, so it's at most quadratic time)
2) Assume Selection(A,n,k) using g groups takesT(n) runtime

In class we solve Selection(A,n,k) recursively mainly in 3 steps(other steps runs in O(1) and won't affect runtime):
1) divide A into groups of 5, finding median of each group and form a new list B, and find index of median of B using selection algorithm. Write the index we find as i.
2) Using Partition(A, n, i) to find the position of this number after reordering.
3) Compare this postion and k, and discard either numbers before or after this position.

Using group of size g will only affect step(1), we now need to do selection problem of size n/g, so it has runtime T(n/g),
step 2) still runs in O(n) time and step 3) still has run time $T(\frac{3}{4}n)$
Therefor, we have $T(n) = O(n) + T(\frac{1}{g}n) + T(\frac{3}{4}n)$

Guess: Selection algorithm now has runtime $O(g^2n)$
Base case: We have assumed that If $n \leq g, k = 1$ or $k = n$ we solve it by brute force with run time $O(g^2)$
Induction hypothesis: Assume Selection(A, n, k) has rumtime $O(g^2n)$ for all $n < N$
Induction Steps: Now we want to prove Selection(A, N, k) runs in $O(g^2N)$ time

$$
\begin{aligned}
T(n) &= O(n) + T(\frac{n}{g}) + T(\frac{3n}{4}) \\
&= C_1 n + C_2 g^2 \frac{n}{g} + C_2 g^2 \frac{3n}{4} \\
&= (\frac{1}{g^2} C_1 + \frac{1}{g} C_2 + \frac{3}{4} C_2) \times g^2 n \\
&\leq (C_1 + \frac{11}{12} C_2) \times g^2 n \quad \text{since } g \geq 6 \\
&\leq C_2 g^2 \quad \text{when } 12C_1 < C_2
\end{aligned}
$$

Question 2:

(a) We want to prove L(n): RadixSort will properly sort natural numbers with length $l > 0$

Base case: $l = 1$, It's just counting sort .Since we know that counting sort will provide the correct result, we know this should be correct.

Induction hypothesis:Any n natural numbers with length $l < k$ can be sorted correctly by radix sort.

Induction step: Suppose now we want to apply radix sort on n natural numbers of length $l = k$. This can be viewed as first apply radix sort on last k-1 digits and then apply counting sort on the first digit.
By induction hypothesis, we know that it will sort last k-1 digits correctly. And since counting sort is stable(it preserves the original order), we know that when we apply counting sort on the first digit, it will preserve its original order. That is if two numbers have the same highest order term, smaller one will come first because its last k-1 digits will be smaller.

(b) Counting sort n different numbers in range d runs in $O(n+d)$ time. And we have to apply counting sort l times. So the run time of radix sort is $O(l(n+d))$

Question 3:

Credit: Naoki Okada(TA) helps me on question 3(d)

(a) if we use standard counting sort, the time complexity would be $O(n + n^k) = O(n^k)$

(b) $l = \lceil klog_d n \rceil$. There are d^l integers with length l, and the radix (base) is d and we want to it be bigger than the range of integer we want to sort(n^k) l decreases as d increases. When we use d^2 instead of d, l will be halved.

(c) Asymptotic runtime is $O(l(n + d)) = O(\lceil klog_2 n \rceil (n + 2)) = O(nlogn)$

(d) I think d = n would be the best. Then $l = klog_n n = k$ and runtime would be linear($O(l(n + d)) = O(k(n + d)) = O(n)$).