//1.INSERTION SORT:

```
#include <stdio.h>
void insertionSort(int arr[], int n){
    for (int i = 1; i < n; ++i) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
void printArray(int arr[], int n){
    for (int i = 0; i < n; ++i)
        printf("%d ", arr[i]);
    printf("\n");
}
int main(){
    int arr[] = { 7,3,10,4,1,11};
    int n = sizeof(arr) / sizeof(arr[0]);
    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

OUTPUT:

```
1 3 4 7 10 11
```

//2.MERGESORT:

```
#include <stdio.h>
#include <stdlib.h>
void merge(int arr[], int l, int m, int r){
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        }
        else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
```

```
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
void mergeSort(int arr[], int l, int r){
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}
void printArray(int A[], int size){
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}
int main(){
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);
```

```
    printf("Given array is \n");
    printArray(arr, arr_size);
    mergeSort(arr, 0, arr_size - 1);
    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}
```

OUTPUT:

```
Given array is
12 11 13 5 6 7
Sorted array is
5 6 7 11 12 13
```

//3.RADIAX SORT:

```
#include <stdio.h>
int getMax(int arr[], int n) {
    int mx = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > mx)
            mx = arr[i];
    return mx;
}
void countSort(int arr[], int n, int exp) {
    int output[n]; // Output array
    int count[10] = {0}; // Initialize count array as 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--) {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }
    for (int i = 0; i < n; i++)
        arr[i] = output[i];
}
void radixSort(int arr[], int n) {
    int m = getMax(arr, n);
    for (int exp = 1; m / exp > 0; exp *= 10)
        countSort(arr, n, exp);
}
void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main() {
    int arr[] = {170, 45, 75, 90, 802, 24, 2, 66};
    int n = sizeof(arr) / sizeof(arr[0]);
    radixSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

OUTPUT:

```
2 24 45 66 75 90 170 802
```