Homework 8

8- 1- a. Implemented as countingsort.cpp

b. implemented as Bucketsort.cpp

c. Similar to the counting sort.

countingsort (arr)

for i := 1 to k do

C[i] := 0

for j := 1 to n do

C[arr[j]] = C[arr[j]] + 1

for k := 2 to k do

C[k] = C[k] + C[k-1]

for j := n to 1 do

B[C[A[j]]] = A[j]

C[A[j]] = C[A[j]] - 1.

d. Implemented as Sorting words Sortingwords.cpp.

e. The worst case scenario would be when all the inputs fall into one bucket. In this case if we use insertion sort to sort that bucket, the time complexity would be $\theta(n^2)$ and the overall time complexity would be $T(n) = \theta(n^2) + \theta(n)$ which would be $\theta(n^2)$ since it dominates.

Problem 7-2

=> 1. Implemented as radixinvariant.cpp

2. Time Complexity

=>

In the algorithm, each step you would need to divide the number into base case and for that the max depth could be d

where $d = \log_b k$ where with b = base

k = max element.

the complexity would be ~~$T(n) = O$~~ $d(c)$

$$T(n) = O(dn)$$

Space complexity

=> Since it is called recursively for d times the complexity would be $\theta(dn)$