1. **Merge sort**

#include <stdio.h>

#include <stdlib.h>

void merge(int arr[], int left, int mid, int right)

{

int i, j, k;

int n1 = mid - left + 1;

int n2 = right - mid;

int leftarr[n1], rightarr[n2];

for (i = 0; i < n1; i++)

{

leftarr[i] = arr[left + i];

}

for (j = 0; j < n2; j++)

{

rightarr[j] = arr[mid + 1 + j];

}

i = 0;

j = 0;

k = left;

while (i < n1 && j < n2)

{

if (leftarr[i] <= rightarr[j])

{

arr[k] = leftarr[i];

i++;

}

else

{

arr[k] = rightarr[j];

j++;

}

k++;

}

while (i < n1)

{

arr[k] = leftarr[i];

i++;

k++;

}

while (j < n2)

{

arr[k] = rightarr[j];

j++;

k++;

}

}

void mergeSort(int arr[], int left, int right)

{

if (left < right)

{

int mid = left + (right - left) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

}

int main()

{

int arr[] = { 26,4,8,1,19,62,34 };

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

mergeSort(arr, 0, n - 1);

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

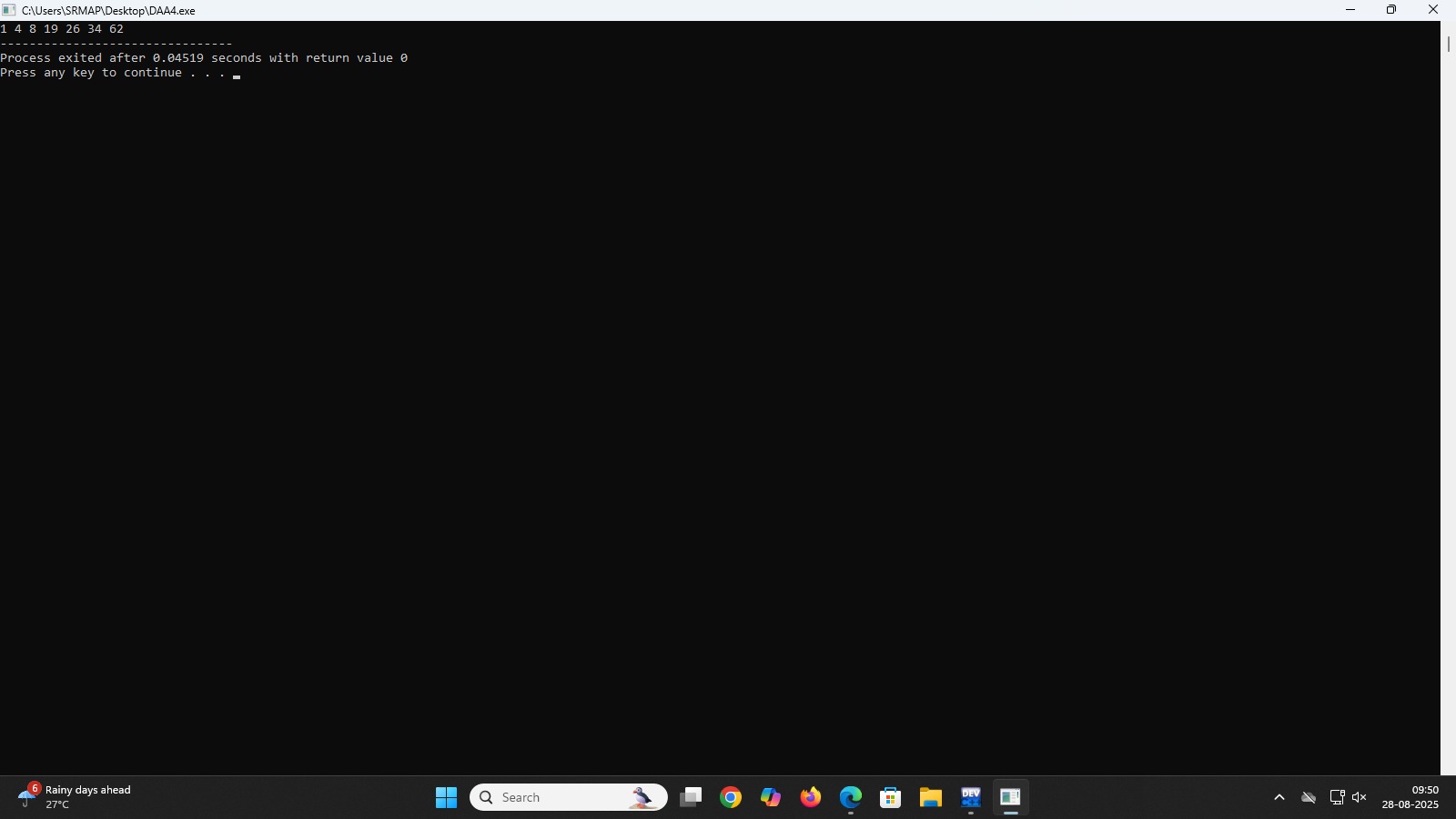
{

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

}

return 0;

}



#include<stdio.h>

1. **Quick sort**

void quicksort(int a[],int left,int right)

{

if(left>=right)return;

int pivot=a[right];

int i=left,j;

for(j=left;j<right;j++)

{

if(a[j]<pivot)

{

int temp=a[i];

a[i]=a[j];

a[j]=temp;

i++;

}

}

int temp=a[i];

a[i]=a[right];

a[right]=temp;

quicksort(a,left,i-1);

quicksort(a,i+1,right);

}

int main()

{

int a[]={4,3,98,76,43,80,54,18};

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

int i;

quicksort(a,0,n-1);

printf("sorted array: ");

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

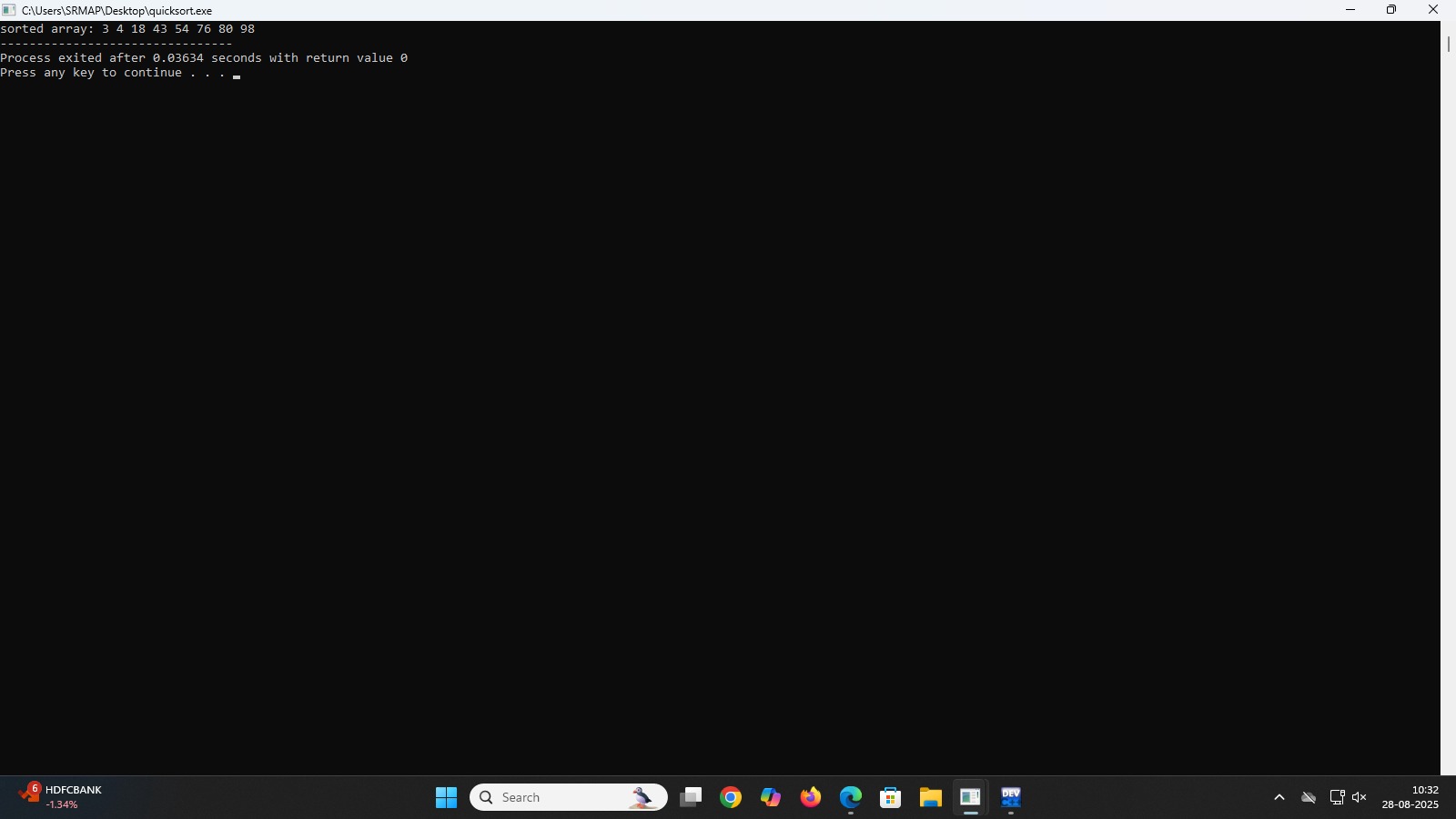
{

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

}

return 0;

}



1. **Bubble sort**

#include <stdio.h>

#include <stdlib.h>

void bucketsort(float arr[], int n)

{

float max\_val = arr[0];

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

{

if (arr[i] > max\_val)

max\_val = arr[i];

}

float norm\_arr[n];

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

{

norm\_arr[i] = arr[i] / max\_val;

}

float buckets[n][n];

int bucketcount[n];

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

{

bucketcount[i] = 0;

}

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

{

int bucketindex = (int)(n \* norm\_arr[i]);

if (bucketindex == n) bucketindex = n - 1;

buckets[bucketindex][bucketcount[bucketindex]++] = norm\_arr[i];

}

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

{

for (int j = 1; j < bucketcount[i]; j++)

{

float key = buckets[i][j];

int k = j - 1;

while (k >= 0 && buckets[i][k] > key)

{

buckets[i][k + 1] = buckets[i][k];

k--;

}

buckets[i][k + 1] = key;

}

}

int index = 0;

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

{

for (int j = 0; j < bucketcount[i]; j++)

{

arr[index++] = buckets[i][j] \* max\_val;

}

}

}

int main()

{

float arr[] = {4, 0.8, 14.5, 63, 6.65, 28.1, 7, 34, 5.4};

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

printf("Original array: ");

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

{

printf("%.2f ", arr[i]);

}

printf("\n");

bucketsort(arr, n);

printf("Sorted array: ");

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

{

printf("%.2f ", arr[i]);

}

printf("\n");

return 0;

}

