Assignment - 6

Name: Dhairya Arora

Enrolment Number: 01616401522

B.Tech(IT) 4th Semester

2022-26

Code:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <chrono>
using namespace std;
using namespace std::chrono;
// Function to perform Bucket Sort
void bucketSort(vector<int> &arr)
{
   auto start_time = high_resolution_clock::now();

   // Find the maximum element in the array
   int maxVal = *max_element(arr.begin(), arr.end());

   // Create buckets
   vector<vector<int>> buckets(maxVal + 1);
   // Distribute elements into buckets
   for (int num : arr)
   {
```

```
        buckets[num].push_back(num);
    }
    // Sort individual buckets and concatenate
    arr.clear();
    for (auto &bucket : buckets)
    {
        sort(bucket.begin(), bucket.end());
        arr.insert(arr.end(), bucket.begin(), bucket.end());
    }
    auto end_time = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(end_time - start_time);
    cout << "Time taken by Bucket Sort - " << duration.count() << "Microseconds" << endl;
}
// Function to perform Radix Sort
void radixSort(vector<int> &arr)
{
    auto start_time = high_resolution_clock::now();
    // Find the maximum number to determine the number of digits
    int maxVal = *max_element(arr.begin(), arr.end());
    int exp = 1;
    while (maxVal / exp > 0)
    {
        vector<vector<int>> buckets(10);
        // Distribute elements into buckets based on current digit
        for (int num : arr)
        {
            buckets[(num / exp) % 10].push_back(num);
        }
        // Concatenate buckets
```

```
        arr.clear();
        for (auto &bucket : buckets)
        {
            arr.insert(arr.end(), bucket.begin(), bucket.end());
        }
        exp *= 10;
    }
    auto end_time = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(end_time - start_time);
    cout << "Time taken by Radix Sort - " << duration.count() << "Microseconds" << endl;
}
// Function to perform Counting Sort
void countingSort(vector<int> &arr)
{
    auto start_time = high_resolution_clock::now();
    // Find the maximum and minimum values in the array
    int maxVal = *max_element(arr.begin(), arr.end());
    int minVal = *min_element(arr.begin(), arr.end());
    // Calculate the range of values
    int range = maxVal - minVal + 1;
    // Create a counting array and initialize with zeros
    vector<int> count(range, 0);
    // Count occurrences of each element
    for (int num : arr)
    {
        count[num - minVal]++;
    }
    // Reconstruct the sorted array
    arr.clear();
```

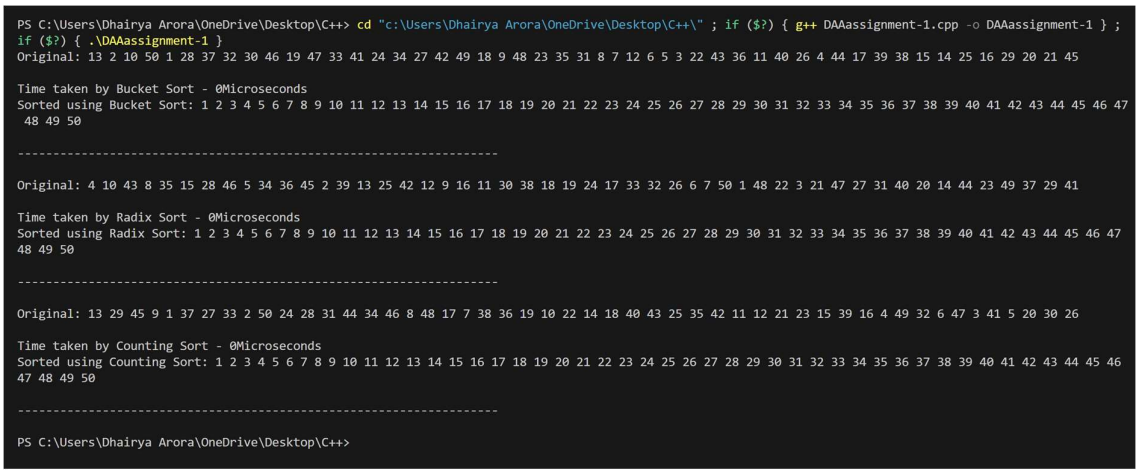
```
    for (int i = 0; i < range; i++)
    {
        while (count[i] > 0)
        {
            arr.push_back(i + minVal);
            count[i]--;
        }
    }
    auto end_time = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(end_time - start_time);
    cout << "Time taken by Counting Sort - " << duration.count() << "Microseconds" << endl;
}
vector<int> generateRandomArray(int size)
{
    vector<int> arr;
    for (int i = 1; i <= size; ++i)
    {
        arr.push_back(i);
    }
    random_shuffle(arr.begin(), arr.end());
    return arr;
}
int main()
{
    void (*sortAlgorithms[6])(vector<int> &) = {bucketSort, radixSort,
                                        countingSort};
    // Applying loop to call each sorting function
    for (int i = 0; i < 3; i++)
    {
```

```
int arraySize = 50;
cout << "Original: ";
vector<int> searchValues = generateRandomArray(arraySize);
for (int num : searchValues)
{
   cout << num << " ";
}
cout << endl
    << endl;
sortAlgorithms[i](searchValues);
// Display sorted array
cout << "Sorted using ";

if (i == 0)
{
   cout << "Bucket Sort: ";
}
else if (i == 1)
{
   cout << "Radix Sort: ";
}
else
{
   cout << "Counting Sort: ";
}
for (int num : searchValues)
{
   cout << num << " ";
}
```

```
        cout << endl
             << "\n-----------------------------------------------------------------\n"
             << endl;
    }
    return 0;
}
```

Output:



```
PS C:\Users\Dhairya Arora\OneDrive\Desktop\C++> cd "c:\Users\Dhairya Arora\OneDrive\Desktop\C++\" ; if ($?) { g++ DAAassignment-1.cpp -o DAAassignment-1 } ;
if ($?) { .\DAAassignment-1 }
Original: 13 2 10 50 1 28 37 32 30 46 19 47 33 41 24 34 27 42 49 18 9 48 23 35 31 8 7 12 6 5 3 22 43 36 11 40 26 4 44 17 39 38 15 14 25 16 29 20 21 45

Time taken by Bucket Sort - 0Microseconds
Sorted using Bucket Sort: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50

-----------------------------------------------------------------

Original: 4 10 43 8 35 15 28 46 5 34 36 45 2 39 13 25 42 12 9 16 11 30 38 18 19 24 17 33 32 26 6 7 50 1 48 22 3 21 47 27 31 40 20 14 44 23 49 37 29 41

Time taken by Radix Sort - 0Microseconds
Sorted using Radix Sort: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50

-----------------------------------------------------------------

Original: 13 29 45 9 1 37 27 33 2 50 24 28 31 44 34 46 8 48 17 7 38 36 19 10 22 14 18 40 43 25 35 42 11 12 21 23 15 39 16 4 49 32 6 47 3 41 5 20 30 26

Time taken by Counting Sort - 0Microseconds
Sorted using Counting Sort: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48 49 50

-----------------------------------------------------------------

PS C:\Users\Dhairya Arora\OneDrive\Desktop\C++>
```