MERGE SORT

```
/ Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
void merge(int arr[],int l,int mid,int h){
      int n1=mid-l+1;
      int n2=h-mid;
      int b[n1],c[n2];
       for(int i=0;i<n1;i++){
           b[i]=arr[l+i];

      }
       for(int j=0;j<n2;j++){
           c[j]=arr[mid+1+j];
      }
       int i=0,j=0,k=l;



      while(i<n1&&j<n2){
      if(b[i]<c[j]){
           arr[k++]=b[i++];
      }else{
           arr[k++]=c[j++];

      }
```

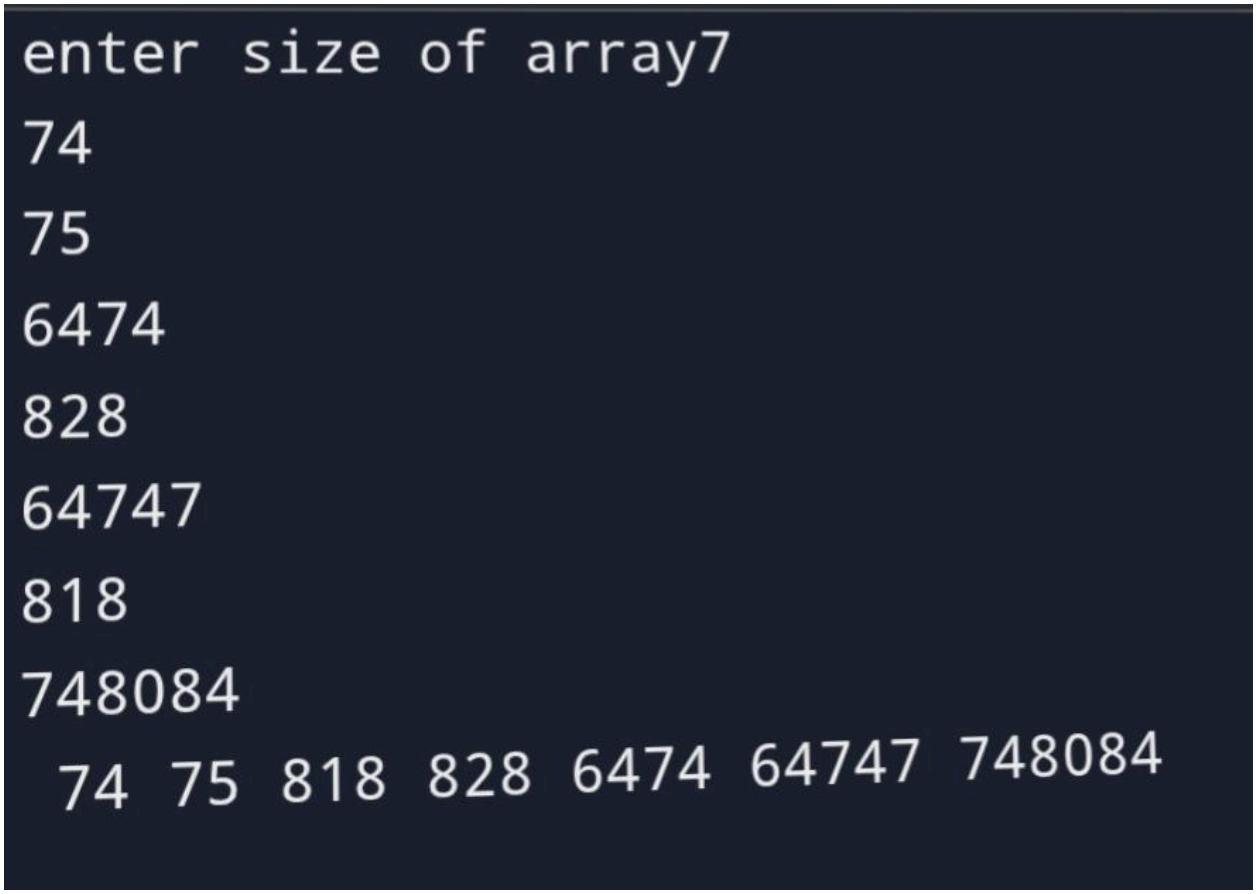
```
        }

        while(i<n1){
            arr[k++]=b[i++];
        }
        while(j<n2){
           arr[k++]=c[j++];
        }

}
void mergesort(int arr[],int l,int h){
    if(l<h){
        int mid=(l+h)/2;
        mergesort(arr ,l,mid);
        mergesort(arr,mid+1,h);
        merge(arr,l,mid,h);
    }
}
int main() {
    int n,l,h,mid;
    cout << "enter size of array";
    cin>> n;
    int arr[n];
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
```

```
    mergesort(arr,0,n-1);
    for(int y=0;y<n;y++)
    cout<<" "<<arr[y];
    return 0;
}
```

#include



QUICK SORT

```
void swap(int &a,int &b){
    int temp=a;
```

```
    a=b;
    b=temp;
}

int partition(int arr[],int l,int h){
    int pivot=arr[h];
    int i=l-1;
    for(int j=l;j<h;j++){
        if(arr[j]<pivot){
            i++;
            swap(arr[i],arr[j]);
        }
    }
    swap(arr[i+1],arr[h]);
    return i+1;
}

void quicksort(int arr[],int l,int h){
    if(l<h){
        int p=partition(arr,l,h);
        quicksort(arr,l,p-1);
        quicksort(arr,p+1,h);
    }
}

int main() {
```

```
int n;
cout << "Enter size of array: ";
cin >> n;

int arr[n];
cout << "Enter elements: ";
for(int i=0;i<n;i++)
    cin >> arr[i];

quicksort(arr,0,n-1);

cout << "Sorted array: ";
for(int i=0;i<n;i++)
    cout << arr[i] << " ";
cout << endl;

return 0;
```

```
}#include <iostream>
#include <vector>
#include <algorithm> // for sort
using namespace std;

void bucketSort(int arr[], int n) {
    // Step 1: Find max element (for range)
    int max_val = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max_val)
            max_val = arr[i];
    }

    // Step 2: Create buckets (size = n)
    vector<vector<int>> buckets(n);

    // Step 3: Put array elements into buckets
    for (int i = 0; i < n; i++) {
        int index = (arr[i] * n) / (max_val + 1); // bucket index
        buckets[index].push_back(arr[i]);
    }

    // Step 4: Sort each bucket
```

```
    for (int i = 0; i < n; i++) {
        sort(buckets[i].begin(), buckets[i].end());
    }

    // Step 5: Concatenate all buckets back to arr
    int k = 0;
    for (int i = 0; i < n; i++) {
        for (int val : buckets[i]) {
            arr[k++] = val;
        }
    }
}

int main() {
    int arr[] = {29, 25, 3, 49, 9, 37, 21, 43};
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Original array: ";
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << endl;

    bucketSort(arr, n);
```

```
    cout << "Sorted array:   ";
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << endl;

    return 0;
}
```