1.Give a c program for insertion sorting.

Program:

```
// Insertion sort program in C

#include <stdio.h>

// Function to perform insertion sort
void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

// Function to print an array
void printArray(int arr[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr[] = {5, 2, 4, 6, 1, 3};
    int n = sizeof(arr) / sizeof(arr[0]);

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

    insertionSort(arr, n);

    printf("Sorted array: \n");
    printArray(arr, n);

    return 0;
}
```

Input:

```
Original array:
12 11 13 15 6
```

Output:

```
Sorted array:
6 11 12 13 15
```

2.Give a c program for merge sorting.

Program:

```
// Merge sort program in C

#include <stdio.h>

// Function to merge two subarrays
void merge(int arr[], int l, int m, int r) {
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    // Create temporary arrays
    int L[n1], R[n2];

    // Copy data to temporary arrays
    for (i = 0; i < n1; i++) {
        L[i] = arr[l + i];
    }
    for (j = 0; j < n2; j++) {
        R[j] = arr[m + 1 + j];
    }

    // Merge the temporary arrays back into arr[l..r]
    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++;
    }

    // Copy the remaining elements of L[], if there are any
    while (i < n1) {
        arr[k] = L[i];
```

```
        i++;
        k++;
    }

    // Copy the remaining elements of R[], if there are any
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

// Function to perform merge sort
void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        // Merge the sorted halves
        merge(arr, l, m, r);
    }
}

// Function to print an array
void printArray(int arr[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int arr[] = {5, 2, 4, 6, 1, 3};
    int n = sizeof(arr) / sizeof(arr[0]);

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

    mergeSort(arr, 0, n - 1);

    printf("Sorted array: \n");
    printArray(arr, n);

    return 0;
}
```

Input:

```
Original array:
48 12 86 3 24 36 9
```

Output:

```
3 9 12 24 36 48 86
```

3.Give a c program for radix sorting.

Program:

```
// Radix sort program in C

#include <stdio.h>

// Function to get the maximum element in the array
```

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

// Function to perform counting sort based on a digit
void countingSort(int arr[], int n, int exp) {
    int output[n];
    int count[10] = {0};

    // Count the occurrences of each digit
    for (int i = 0; i < n; i++) {
        count[(arr[i] / exp) % 10]++;
    }

    // Calculate the cumulative count
    for (int i = 1; i < 10; i++) {
        count[i] += count[i - 1];
    }

    // Build the output array
    for (int i = n - 1; i >= 0; i--) {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }
    // Copy the output array to the original array
    for (int i = 0; i < n; i++) {
        arr[i] = output[i];
    }
}

// Function to perform radix sort
void radixSort(int arr[], int n) {
    int max = getMax(arr, n);

    // Perform counting sort for each digit
    for (int exp = 1; max / exp > 0; exp *= 10) {
        countingSort(arr, n, exp);
    }
}

// Function to print an array
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]);

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


    printf("Sorted array: \n");
    printArray(arr, n);


    return 0;
}
```

Input:

Original array:

126 328 636 90 341

Output:

90 126 328 341 636