1.a) In q1a.cpp.

b) In q1b.cpp.

c).

count [k] = {0} . //all values have 0.

for i = 0 to n.

count [arr [i]]++.

Endfor.

for i = 1 to k.

count [i] = count[i] + count [i-1].

Endfor.

Range = count [b] - count [a].

d) In q1d. cpp.

e) Worst case is when all inputs are in the same bucket, if insertion sort is used to sort them, then ~~$O(n) = n$~~ it would be $O(n^2)$.

$\therefore$ Time complexity $= O(n) + O(n^2)$

$\therefore T(n) = O(n^2)$.

2.

a) In q2.cpp.

b)

Radix sort take $O(d \cdot (n+b))$

where, d is digits in input integer.

b is base for the number representation.

in our case $b = 10$.

Let, k be the max value:

$d = \log_b(k)$

$= \log(k)$.

$\therefore O((n+10)\log(k))$.

if $k \leq n^c$, c being an arbitrary constant.

$\Rightarrow O((n+10)\, c\log n)$

$\Rightarrow O(\log n\,(n+10))$

$\Rightarrow O(n\log(n) + 10\log n)$

$\Rightarrow O(n\log n)$.

Space complexity.

For every bucket ~~sort~~ there would a complexity $\Theta(n)$.

But bucket sort is called recursively called d time $(d = \log_b k)$.

Let, $k \le n^c$

$\therefore$ space complexity is $O(n \log n)$.