1.WRITE A C PROGRAM ON INSERTION SORT.

```
#include <stdio.h>

int main() {

    int array[] = {12, 11, 13, 5, 6};

    int n = sizeof(array) / sizeof(array[0]);

    printf("Original array:\n");

    for (int i = 0; i < n; i++) {

        printf("%d ", array[i]);

    }

    printf("\n");

    for (int i = 1; i < n; i++) {

        int key = array[i];

        int j = i - 1;

        while (j >= 0 && array[j] > key) {

            array[j + 1] = array[j];

            j = j - 1;

        }

        array[j + 1] = key;

    }

    printf("Sorted array:\n");

    for (int i = 0; i < n; i++) {

        printf("%d ", array[i]);

    }

    printf("\n");


    return 0;

}
```

SAMPLE OUTPUT:

```
Original array:

12 11 13 5 6

Sorted array:
```

```
5 6 11 12 13
```

2.WRITE A C PROGRAM ON MERGE SORT.

```
#include <stdio.h>
void merge(int array[], int left, int middle, int right) {
    int n1 = middle - left + 1;
    int n2 = right - middle;
    int leftArray[n1], rightArray[n2];
    for (int i = 0; i < n1; i++)
        leftArray[i] = array[left + i];
    for (int j = 0; j < n2; j++)
        rightArray[j] = array[middle + 1 + j];
    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if (leftArray[i] <= rightArray[j]) {
            array[k] = leftArray[i];
            i++;
        } else {
            array[k] = rightArray[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
        array[k] = leftArray[i];
        i++;
        k++;
    }
    while (j < n2) {
        array[k] = rightArray[j];
        j++;
        k++;
    }
}
int main() {
    int array[] = {12, 11, 13, 5, 6, 7};
    int n = sizeof(array) / sizeof(array[0]);
    printf("Original array:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");
    int currentSize;
    int leftStart;
    for (currentSize = 1; currentSize <= n - 1; currentSize = 2 * currentSize) {
        for (leftStart = 0; leftStart < n - 1; leftStart += 2 * currentSize) {
            int mid = leftStart + currentSize - 1;
            int rightEnd = ((leftStart + 2 * currentSize - 1) < (n - 1)) ? (leftStart + 2 * currentSize - 1) : (n -
1);
            merge(array, leftStart, mid, rightEnd);
        }
    }
    printf("Sorted array:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");
    return 0;
}
```

SAMPLE OUTPUT:

```
Original array:
12 11 13 5 6 7
Sorted array:
```

```
5 6 7 11 12 13
```

3.WRITE A C PROGRAM ON RADIX SORT.

```
#include <stdio.h>

#include <stdlib.h>

int getMax(int array[], int n) {

    int max = array[0];

    for (int i = 1; i < n; i++) {

        if (array[i] > max) {

            max = array[i];

        }

    }

    return max;

}

void countingSort(int array[], int n, int exp) {

    int output[n];

    int i, count[10] = {0};

    for (i = 0; i < n; i++) {

        count[(array[i] / exp) % 10]++;

    }

    for (i = 1; i < 10; i++) {

        count[i] += count[i - 1];

    }

    for (i = n - 1; i >= 0; i--) {

        output[count[(array[i] / exp) % 10] - 1] = array[i];

        count[(array[i] / exp) % 10]--;

    }

    for (i = 0; i < n; i++) {

        array[i] = output[i];

    }

}

int main() {
    int array[] = {170, 45, 75, 90, 802, 24, 2, 66};
    int n = sizeof(array) / sizeof(array[0]);
    printf("Original array:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");
    int m = getMax(array, n);
    for (int exp = 1; m / exp > 0; exp *= 10) {
        countingSort(array, n, exp);
    }
    printf("Sorted array:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");

    return 0;
}
```

SAMPLE OUTPUT:

```
Original array:
170 45 75 90 802 24 2 66
Sorted array:
2 24 45 66 75 90 170 802
```