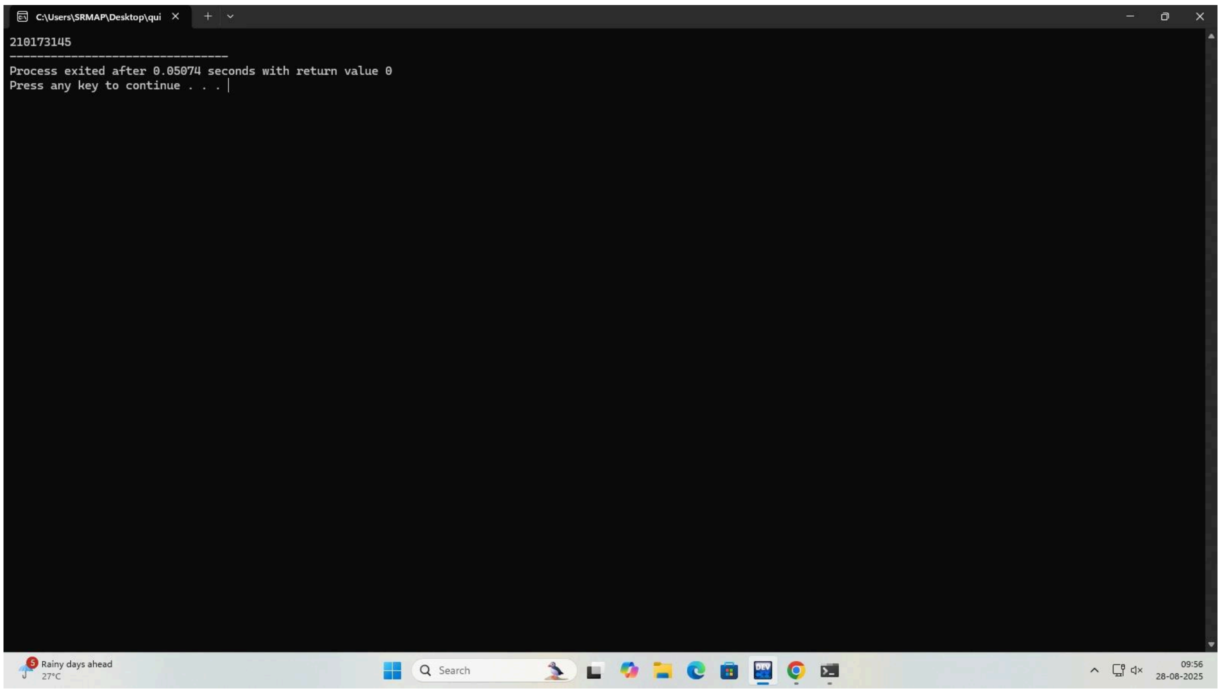
1.Quick sort

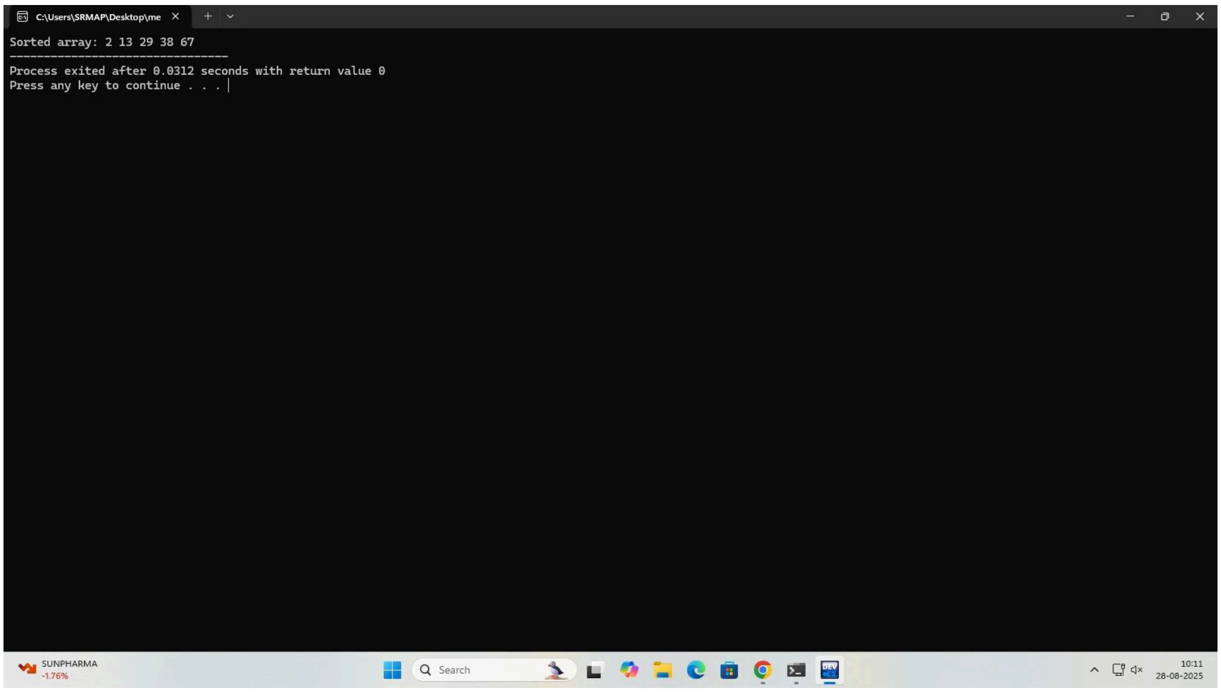
```
#include<stdio.h>
void qsort(int arr[],int left,int right)
{
        if(left>=right)
        return;
        int pivot=arr[right];
        int i=left;
        int j;
        for(j=left;j<right;j++)
        {
                if(arr[j]<pivot)
                {
                        int temp=arr[i];
                        arr[i]=arr[j];
                        arr[j]=temp;
                        i++;
                }
        }
        int temp=arr[i];
        arr[i]=arr[right];
        arr[right]=temp;
        qsort(arr,left,i-1);
        qsort(arr,i+1,right);
}
int main()
{
        int arr[]={10,2,45,17,31};
        int n=sizeof (arr)/sizeof(arr[0]);
        qsort(arr,0,n-1);
        for(int i=0;i<n;i++){
        printf("%d",arr[i]);
}
        return 0;
}
```



```
210173145
--------------------------------
Process exited after 0.05074 seconds with return value 0
Press any key to continue . . .
```

2.Merge sort

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

    while (j < n2)
        arr[k++] = R[j++];
}
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[] = {10, 2, 45, 17, 31};
    int n = sizeof(arr) / sizeof(arr[0]);

    mergeSort(arr, 0, n - 1);

    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

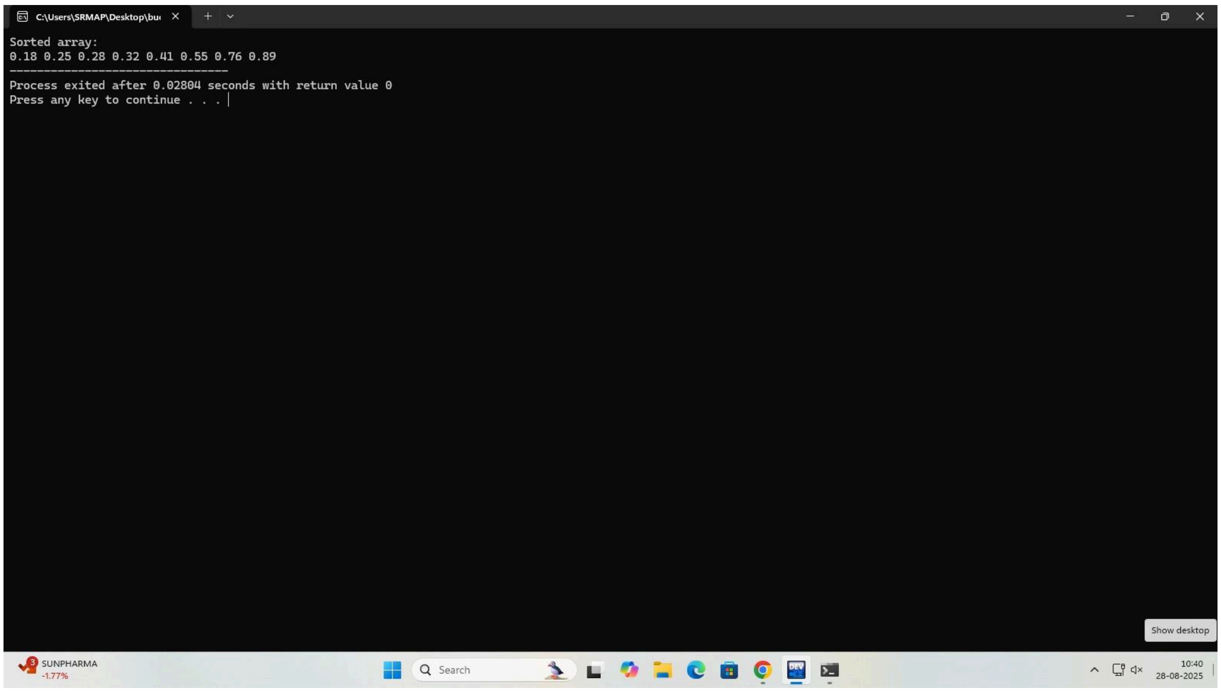


```
Sorted array: 2 13 29 38 67
--------------------------------
Process exited after 0.0312 seconds with return value 0
Press any key to continue . . .
```

3. Bucket sort

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
    float data;
    struct Node* next;
```

```
} Node;

Node* insertSorted(Node* head, float value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = NULL;

    if (head == NULL || value < head->data) {
        newNode->next = head;
        return newNode;
    }

    Node* current = head;
    while (current->next != NULL && current->next->data < value) {
        current = current->next;
    }

    newNode->next = current->next;
    current->next = newNode;
    return head;
}

void freeList(Node* head) {
    while (head) {
        Node* temp = head;
        head = head->next;
        free(temp);
    }
}

void bucketSort(float arr[], int n) {
    Node* buckets[n];
    for (int i = 0; i < n; i++)
        buckets[i] = NULL;
    for (int i = 0; i < n; i++) {
        int index = arr[i] * n;
        buckets[index] = insertSorted(buckets[index], arr[i]);
    }

    int idx = 0;
    for (int i = 0; i < n; i++) {
        Node* node = buckets[i];
        while (node != NULL) {
            arr[idx++] = node->data;
            node = node->next;
        }
        freeList(buckets[i]);
    }
}

int main() {
    float arr[] = {0.25, 0.36, 0.58, 0.41, 0.29, 0.22, 0.45, 0.79};
    int n = sizeof(arr) / sizeof(arr[0]);

    bucketSort(arr, n);

    printf("Sorted array:\n");
    for (int i = 0; i < n; i++)
        printf("%.2f ", arr[i]);

    return 0;
}
```



```
Sorted array:
0.18 0.25 0.28 0.32 0.41 0.55 0.76 0.89
--------------------------------
Process exited after 0.02804 seconds with return value 0
Press any key to continue . . .
```