Name- Antas Jain
Roll no- BT22CSH015

QUESTION 1

Best Case: O(nk) Worst Case: O(nk)
Average Case: O(nk)

```
PS D:\Secondyear> cd "d:\Secondyear\" ; if ($?) { g++ countradixs.cpp -o countradixs } ; if ($?) { .\countradixs }
Sorted array:
 11 12 22 25 34 64 90
PS D:\Secondyear>
```

```
#include<bits/stdc++.h>
using namespace std;

void count(int a[], int pos, int n) {
    int i;
    int co[10] = {0};
    int cu[10] = {0};

    for (i = 0; i < n; i++) {
        co[(a[i] / pos) % 10]++;
    }

    cu[0] = co[0];
    for (i = 1; i < 10; i++) {
```

```
        cu[i] = co[i] + cu[i - 1];
    }

    int b[n];
    for (i = n - 1; i >= 0; i--) {
        cu[(a[i] / pos) % 10]--;
        b[cu[(a[i] / pos) % 10]] = a[i];
    }

    for (i = 0; i < n; i++) {
        a[i] = b[i];
    }
}

void radix(int a[], int n, int mdig) {
    for (int pos = 1; mdig / pos > 0; pos *= 10) {
        count(a, pos, n);
    }
}

int main() {
    int n, i;
    cin >> n;
    int mdig;
    cin >> mdig;
```

```
    int a[n];
    cout<<”sorted array:\n”;
    for (i = 0; i < n; i++) {
        cin >> a[i];
    }

    radix(a, n, mdig);

    for (i = 0; i < n; i++) {
        cout << a[i] << " ";
    }

    return 0;
}
```

Radix Sort's time complexity is O(nk), where 'n' is the number of elements, and 'k' is the number of digits in the maximum number. It sorts by processing each digit using a stable sorting algorithm within a loop that runs 'k' times. This results in linear time 'n' for each digit iteration, leading to a total time complexity of O(nk) for all cases.

QUESTION2

best Case Time: O(d * n) - Uniform distribution in buckets.
Average Case Time: O(d * n) - Reasonable bucket distribution.
Worst Case Time: O(d * n^2) - All in one bucket per pass.
Given scenario: O(d * n) - Specific case time complexity.

```
#include <bits/stdc++.h>
using namespace std;

struct N
{
    int d;
    struct N *n;
};

N *cN(int d)
{
    N *nn = new N;
    nn->d = d;
    nn->n = NULL;
    return nn;
}

void insN(N *&h, int d)
{
    N *nn = cN(d);
    if (!h)
    {
        h = nn;
```

```
    }
    else
    {
        N *c = h;
        while (c->n)
        {
            c = c->n;
        }
        c->n = nn;
    }
}

void prN(N *h)
{
    N *c = h;
    while (c)
    {
        cout << c->d << " ";
        c = c->n;
    }
    cout << endl;
}

int numDigits(int num)
{
    int count = 0;
    while (num > 0)
    {
        num /= 10;
        count++;
    }
    return count;
```

```
}

void rs(N *&h)
{
    int mD = 0;
    N *c = h;
    while (c)
    {
        int dgt = numDigits(c->d);
        mD = max(mD, dgt);
        c = c->n;
    }

    for (int i = 1; i <= mD; ++i)
    {
        N *b[10] = {NULL};
        c = h;
        while (c)
        {
            int dgt = (c->d / int(pow(10, i - 1))) % 10;
            insN(b[dgt], c->d);
            c = c->n;
        }

        N *sl = NULL;
        for (int j = 0; j < 10; ++j)
        {
            N *bk = b[j];
            while (bk)
            {
                insN(sl, bk->d);
                bk = bk->n;
```

```
            }
        }
        h = sl;
    }
}

int main()
{
    N *h = NULL;
    int a[] = {1234, 4321, 5678, 9876, 3456, 6789, 8765, 2109,
7890, 9012};
    int n = sizeof(a) / sizeof(a[0]);
    for (int i = 0; i < n; i++)
    {
        insN(h, a[i]);
    }

    rs(h);
    cout << "Sorted array:\n ";
    prN(h);
    return 0;
}
```