Practical – 10 Sorting Techniques

1. Write a Program to collect an unsorted array from the user. Implement sorting of the array using following techniques.

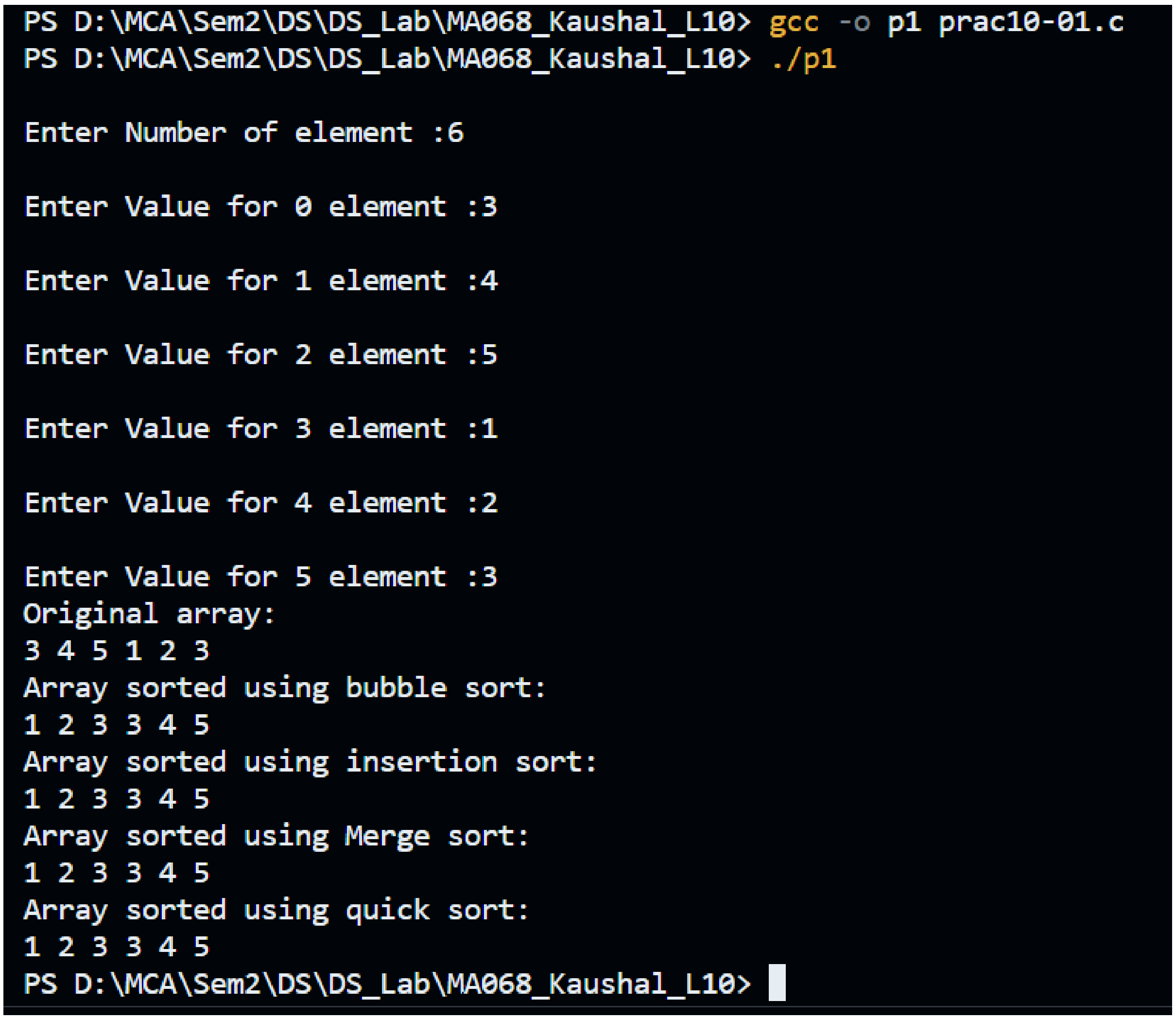
- **bubble sort**
- **quick sort.**
- **insertion sort**
- **Merge sort**

Code :

```
#include <stdio.h>

void print_array(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void bubble_sort(int arr[], int n) {
    int temp;
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

void insertion_sort(int arr[], int n) {
    int i, j, key;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = key;
    }
}


void merge(int arr[], int left, int mid, int right)
{
int i, j, k;
int s1 = mid - left + 1;
int s2 = right - mid;

int left_arr[s1], right_arr[s2];
```

```
    for (i = 0; i < s1; i++)
        left_arr[i] = arr[left + i];
    for (j = 0; j < s2; j++)
        right_arr[j] = arr[mid + 1 + j];

    i = 0, j = 0;
    k = left;

    while (i < s1 && j < s2) {
        if (left_arr[i] <= right_arr[j]) {
            arr[k] = left_arr[i];
            i++;
        }
        else {
            arr[k] = right_arr[j];
            j++;
        }
    k++;
    }

    while (i < s1) {
        arr[k] = left_arr[i];
        i++;
        k++;
    }

    while (j < s2) {
        arr[k] = right_arr[j];
        j++;
        k++;
    }
}
void merge_sort(int arr[], int left, int right)
{
    if (left < right) {

        int mid = left + (right - left) / 2;

        merge_sort(arr, left, mid);

        merge_sort(arr, mid + 1, right);

        merge(arr, left, mid, right);
    }
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    int temp;
    for (int j = low; j <= high-1; j++) {
        if (arr[j] <= pivot) {
            i++;
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    temp = arr[i+1];
    arr[i+1] = arr[high];
    arr[high] = temp;
    return i+1;
}

void quick_sort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quick_sort(arr, low, pi-1);
        quick_sort(arr, pi+1, high);
    }
}

int main() {
      int n;

    //Create Array
    printf("\nEnter Number of element :");
    scanf("%d",&n);

    int arr[n];
    for(int i=0; i<n; i++){
        printf("\nEnter Value for %d element :",i);
        scanf("%d",&arr[i]);
    }

    printf("Original array:\n");
    print_array(arr, n);

    bubble_sort(arr, n);
    printf("Array sorted using bubble sort:\n");
    print_array(arr, n);

    insertion_sort(arr, n);
    printf("Array sorted using insertion sort:\n");
    print_array(arr, n);

    merge_sort(arr,0, n-1);
    printf("Array sorted using Merge sort:\n");
    print_array(arr, n);

    quick_sort(arr, 0, n-1);
    printf("Array sorted using quick sort:\n");
    print_array(arr, n);

    return 0;
}
```

Output :



```
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L10> gcc -o p1 prac10-01.c
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L10> ./p1

Enter Number of element :6

Enter Value for 0 element :3

Enter Value for 1 element :4

Enter Value for 2 element :5

Enter Value for 3 element :1

Enter Value for 4 element :2

Enter Value for 5 element :3
Original array:
3 4 5 1 2 3
Array sorted using bubble sort:
1 2 3 3 4 5
Array sorted using insertion sort:
1 2 3 3 4 5
Array sorted using Merge sort:
1 2 3 3 4 5
Array sorted using quick sort:
1 2 3 3 4 5
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L10>
```