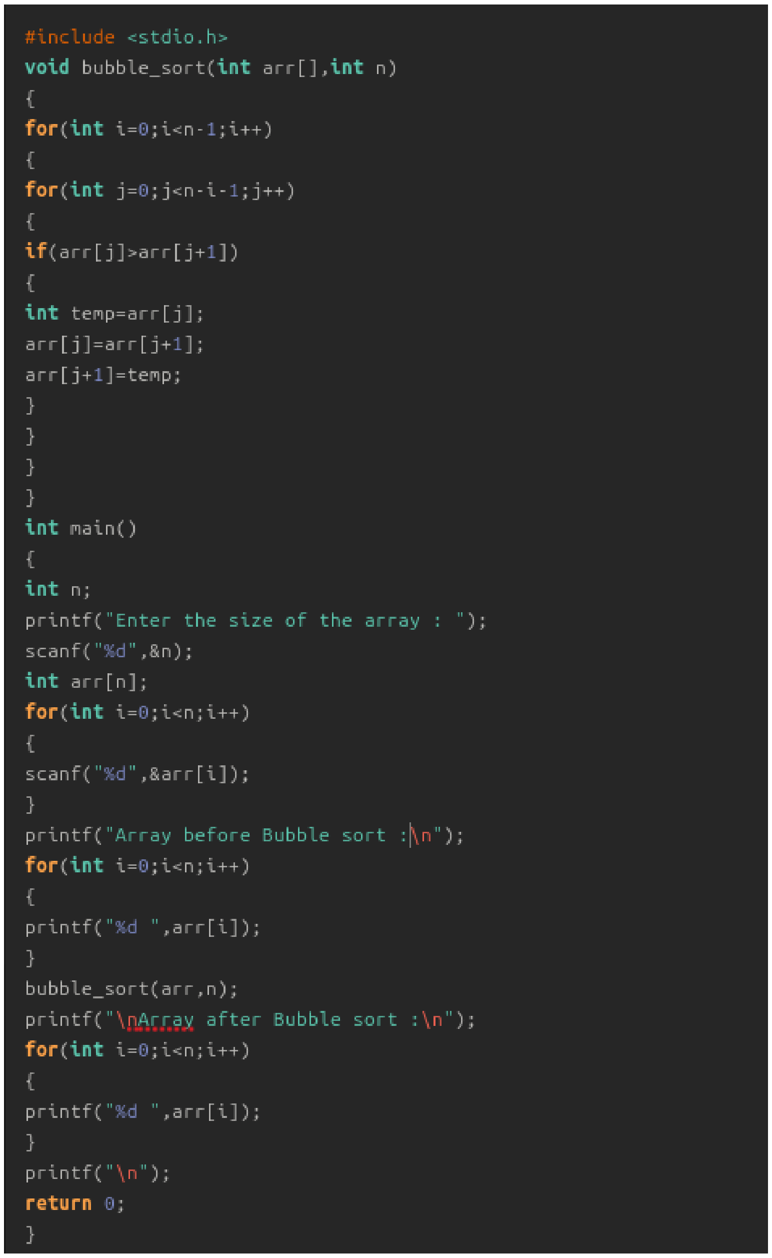
23CSE211-DESGIN AND ANALYSIS OF ALOGARTHIMS

LAB –2(Sorting techniques)

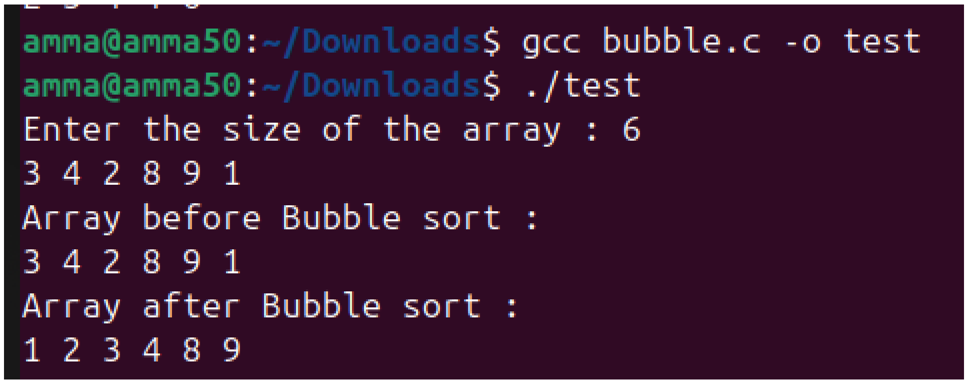
1.Bubble sort

Code:



```
#include <stdio.h>
void bubble_sort(int arr[],int n)
{
for(int i=0;i<n-1;i++)
{
for(int j=0;j<n-i-1;j++)
{
if(arr[j]>arr[j+1])
{
int temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
}
}
}
}
int main()
{
int n;
printf("Enter the size of the array : ");
scanf("%d",&n);
int arr[n];
for(int i=0;i<n;i++)
{
scanf("%d",&arr[i]);
}
printf("Array before Bubble sort :\n");
for(int i=0;i<n;i++)
{
printf("%d ",arr[i]);
}
bubble_sort(arr,n);
printf("\nArray after Bubble sort :\n");
for(int i=0;i<n;i++)
{
printf("%d ",arr[i]);
}
printf("\n");
return 0;
}
```

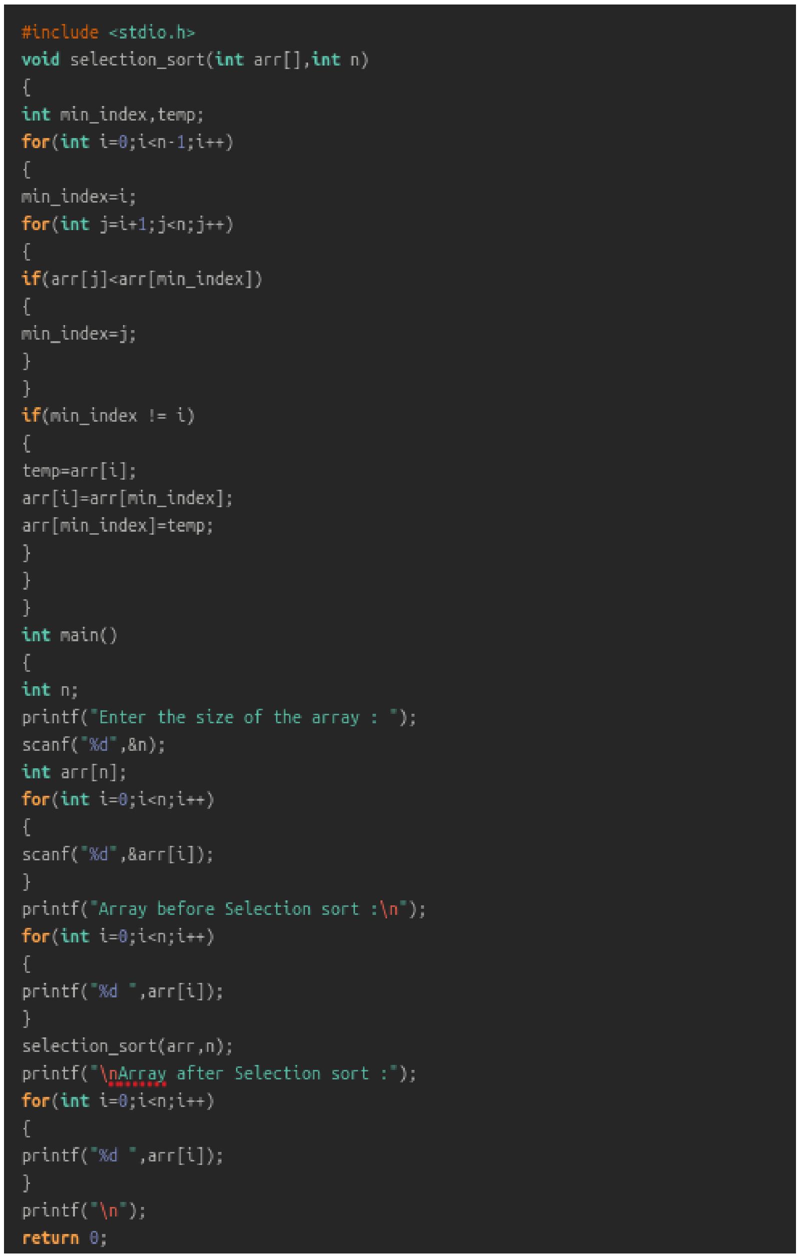
Output:



```
amma@amma50:~/Downloads$ gcc bubble.c -o test
amma@amma50:~/Downloads$ ./test
Enter the size of the array : 6
3 4 2 8 9 1
Array before Bubble sort :
3 4 2 8 9 1
Array after Bubble sort :
1 2 3 4 8 9
```

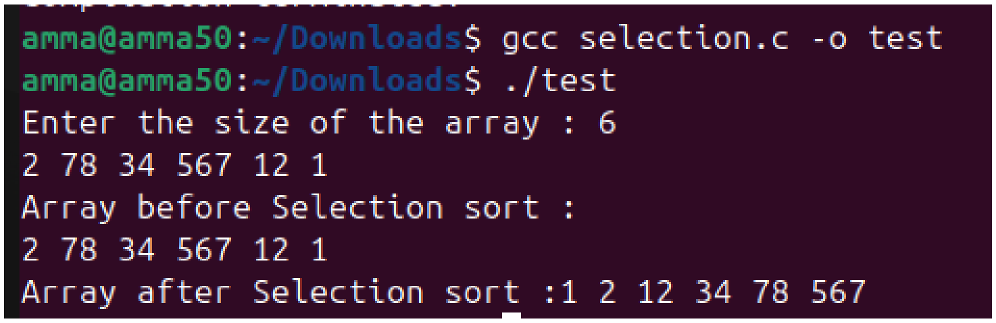
2.Selection sort

Code:



```
#include <stdio.h>
void selection_sort(int arr[],int n)
{
int min_index,temp;
for(int i=0;i<n-1;i++)
{
min_index=i;
for(int j=i+1;j<n;j++)
{
if(arr[j]<arr[min_index])
{
min_index=j;
}
}
if(min_index != i)
{
temp=arr[i];
arr[i]=arr[min_index];
arr[min_index]=temp;
}
}
}
int main()
{
int n;
printf("Enter the size of the array : ");
scanf("%d",&n);
int arr[n];
for(int i=0;i<n;i++)
{
scanf("%d",&arr[i]);
}
printf("Array before Selection sort :\n");
for(int i=0;i<n;i++)
{
printf("%d ",arr[i]);
}
selection_sort(arr,n);
printf("\nArray after Selection sort :");
for(int i=0;i<n;i++)
{
printf("%d ",arr[i]);
}
printf("\n");
return 0;
```

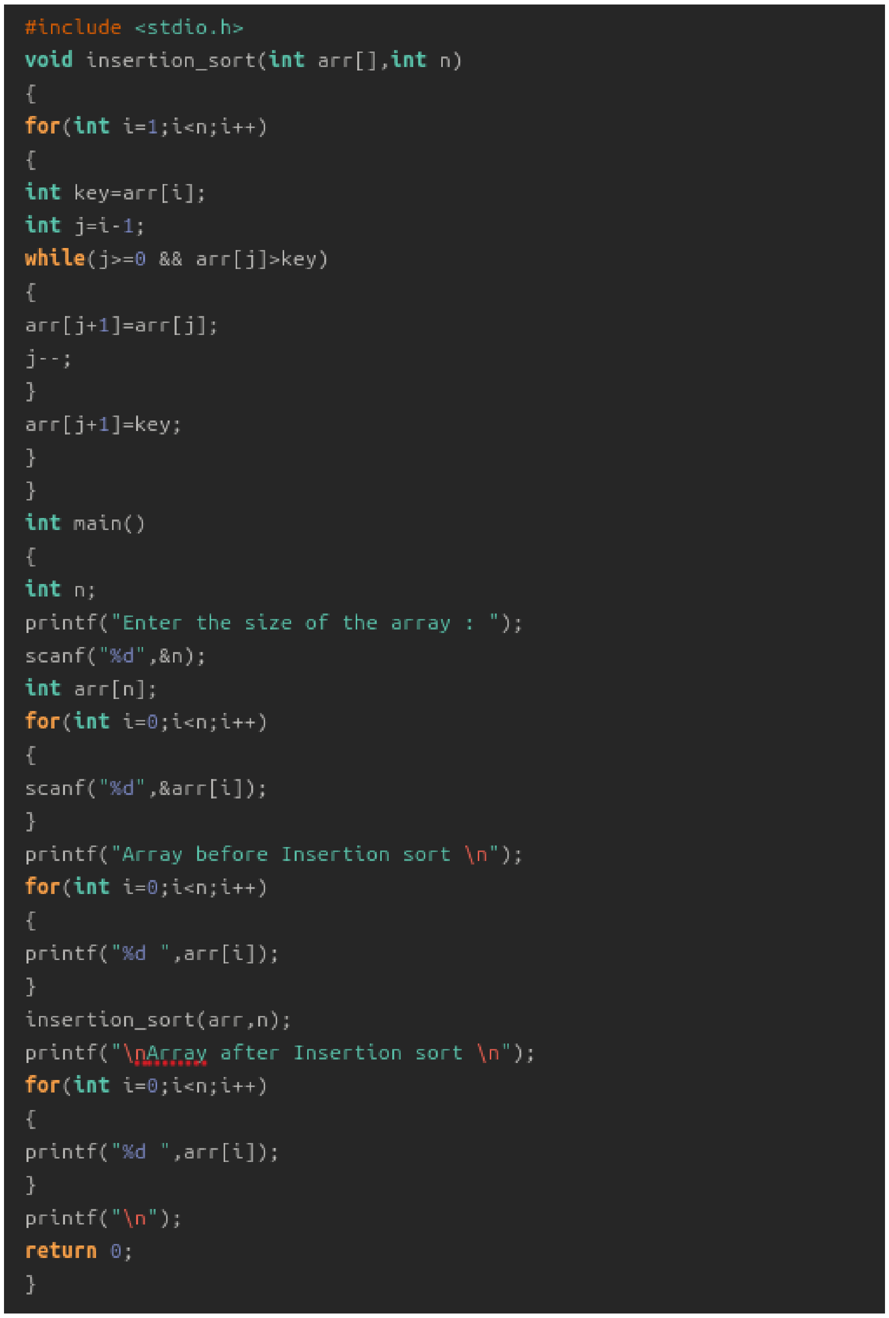
Output:’



```
amma@amma50:~/Downloads$ gcc selection.c -o test
amma@amma50:~/Downloads$ ./test
Enter the size of the array : 6
2 78 34 567 12 1
Array before Selection sort :
2 78 34 567 12 1
Array after Selection sort :1 2 12 34 78 567
```

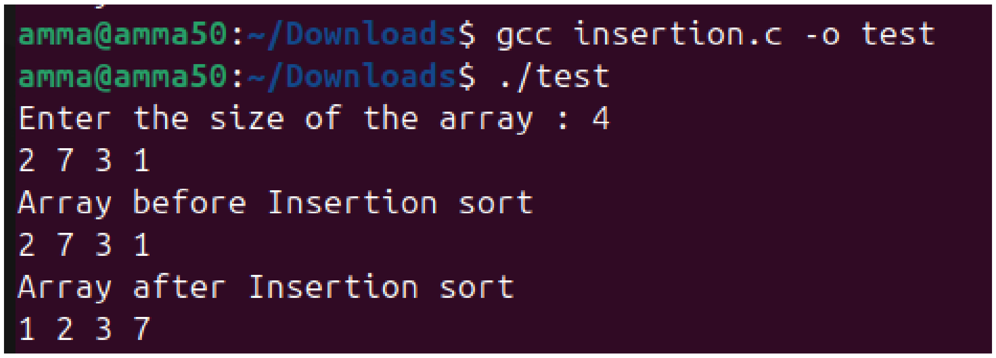
3.Insertion sort:

Code:



```
#include <stdio.h>
void insertion_sort(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--;
}
arr[j+1]=key;
}
}
int main()
{
int n;
printf("Enter the size of the array : ");
scanf("%d",&n);
int arr[n];
for(int i=0;i<n;i++)
{
scanf("%d",&arr[i]);
}
printf("Array before Insertion sort \n");
for(int i=0;i<n;i++)
{
printf("%d ",arr[i]);
}
insertion_sort(arr,n);
printf("\nArray after Insertion sort \n");
for(int i=0;i<n;i++)
{
printf("%d ",arr[i]);
}
printf("\n");
return 0;
}
```

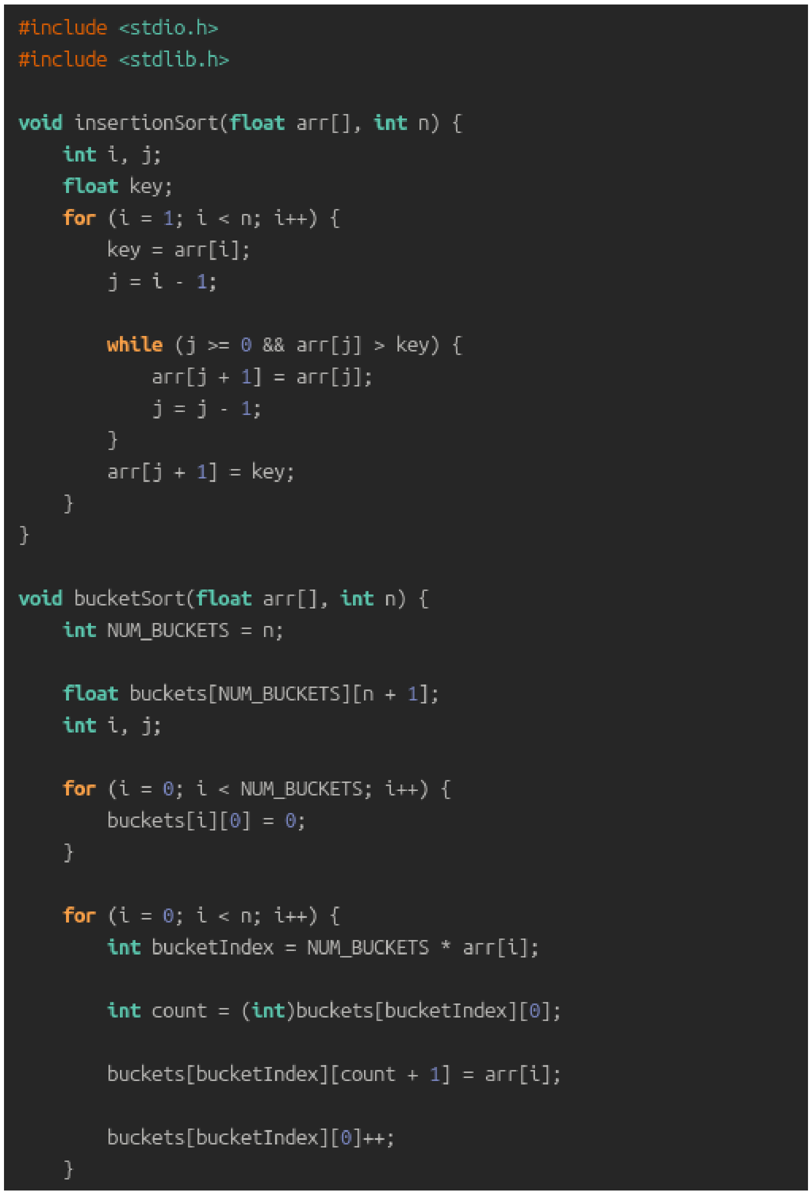
Output:



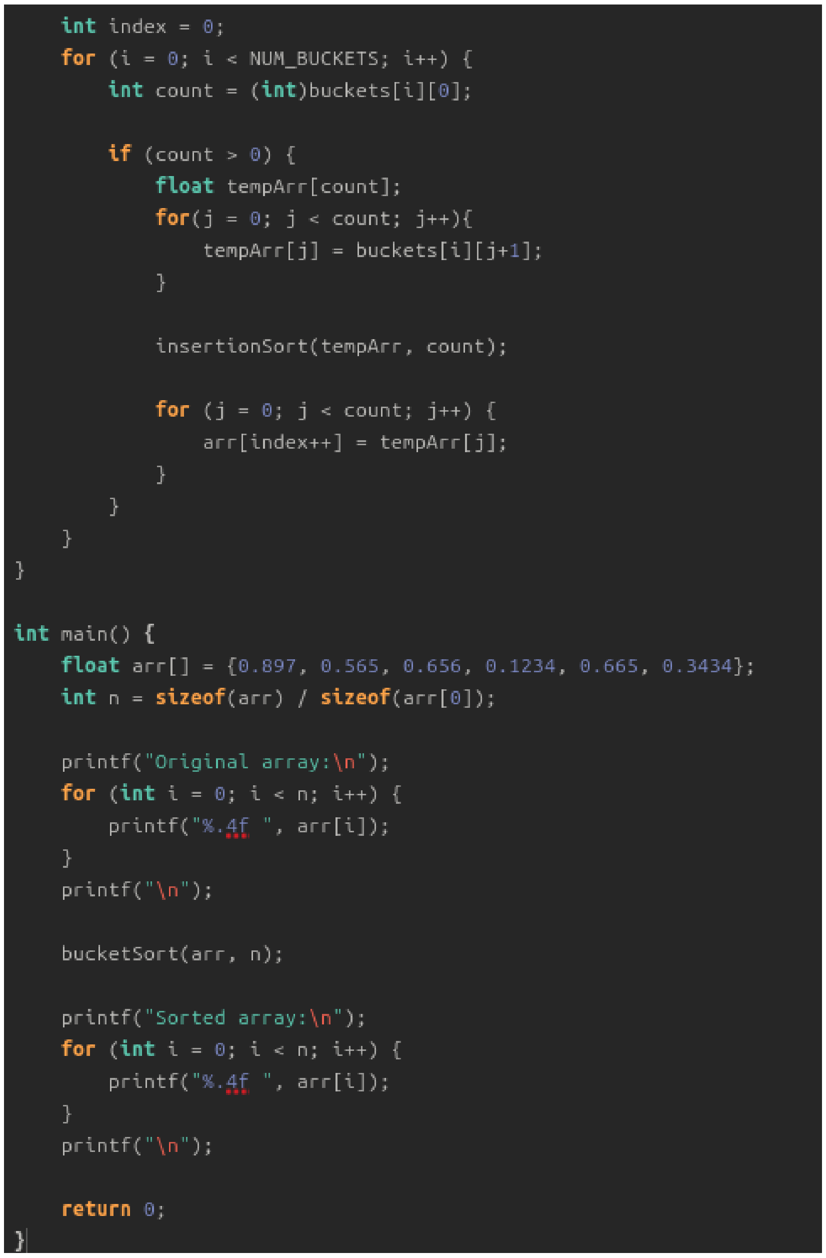
```
amma@amma50:~/Downloads$ gcc insertion.c -o test
amma@amma50:~/Downloads$ ./test
Enter the size of the array : 4
2 7 3 1
Array before Insertion sort
2 7 3 1
Array after Insertion sort
1 2 3 7
```

4.Bucket sort

Code:

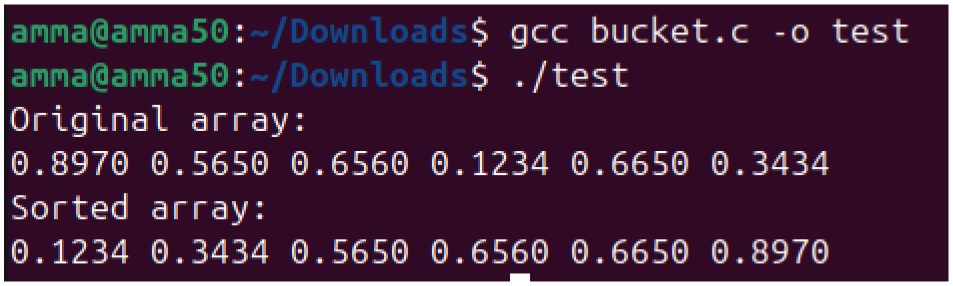


```c
#include <stdio.h>
#include <stdlib.h>

void insertionSort(float arr[], int n) {
    int i, j;
    float key;
    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 bucketSort(float arr[], int n) {
    int NUM_BUCKETS = n;

    float buckets[NUM_BUCKETS][n + 1];
    int i, j;

    for (i = 0; i < NUM_BUCKETS; i++) {
        buckets[i][0] = 0;
    }

    for (i = 0; i < n; i++) {
        int bucketIndex = NUM_BUCKETS * arr[i];

        int count = (int)buckets[bucketIndex][0];

        buckets[bucketIndex][count + 1] = arr[i];

        buckets[bucketIndex][0]++;
    }
```



```c
    int index = 0;
    for (i = 0; i < NUM_BUCKETS; i++) {
        int count = (int)buckets[i][0];

        if (count > 0) {
            float tempArr[count];
            for(j = 0; j < count; j++){
                tempArr[j] = buckets[i][j+1];
            }

            insertionSort(tempArr, count);

            for (j = 0; j < count; j++) {
                arr[index++] = tempArr[j];
            }
        }
    }
}

int main() {
    float arr[] = {0.897, 0.565, 0.656, 0.1234, 0.665, 0.3434};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Original array:\n");
    for (int i = 0; i < n; i++) {
        printf("%.4f ", arr[i]);
    }
    printf("\n");

    bucketSort(arr, n);

    printf("Sorted array:\n");
    for (int i = 0; i < n; i++) {
        printf("%.4f ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Output:

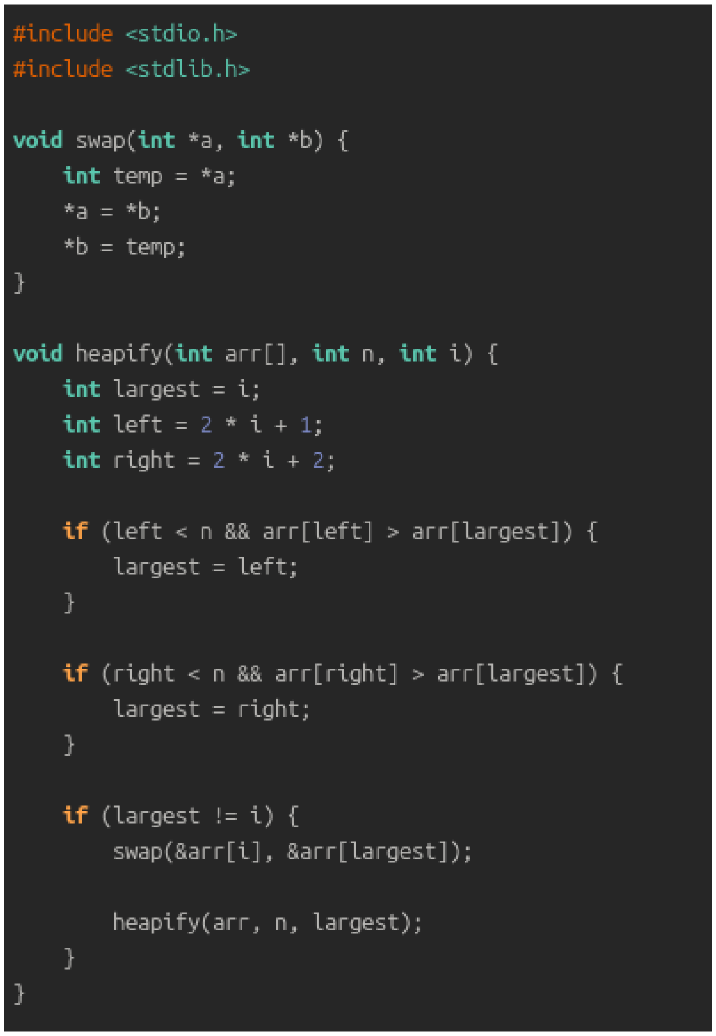


```
amma@amma50:~/Downloads$ gcc bucket.c -o test
amma@amma50:~/Downloads$ ./test
Original array:
0.8970 0.5650 0.6560 0.1234 0.6650 0.3434
Sorted array:
0.1234 0.3434 0.5650 0.6560 0.6650 0.8970
```

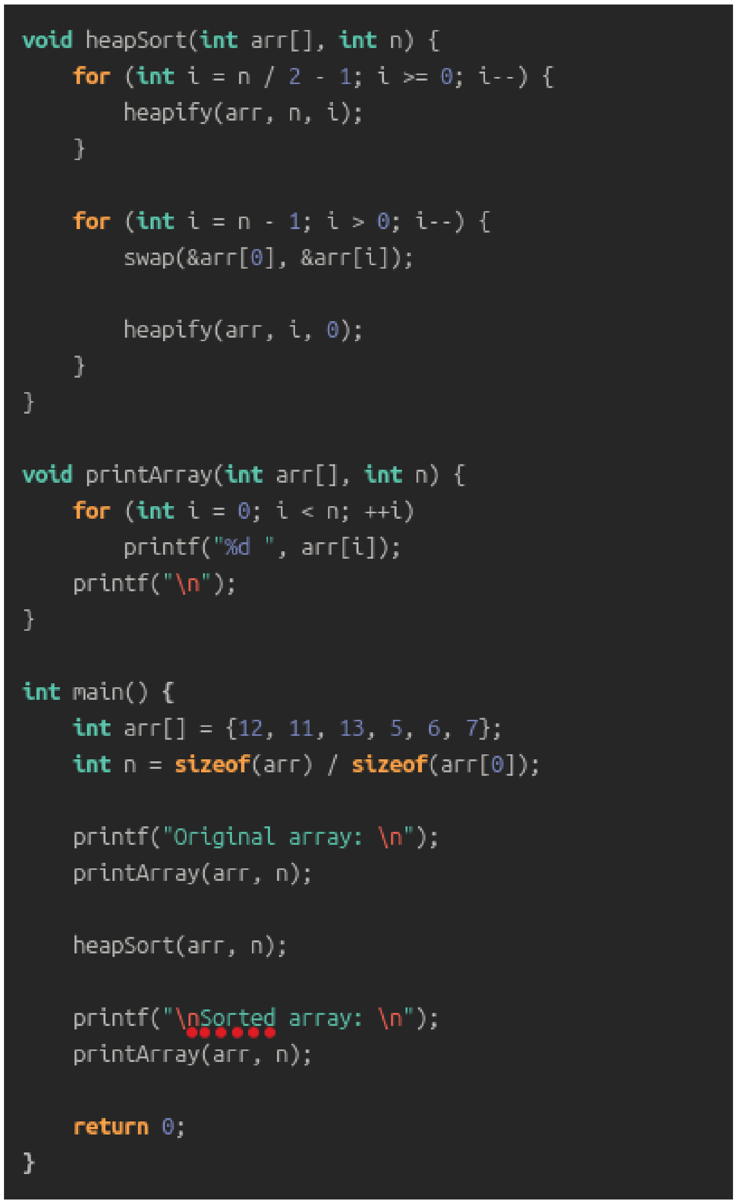
]

5.Heap sort

Code:

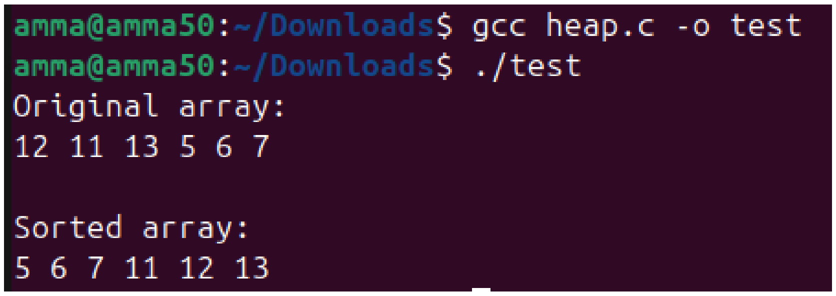


```c
#include <stdio.h>
#include <stdlib.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void heapify(int arr[], int n, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && arr[left] > arr[largest]) {
        largest = left;
    }

    if (right < n && arr[right] > arr[largest]) {
        largest = right;
    }

    if (largest != i) {
        swap(&arr[i], &arr[largest]);

        heapify(arr, n, largest);
    }
}
```



```c
void heapSort(int arr[], int n) {
    for (int i = n / 2 - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }

    for (int i = n - 1; i > 0; i--) {
        swap(&arr[0], &arr[i]);

        heapify(arr, i, 0);
    }
}

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

int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int n = sizeof(arr) / sizeof(arr[0]);

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

    heapSort(arr, n);

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

    return 0;
}
```

Output:



```
amma@amma50:~/Downloads$ gcc heap.c -o test
amma@amma50:~/Downloads$ ./test
Original array:
12 11 13 5 6 7

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