Mayank Gupta 20BCE1538\_\_DSA

AIM Randomized Quick and Merge Sort

# A set of children were made to stand in a line. Each child has been asked to pick a card which has a unique number (1 to n). The children have to rearrange themselves in ascending order according to the number they are holding. The arrangement has to begin by considering any of the children as a pivot element. Moreover the kids are supposed to use divide and conquer strategy for this arrangement process.

Write a program to help the kids arrange themselves.

# Note: The pivot element is the element of a matrix, or an array, which is selected first by an algorithm.

CODE;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std; void swap(int &x,int &y){

int temp=y; y=x; x=temp;

cout<<x<<" is swapped with "<<y<<endl;

}

int partition (int arr[], int low, int high)

{

cout<<"students in focus : ";

for(int i=low;i<=high;i++){

cout<<arr[i]<<" ";

}

cout<<endl;

int pivot = arr[high]; // pivot

cout<<"The Pivot element is (in this case last element) "<<pivot<<endl;

int i = (low - 1); // Index of smaller element and indicates the right position of pivot found so far

for (int j = low; j <= high - 1; j++)

{

// If current element is smaller than the pivot

if (arr[j] < pivot)

{

i++; // increment index of smaller element

swap(arr[i], arr[j]);

cout<<"Arrangement after swapping : ";

for(int i=low;i<=high;i++){

cout<<arr[i]<<" ";

}

cout<<endl;

}

}

swap(arr[i + 1], arr[high]);

cout<<"Arrangement after swapping : ";

for(int i=low;i<=high;i++){

cout<<arr[i]<<" ";

}

cout<<endl;

return (i + 1);

}

/\* The main function that implements QuickSort

arr[] --> Array to be sorted,

low --> Starting index,

high --> Ending index \*/

void quickSort(int arr[], int low, int high)

{

if (low < high)

{

/\* pi is partitioning index, arr[p] is now

at right place \*/

int pi = partition(arr, low, high);

// Separately sort elements before

// partition and after partition

cout<<"Arrangement after left division"<<endl;

quickSort(arr, low, pi - 1);

cout<<"Arrangement after right division"<<endl;

quickSort(arr, pi + 1, high);

}

}

int main()

{

int n;

cin>>n;

int arr[n];

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

int x;

cin>>x;

if(x<=n){

arr[i]=x;

}

else{

cout<<"students till n roll no. to be entered"<<endl;

}

}

cout<<"\*\*\*\*\*Quick Sort\*\*\*\*\*"<<endl;

quickSort(arr,0,n-1);

cout<<"Final arrangement of Students is : ";

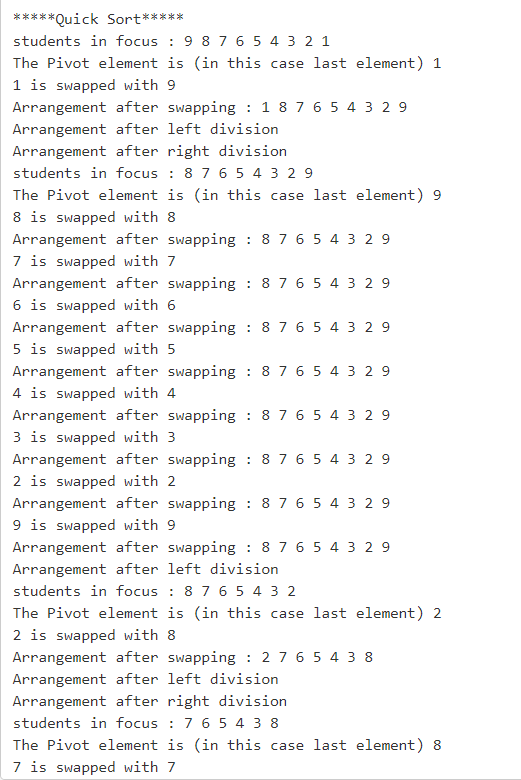
for(int i=0;i<n;i++){

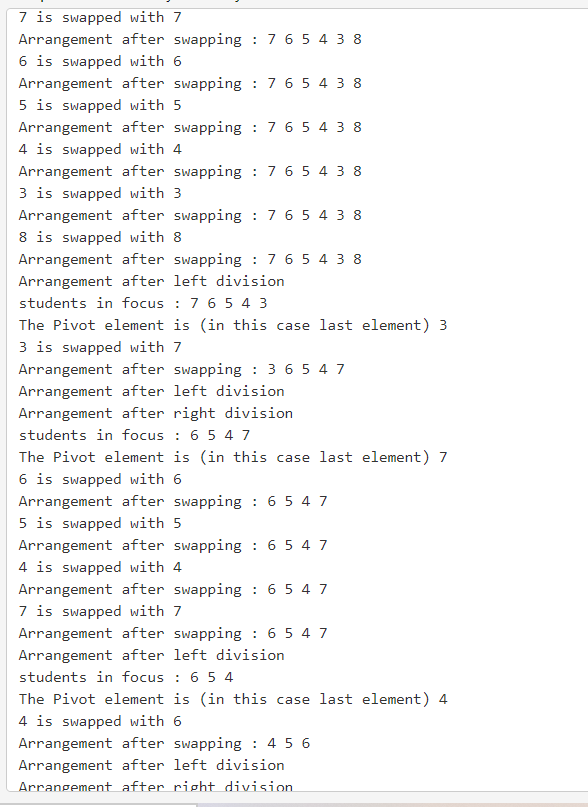
cout<<arr[i]<<" ";

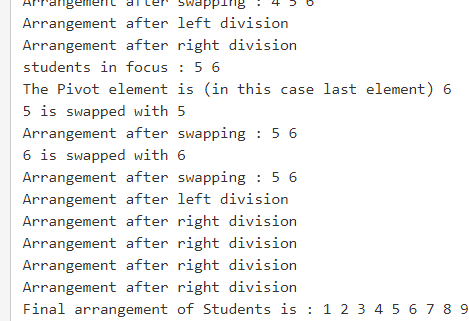
}

**return 0;**

**}**







ALGO

/\* low --> Starting index, high --> Ending index \*/ quickSort(arr[], low, high)

{

if (low < high)

{

/\* pi is partitioning index, arr[p] is now at right place \*/

pi = partition(arr, low, high);

quickSort(arr, low, pi - 1); // Before pi quickSort(arr, pi + 1, high); // After pi

}

# Microsoft organized an App fest for students wherein 10 students participated and posted applications. Students were given prizes based on the number of applications posted.

Arrange the students in the increasing order of applications using merge sort.

|  |  |
| --- | --- |
| Student Number | No.of App’s  posted |
| 1 | 4 |
| 2 | 6 |
| 3 | 7 |
| 4 | 2 |
| 5 | 5 |
| 6 | 4 |
| 7 | 1 |
| 8 | 8 |
| 9 | 3 |
| 10 | 9 |

Code;

**#include<iostream> using namespace std; struct stud{**

**int n,ap;**

**};**

**void copy(struct stud &a,struct stud &b){ a.ap=b.ap;**

**a.n=b.n;**

**}**

**void merge(struct stud ar[],int l,int r,int mid){ int i=l,j=mid+1,k=l;**

struct stud b[100];

while(i<=mid && j<=r){

if(ar[i].ap<ar[j].ap){

copy(b[k],ar[i]);

i++;k++;

}else{

copy(b[k],ar[j]);

j++;k++;

}

}if(i>mid){

while(j<=r){

copy(b[k],ar[j]);

j++;k++;

}

}else{

while(i<=mid){

copy(b[k],ar[i]);

i++;k++;

}

}

for(k=l;k<=r;++k) copy(ar[k],b[k]);

}

void mergesort(struct stud ar[],int l, int r){

if(l<r){

int mid=(l+r)/2;

**mergesort(ar,l,mid); mergesort(ar,mid+1,r); merge(ar,l,r,mid);**

**}**

**}**

**int main(){**

**int n;cout<<"Number of Students\n"; cin>>n;**

**struct stud apps[n]; for(int i=0;i<n;++i){**

**cout<<"Apps made by Student No. "<<i+1<<": "; cin>>apps[i].ap;**

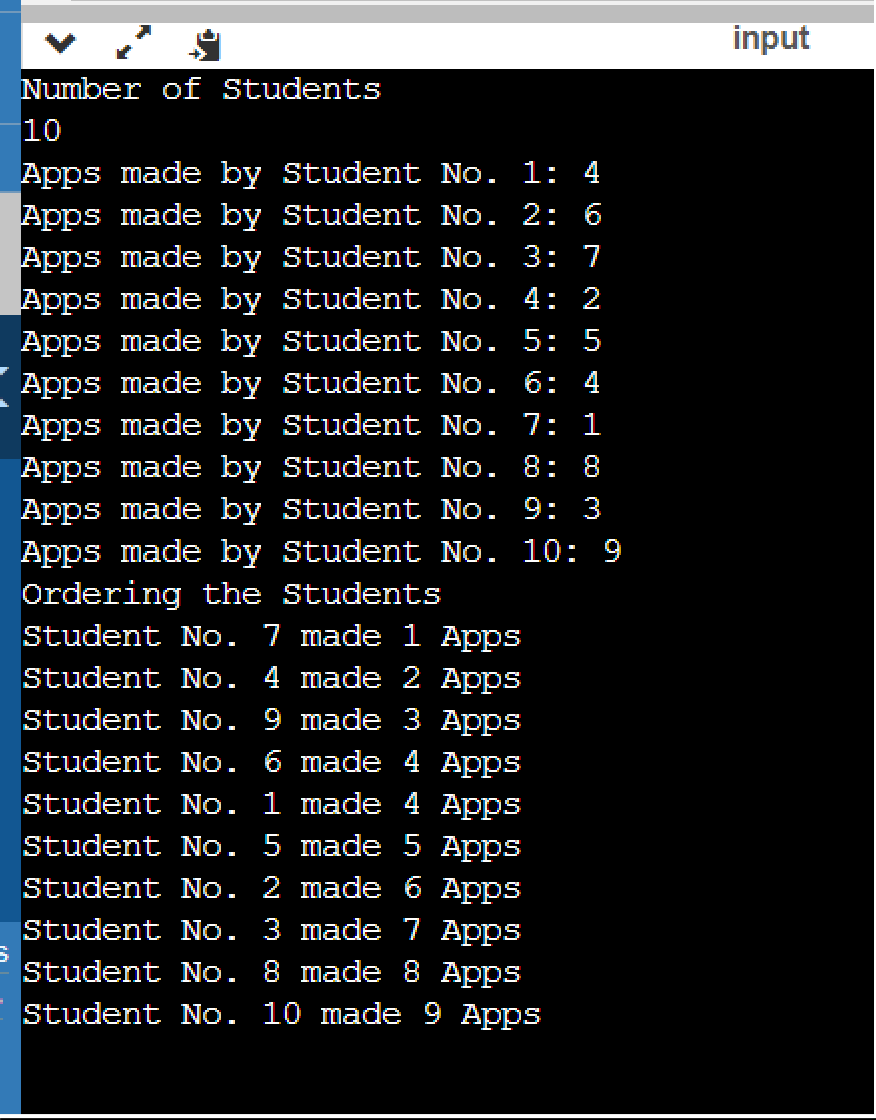
**apps[i].n=i+1;**

**}**

**mergesort(apps,0,n-1); cout<<"Ordering the Students\n";**

**for(int i=0;i<n;++i) cout<<"Student No. "<<apps[i].n<<" made "<<apps[i].ap<<" Apps\n"; cout<<endl;**

**}**



Algo

**MergeSort(arr[], l, r)**

If r > l

1. Find the middle point to divide the array into two halves: middle m = l+ (r-l)/2
2. Call mergeSort for first half: Call mergeSort(arr, l, m)
3. Call mergeSort for second half: Call mergeSort(arr, m+1, r)
4. Merge the two halves sorted in step 2 and 3: Call merge(arr, l, m, r)