Homework 2

1)

a) The first modification is needed for making strings Same length with each other. In normal Radix sort, numbers are padded with zeros from left (e.g. 25→025). In our case, we can put any character which has lower ASCII value than "A" (a). I chose to put "?" as right padding. (e.g. EFE→EFE???). Second modification is the changing number of bins from (10) to (26+1), because we have 26 alphabetic characters and 1 "?" character. Now, we can perform sorting from right to left.

b) init : [HUSNU, FURKAN, ALI, EMRE, EFE]

Assuming all padding with '?' operation is done before main loop.

Operation : Look for most right char.

Bins : ?, N

?	N
HUSNU?	FURKAN
ALI???	
EMRE??	
EFE???	

iter 0: [HUSNU?, ALI???, EMRE??, EFE???, FURKAN]

operation : Look for next most right char.

Bins :

?	A	U
ALI???	FURKAN	HUSNU?
EMRE??		
EFE???		

iter 1 : [ALI???, EMRE??, EFE???, FURKAN, HUSNU?]

operation : 3rd most right char

Bins : ? E K N

ALI??? EMRE?? FURKAN HUSNU?
EFE???

iter 2 : [ALI???, EFE???, EMRE??, FURKAN, HUSNU?]

operation : 4th most right char.

Bins : E I R S

EFE??? ALI??? EMRE?? HUSNU?
FURKAN

iter 3 : [EFE???, ALI???, EMRE??, FURKAN, HUSNU?]

operation : 5th most right char

Bins : F L M U

EFE??? ALI??? EMRE?? FURKAN
HUSNU?

iter 4 : [EFE???, ALI???, EMRE??, FURKAN, HUSNU?]

operation : 6th most right char

bins : A E F H

ALI??? EFE??? FURKAN HUSNU?
EMRE??

iter 5 : [ALI???, EFE???, EMRE??, FURKAN, HUSNU?]

Sorted list : [ALI, EFE, EMRE, FURKAN, HUSNU]

1) c) Before main loop of Radix Sort, we need to find longest string (word) for padding operation. This will take $O(n)$ as we traverse all list. Then, for adding '?' and removing at the end, we will traverse the list again $2 \times O(n)$.

Then, our main loop will run as d times where d is equal to length of longest word. We know that time complexity for the main loop Radix Sort is $O(d(k+n))$ where k is the number of bins.

In total;

$$T(n) = O(n) + O(n) + O(n) + O(d(k+n))$$

$$\boxed{T(n) = O(d(k+n))} \rightsquigarrow \text{since "k" is constant,} \rightsquigarrow \boxed{O(dn)}$$

In our case $k = 27$, $d = 6$;

$$T(n) = O(6(27+n)) = O(6n)$$

2) a) Following lecture slides analysis for $k=5$;

* Dividing n elements into groups of k elements → $\boxed{O(n)}$

* We need to find medians of each group: If we use Merge Sort algorithm for sorting groups, we have $O(k\log k)$ for each group. Since we have $\lceil \frac{n}{k} \rceil$ groups, total complexity for finding medians → $\lceil \frac{n}{k} \rceil \cdot k\log k = \boxed{O(n \log k)}$

* Now, finding median of these $\lceil \frac{n}{k} \rceil$ medians can be done using WCL_Select function recursively → $\boxed{T(\lceil \frac{n}{k} \rceil)}$

* Since portioning is same for each call → $\boxed{O(n)}$

* m_m is the median of k elements. Therefore, medians of $\frac{1}{2}\frac{n}{k}$ (half of the groups) are smaller or equal to m_m. These $\frac{1}{2}\frac{n}{k}$ medians are medians of their groups. Each group has "k" elements, then there are $\lceil \frac{k}{2} \rceil$ elements in each group that are $\leq$ to their medians. Thus, we have at least $\boxed{\frac{n}{2k} \cdot \lceil \frac{k}{2} \rceil}$ elements in the left-array. This means there will be at most $\boxed{n - \frac{n}{2k} \cdot \lceil \frac{k}{2} \rceil}$ in the right-array. If we think in a same way from right-array perspective upper bound for left-array will be $n - \frac{n}{2k}\lceil \frac{k}{2} \rceil$. Therefore,

$$T(n) \leq \underbrace{O(n)}_{\text{dividing into groups}} + \underbrace{O(n\log k)}_{\text{finding medians}} + \underbrace{T(\lceil \tfrac{n}{k} \rceil)}_{\text{medians of medians}} + \underbrace{O(n)}_{\text{partitioning}} + \underbrace{T(n - \tfrac{n}{2k}\lceil \tfrac{k}{2} \rceil)}_{\text{Recursive Call}}$$

2) b)

1. * Dividing into groups $\rightarrow O(n)$
   * Finding medians with sorting $\rightarrow O(n \log k)$ "merge sort"
   * Finding medians of medians $\rightarrow T\left(\left\lceil \frac{n}{k} \right\rceil\right)$
   * Partitioning $\rightarrow O(n)$

* Recursive Call;

$$M_1 \; \cdots \; \underbrace{\qquad}_{\leq 1/4} \; M_m \; \cdots \; \underbrace{\qquad}_{\leq 3/4} \; M_{\lceil \frac{n}{k} \rceil}$$

$\frac{1}{4} \cdot \frac{n}{k}$ groups are smaller or equal to m_m. There are $\left\lceil \frac{k}{2} \right\rceil$ elements in each group that are $\leq$ their medians.

Thus, we have at least $\frac{n}{4k} \cdot \left\lceil \frac{k}{2} \right\rceil$ elements in the left array.

This means that there will be at most $n - \frac{n}{4k}\left\lceil \frac{k}{2} \right\rceil$ in the right array.

Thus, we can say $n - \frac{n}{4k}\left\lceil \frac{k}{2} \right\rceil$ is upper bound for left-array.

Consequently,

$$\boxed{\begin{gathered} T(n) \leq \underbrace{O(n)}_{\substack{\text{dividing} \\ \text{into} \\ \text{subgroups}}} + \underbrace{O(n \log k)}_{\substack{\text{finding} \\ \text{medians} \\ \text{with sorting}}} + \underbrace{T\left(\left\lceil \frac{n}{k} \right\rceil\right)}_{\substack{\text{medians} \\ \text{of} \\ \text{medians}}} + \underbrace{O(n)}_{\text{partitioning}} + \underbrace{T\left(n - \frac{n}{4k}\left\lceil \frac{k}{2} \right\rceil\right)}_{\substack{\text{Recursive} \\ \text{Call}}} \\ \\ T(n) \leq O(n) + T\left(\left\lceil \frac{n}{k} \right\rceil\right) + T\left(n - \frac{n}{4k}\left\lceil \frac{k}{2} \right\rceil\right) + O(n \log k) \end{gathered}}$$

2) b) 2. We need to find c and n_0 such that $f(n) \geq c \cdot g(n)$ for all $n \geq n_0$.

If we put $k=5$ in our recurrence for <u>lower</u> bound.

$$T(n) \underset{\text{lower bound}}{\geq} O(n) + T\left(\left\lceil \frac{n}{k} \right\rceil\right) + T\left(\frac{n}{4k}\left\lceil \frac{k}{2} \right\rceil\right) + \underbrace{O(n \log k)}_{\text{this is } O(n) \text{ as log } k \text{ const}}$$

$k=5$;

$$T(n) \geq O(n) + T\left(\frac{n}{5}\right) + T\left(\frac{3n}{20}\right)$$

Prove by substitution method;

Initial Guess: $T(n) = \Omega(n \log n)$

Show $\longrightarrow$ $T(n) \geq c n \log n$

$$T(n) \geq O(n) + c\frac{n}{5}\log\left(\frac{n}{5}\right) + c\frac{3}{20}n\log\left(\frac{3n}{20}\right)$$

$$\geq O(n) + \frac{cn}{5}\log n - \frac{cn}{5}\underbrace{\log(5)}_{2.32} + \frac{3cn}{20}\log n - \frac{3cn}{20}\underbrace{\log\left(\frac{20}{3}\right)}_{2.73}$$

$$\geq \underbrace{O(n)}_{*} + 0.35cn\log n \underbrace{-0.87cn}_{**}$$

These will be dominated for large c and $n \geq 2$ as $n\log n$ dominates $O(n)$.

Then, $\boxed{T(n) \geq 0.35\, c\, n \log n}$

This verifies our initial guess.

Therefore $T(n) = \Omega(n \log n)$

3. Initial Guess: $\boxed{T(n) = O(n)}$

Show $\boxed{T(n) \leq cn}$ for some c and $n \geq n_0$

$$O(n) + T\left(\frac{n}{k}\right) + T\left(n - \frac{n}{4k}\underbrace{\left\lceil \frac{k}{2} \right\rceil}\right)$$

when k is odd, this equals to $\frac{k+1}{2}$

$$O(n) + T\left(\frac{n}{k}\right) + T\left(n - \frac{n}{4k} \cdot \frac{k+1}{2}\right)$$

$$O(n) + \frac{cn}{k} + cn - \frac{cn}{4k} \cdot \frac{k+1}{2} \leq cn$$

$$O(n) + \left(\underset{(8)}{\frac{1}{k}} + \underset{(8k)}{1} - \frac{k+1}{8k}\right)cn \leq cn$$

$$O(n) + \frac{7+7k}{8k} cn$$

$$cn + \underbrace{\left(\frac{7-k}{8k} cn + O(n)\right)}_{\text{this part should be negative}} \leq cn$$

this part should be negative. when $c \geq 1$, cn dominates $O(n)$

$$\frac{7-k}{8k} < 0$$ should be satisfied. (k can't be 7)

if $k = 7$, $O(n)$ can not be dominated

$$7 - k < 0$$

$\boxed{k = 9}$ is the smallest odd number for linear running time