Homework 7.

a) Impdemted in "Counting sort .cpp"

b) Implemted in "Buckdtsort. cpp"

c) This is similar to the counting sort algorithm that we did. The pseudocode is.

```
Temp[k]
For i=0 to k
    Temp[k] ← 0            //intializing to 0
Endfor
For i=0 to n
    Temp[arr[i]] ~~+=~~ ← Temp[arr[i]]+ 1     //calculating instances
                                               of each number
End For
FOR i=~~0~~1 to k
    Temp[i] ← Temp[i] + Temp[i-1]      // Adding them up
End For.
~~Ans =~~ ans = Temp[b] - Temp[a]      // Range is.
```

d) Implemeted in "Wordsort. cpp"

e) ~~This~~ The worst case for Bucetsort would be when all the ~~elem~~ inputs fall into the same bucket and we need to sort only that bucket. If we are using insertion sort the worst case would $\Theta(n^2)$ for insertion sort.

Hence, the overall time complexity would ~~nowe~~ now only be $T(n) = \Theta(n) + \Theta(n^2)$.
hence $T(n) = \Theta(n^2)$ //.

Problem 7.2

a) Implemented in "Radixsortvariant.cpp"

b) Time complexity.

Here in the algorithm each step you would need to divide the number into the base case, and for that the max depth could be is d where $d = \log_b k$ where b is base and k is max element. Therefore simply the complexity would be $T(n) = O(dn)$.

Space complexity

For every ~~buck~~ The bucket sort would have complexity $\Theta(n)$ but bucket sort is called recursively d times as well so the complexity would $\Theta(dn)$.