

# Assignment on Radix Sort (C Program)

Radix Sort is a sorting technique that sorts numbers digit by digit starting from the least significant digit (unit place) to the most significant digit. It uses Counting Sort as a subroutine to sort numbers based on each digit. This method works efficiently for integers and avoids direct comparisons between numbers.

## Algorithm for Radix Sort:

1. Find the maximum number in the list to determine the number of digits.
2. Set  $\text{exp} = 1$  (for units place).
3. While  $(\text{max} / \text{exp}) > 0$  do:
  - a. Perform Counting Sort on the array according to the digit at  $\text{exp}$  position.
  - b. Multiply  $\text{exp}$  by 10 to move to the next digit place.
4. Stop when all digits are processed.

## C Program for Radix Sort:

```
#include <stdio.h>

int Max_val(int arr[], int n) {
    int max = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];
    return max;
}

void countingSort(int arr[], int n, int exp) {
    int a[n];
    int count[10] = {0};

    for (int i = 0; i < n; i++)
        count[(arr[i] / exp) % 10]++;

    for (int i = 1; i < 10; i++)
        count[i] += count[i - 1];

    for (int i = n - 1; i >= 0; i--) {
        int digit = (arr[i] / exp) % 10;
        a[count[digit] - 1] = arr[i];
        count[digit]--;
    }

    for (int i = 0; i < n; i++)
        arr[i] = a[i];
}

void radixSort(int arr[], int n) {
    int max = Max_val(arr, n);
    for (int exp = 1; max / exp > 0; exp *= 10)
        countingSort(arr, n, exp);
}

int main() {
    int arr[5];
    printf("Enter array elements:\n");
    for (int i = 0; i < 5; i++) {
        scanf("%d", &arr[i]);
    }

    int n = sizeof(arr) / sizeof(arr[0]);
    radixSort(arr, n);

    printf("Sorted array:\n");
    for (int i = 0; i < n; i++)
        printf("%d\n", arr[i]);
    return 0;
}
```

}

**Diagram of Working Process:**

Below diagram shows how Radix Sort works step-by-step.

Numbers are sorted digit by digit using Counting Sort starting from units place (1s), then tens (10s), then hundreds (100s), and so on.

Example: Sorting [170, 45, 75, 90, 802, 24, 2, 66]

Step 1: Sort by Units place

[170, 90, 802, 2, 24, 45, 75, 66]

Step 2: Sort by Tens place

[802, 2, 24, 45, 66, 75, 170, 90]

Step 3: Sort by Hundreds place

[2, 24, 45, 66, 75, 90, 170, 802]

Final Sorted Array: [2, 24, 45, 66, 75, 90, 170, 802]