Problem 3:

Pseudocode:

CountingSort(A, k (range of digits, 1-9), p):

Compute the count of each element in A

For j = 1 to k

Add the value of the element before each element

For j = len(A) to 1

Place each element in sorted order

RadixSort(A):

Calculate largest element in A (max(A))

Start at 1's column: p = 1

While the largest element still has digits to sort:

CountingSort(A, k (range of digits, 1-9), p)

Go to next significant digit: P = p * 10

Analysis:

RadixSort()

The algorithm starts by sorting the elements based on the value of the least significant digit (the ones column). It relies on a modified version of counting sort to do this. The time complexity of RadixSort() is O(n + k) where n is the size of the input and k is the largest element in either the one's, ten's, hundred's... etc. column. Counting sort will run a for loop that goes up to the last digit of the largest element in the input array. As a result, you can say that the overall time complexity for RadixSort() is O(d*(n + k)).