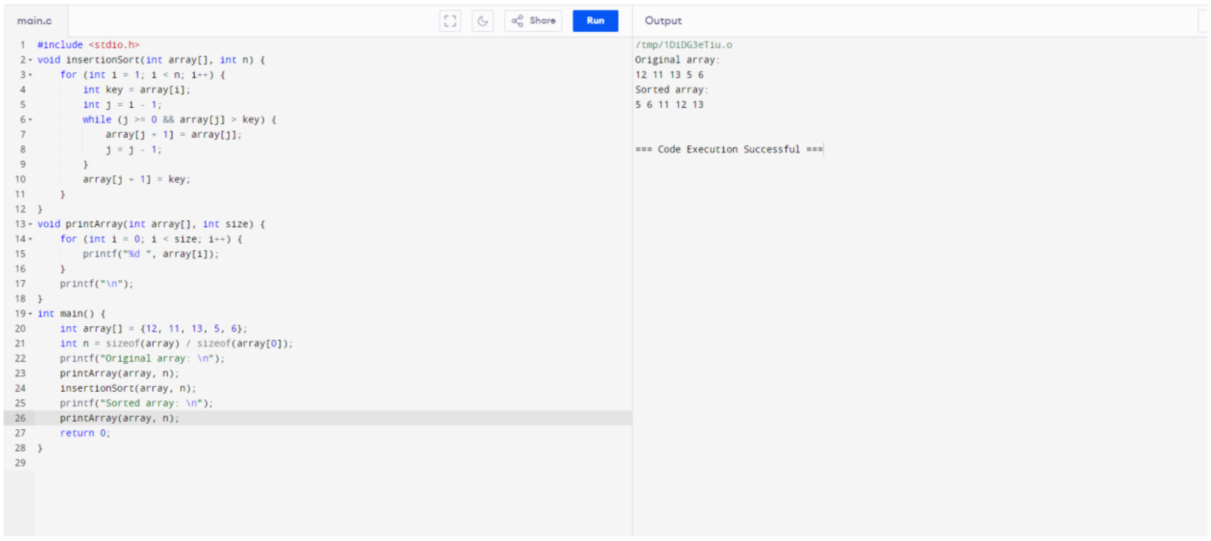
1.Write a c program for insertion sort.

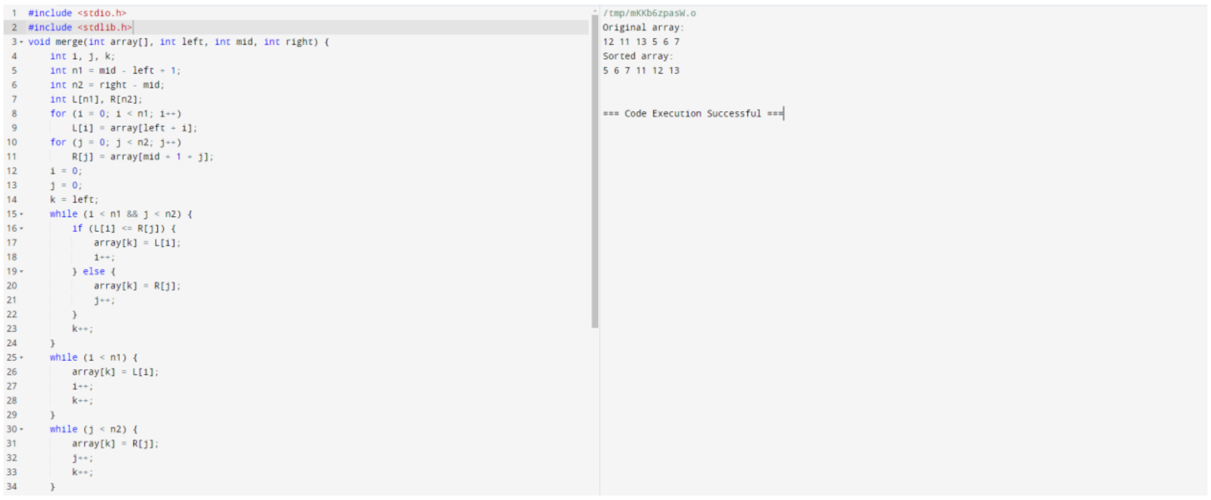


```c
#include <stdio.h>
void insertionSort(int array[], int n) {
    for (int i = 1; i < n; i++) {
        int key = array[i];
        int j = i - 1;
        while (j >= 0 && array[j] > key) {
            array[j + 1] = array[j];
            j = j - 1;
        }
        array[j + 1] = key;
    }
}
void printArray(int array[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");
}
int main() {
    int array[] = {12, 11, 13, 5, 6};
    int n = sizeof(array) / sizeof(array[0]);
    printf("Original array: \n");
    printArray(array, n);
    insertionSort(array, n);
    printf("Sorted array: \n");
    printArray(array, n);
    return 0;
}
```

```
/tmp/1D1DG3eTiu.o
Original array:
12 11 13 5 6
Sorted array:
5 6 11 12 13


=== Code Execution Successful ===
```

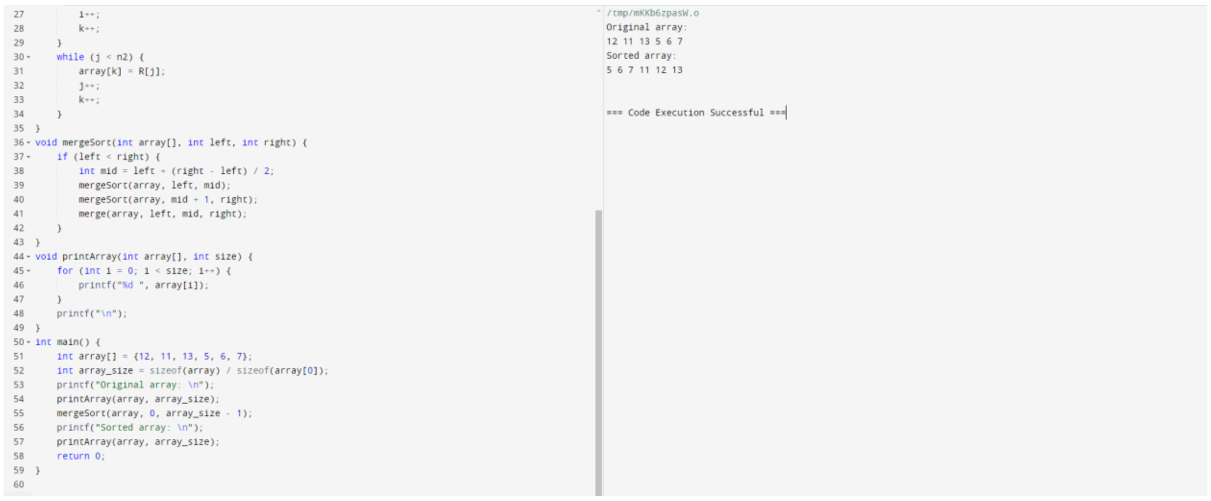
2.Write a c program for merge sort.



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

```
/tmp/mKKb6zpasW.o
Original array:
12 11 13 5 6 7
Sorted array:
5 6 7 11 12 13


=== Code Execution Successful ===
```

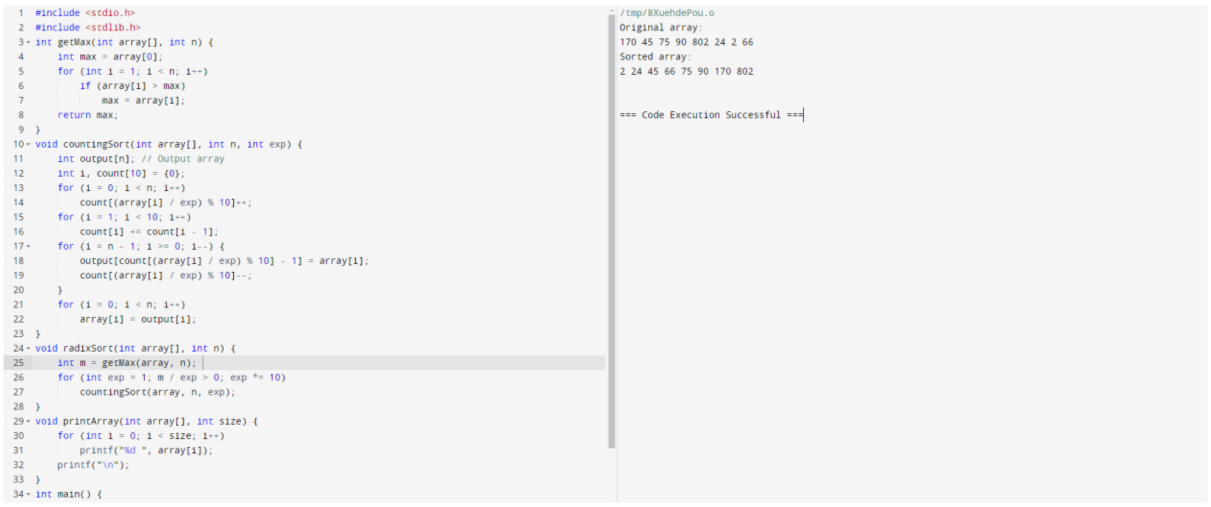


```c
        i++;
        k++;
    }
    while (j < n2) {
        array[k] = R[j];
        j++;
        k++;
    }
}
void mergeSort(int array[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(array, left, mid);
        mergeSort(array, mid + 1, right);
        merge(array, left, mid, right);
    }
}
void printArray(int array[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");
}
int main() {
    int array[] = {12, 11, 13, 5, 6, 7};
    int array_size = sizeof(array) / sizeof(array[0]);
    printf("Original array: \n");
    printArray(array, array_size);
    mergeSort(array, 0, array_size - 1);
    printf("Sorted array: \n");
    printArray(array, array_size);
    return 0;
}
```

```
/tmp/mKKb6zpasW.o
Original array:
12 11 13 5 6 7
Sorted array:
5 6 7 11 12 13


=== Code Execution Successful ===
```

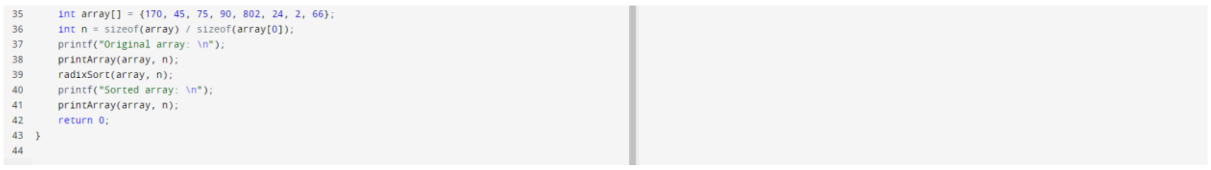
3.Write a c program for Radix sort.



```c
#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]; // Output array
    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];
}
void radixSort(int array[], int n) {
    int m = getMax(array, n);
    for (int exp = 1; m / exp > 0; exp *= 10)
        countingSort(array, n, exp);
}
void printArray(int array[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", array[i]);
    printf("\n");
}
int main() {
```

```
/tmp/8XuehdePou.o
Original array:
170 45 75 90 802 24 2 66
Sorted array:
2 24 45 66 75 90 170 802


=== Code Execution Successful ===
```



```c
    int array[] = {170, 45, 75, 90, 802, 24, 2, 66};
    int n = sizeof(array) / sizeof(array[0]);
    printf("Original array: \n");
    printArray(array, n);
    radixSort(array, n);
    printf("Sorted array: \n");
    printArray(array, n);
    return 0;
}
```