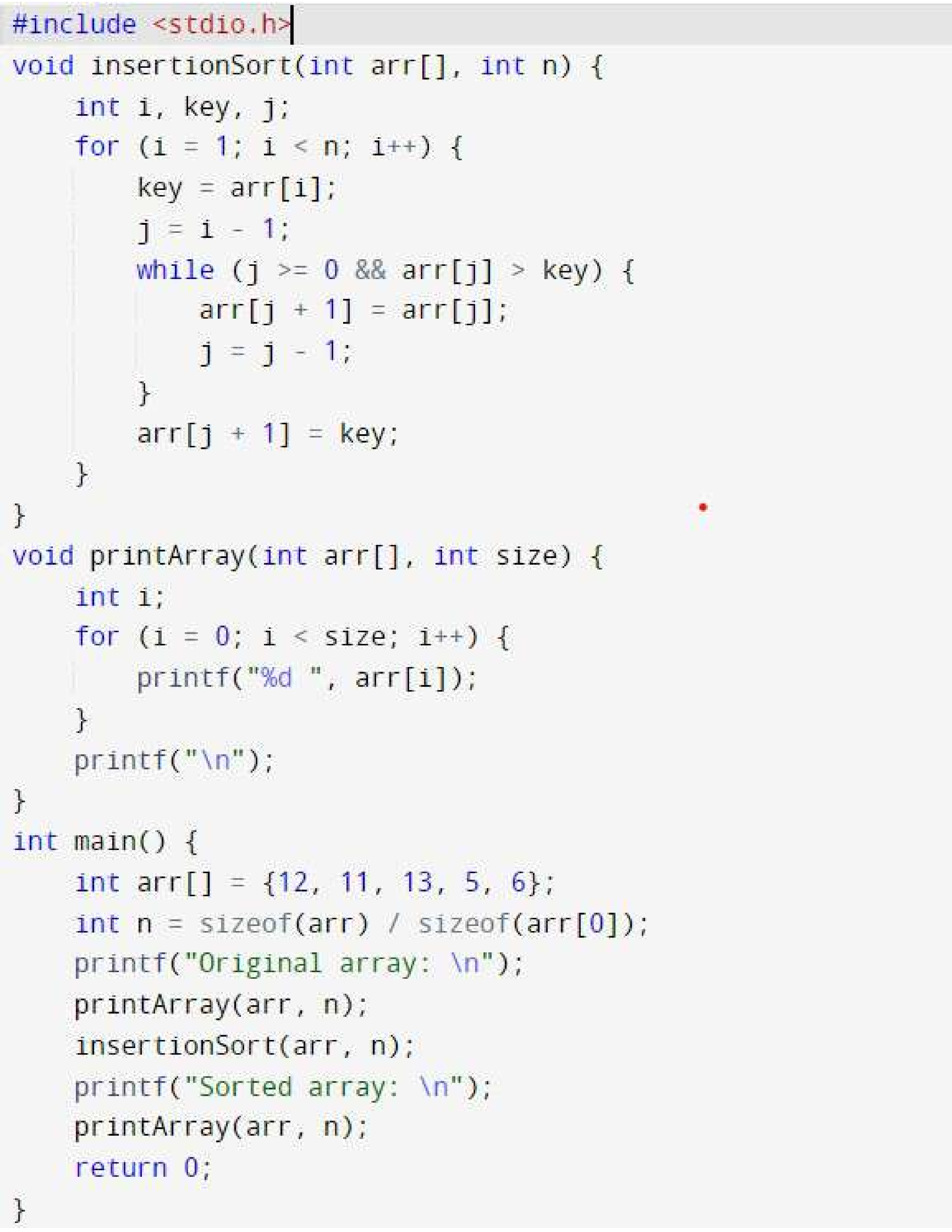
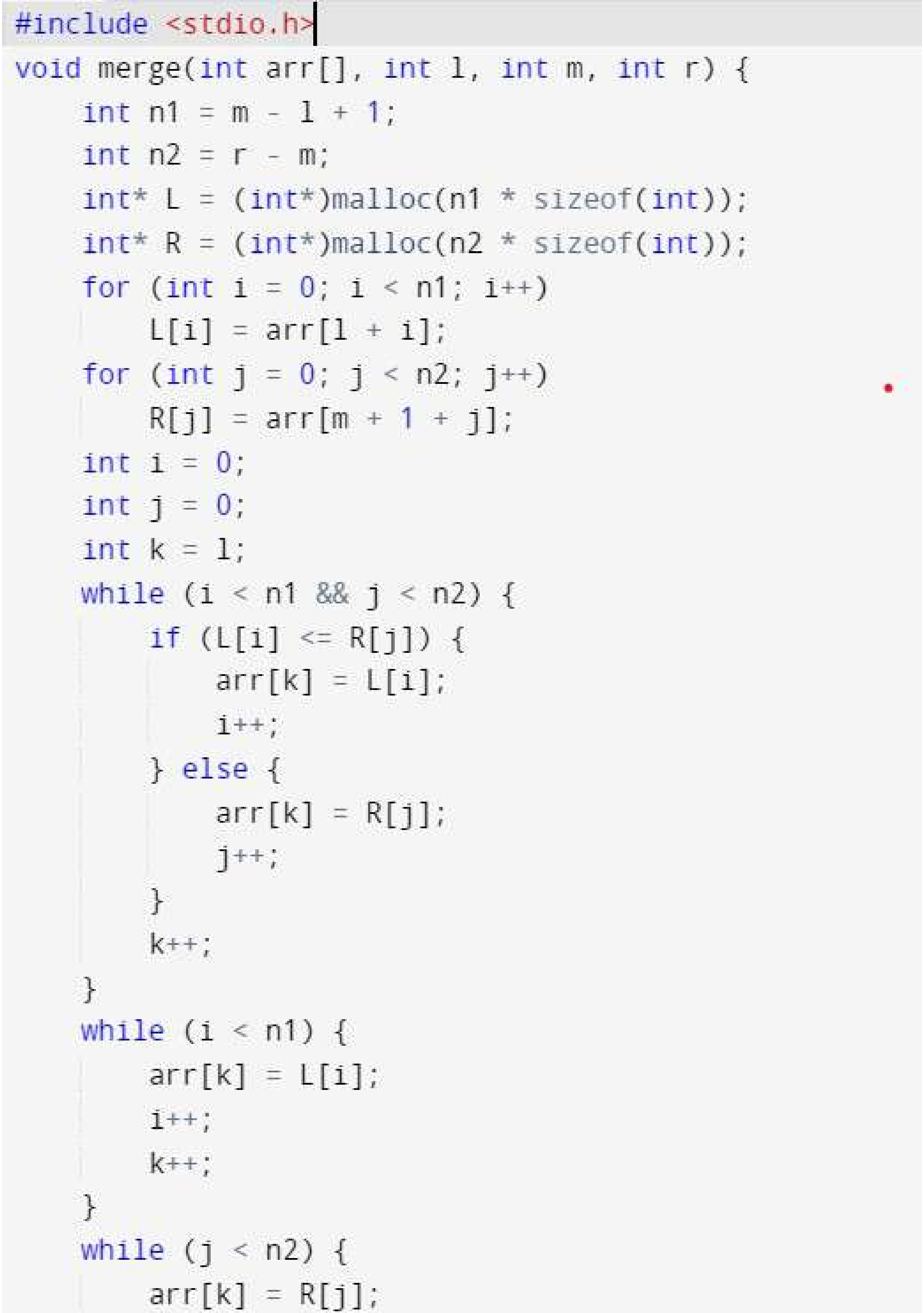
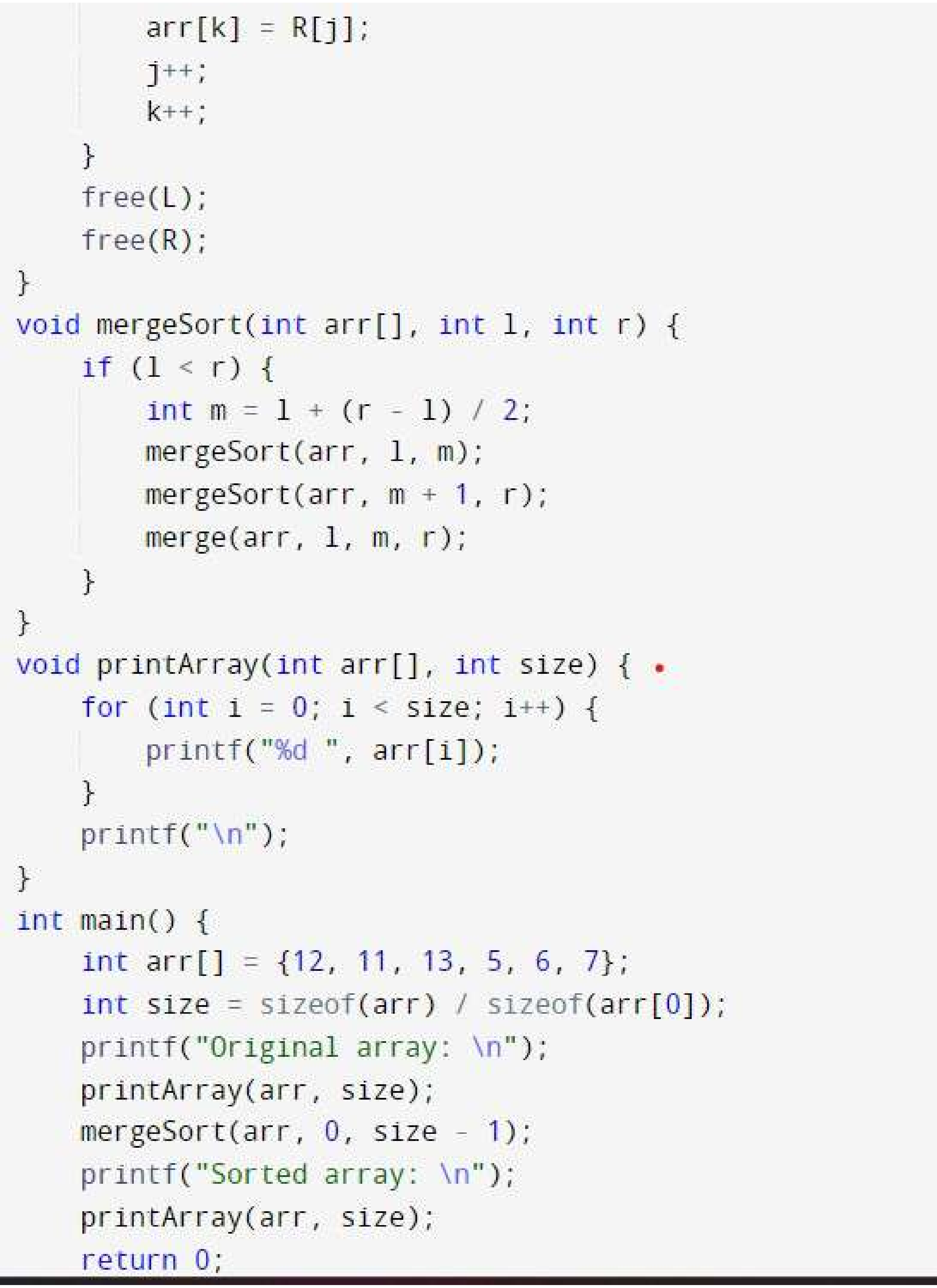


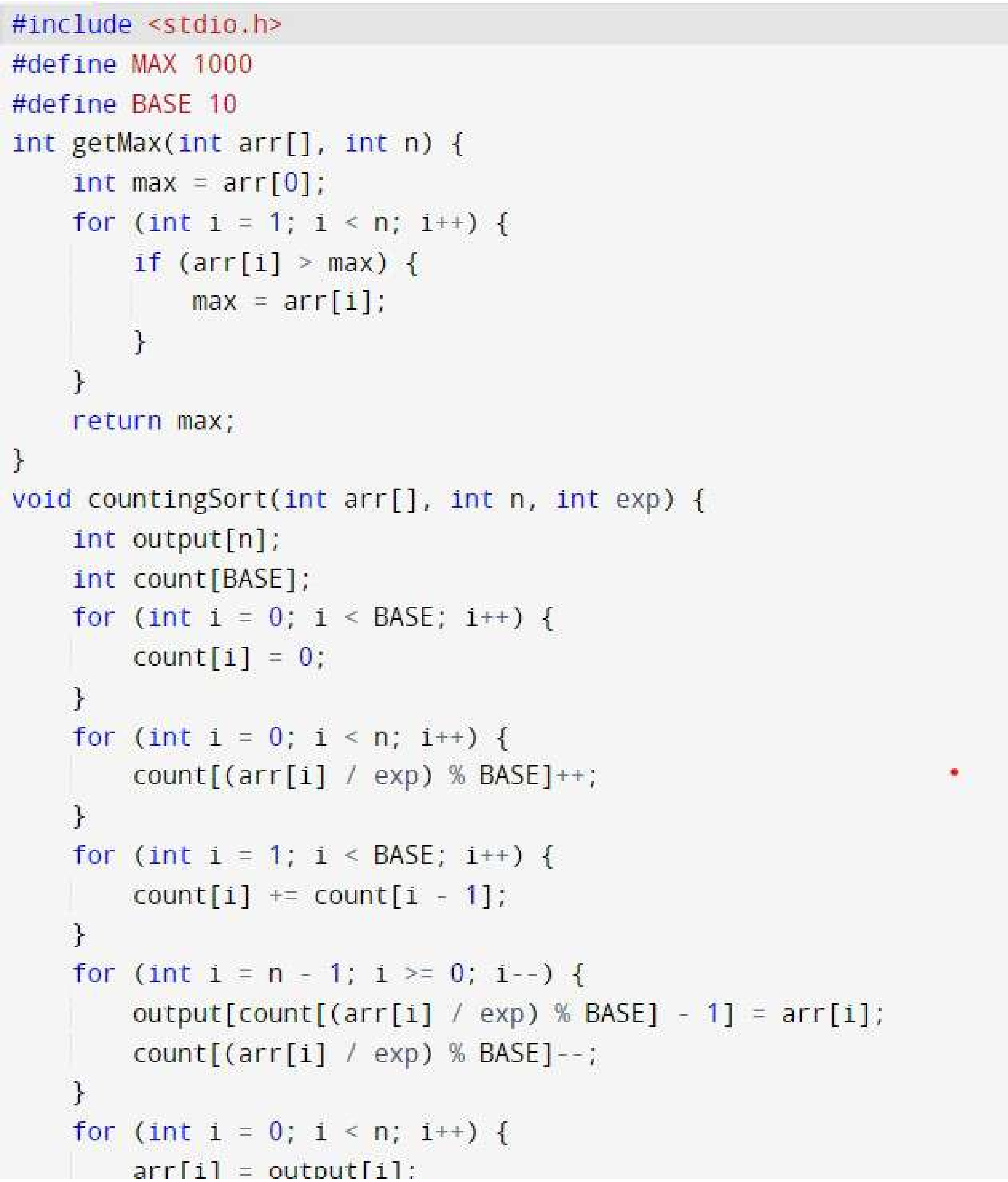
```c
#include <stdio.h>
void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
void printArray(int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



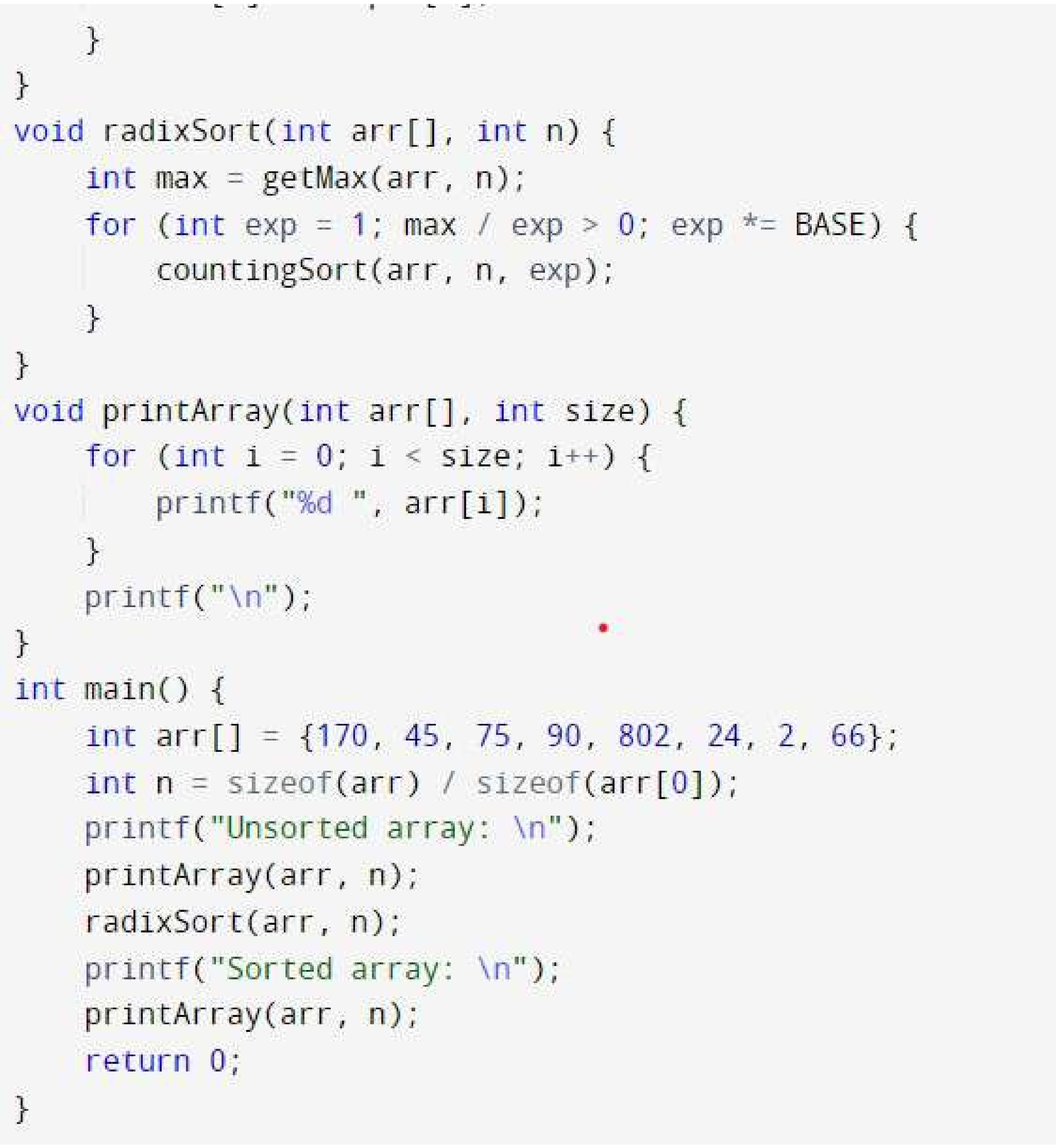
```c
#include <stdio.h>
void merge(int arr[], int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;
    int* L = (int*)malloc(n1 * sizeof(int));
    int* R = (int*)malloc(n2 * sizeof(int));
    for (int i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    int i = 0;
    int j = 0;
    int 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];
```



```c
        arr[k] = R[j];
        j++;
        k++;
    }
    free(L);
    free(R);
}
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 arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Original array: \n");
    printArray(arr, size);
    mergeSort(arr, 0, size - 1);
    printf("Sorted array: \n");
    printArray(arr, size);
    return 0;
```



```c
#include <stdio.h>
#define MAX 1000
#define BASE 10
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;
}
void countingSort(int arr[], int n, int exp) {
    int output[n];
    int count[BASE];
    for (int i = 0; i < BASE; i++) {
        count[i] = 0;
    }
    for (int i = 0; i < n; i++) {
        count[(arr[i] / exp) % BASE]++;
    }
    for (int i = 1; i < BASE; i++) {
        count[i] += count[i - 1];
    }
    for (int i = n - 1; i >= 0; i--) {
        output[count[(arr[i] / exp) % BASE] - 1] = arr[i];
        count[(arr[i] / exp) % BASE]--;
    }
    for (int i = 0; i < n; i++) {
        arr[i] = output[i];
```



```c
    }
}
void radixSort(int arr[], int n) {
    int max = getMax(arr, n);
    for (int exp = 1; max / exp > 0; exp *= BASE) {
        countingSort(arr, n, exp);
    }
}
void printArray(int arr[], int size) {
    for (int i = 0; i < size; 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("Unsorted array: \n");
    printArray(arr, n);
    radixSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```