ASSIGNMENT 4 (Quick Sort)

CODE :

```
//Merwin Pinto
//202100102
//Div D
#include<stdio.h>
int elementpartion(int array[] ,int low , int high);
void quicksort(int array[],int lower , int higher) ;
void swap(int* a, int* b);
void printarray(int array[], int size);
int main()        //main function
{
   printf("Program on QUICK SORT ! \n\n");
   int n,i,array[100];
   printf("Enter for how many elements you need : ");
   scanf("%d",&n);
   for(i=0;i<n;i++)                        //no of elements
   {
     scanf("%d",&array[i]);                  //array of elements to be sorted
   }
   printf("Unsorted array : \n");
   printarray(array, n);           //printing unsorted array

   printf("\n\nSorted array: \n");
   quicksort(array, 0, n - 1);
   printarray(array, n);            //printing sorted array
```

```
   return 0;
}
void quicksort(int array[],int lower , int higher)    //quicksort algorithm
{
   int index;
   if(lower < higher)
   {
      index = elementpartion(array ,lower ,higher);
      quicksort( array, lower, index - 1);
      quicksort( array, index + 1, higher);
   }
}
int elementpartion(int array[] ,int lower , int higher)      //seperation of
elements algorithm
{
   int i, j, k;
   int   pivot;            //declaring pivot
   pivot = array[higher] ;   //pivot assigned to higher value
   i = lower - 1;
   k = lower;
   for(j = k ; j<= higher-1;j++)
   {
      if(array[j]<= pivot)               //comparing elements with pivot
      {
         i++;
         swap(&array[i],&array[j]);       //swapping the correct compared values
                                          //call by reference
```

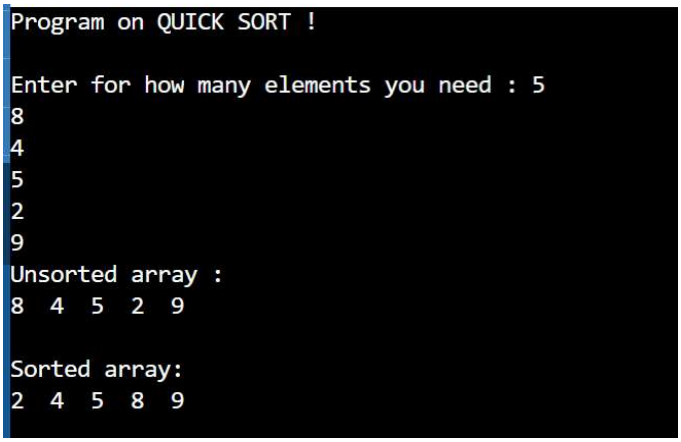
```
        }
    }
    swap(&array[i+1],&array[higher]);
    return(i+1);
}

void swap(int* a, int* b)               //function for swapping elements
{                                       //pass by reference
    int temp = *a;
    *a = *b;
    *b = temp;
}




void printarray(int array[], int n)      //function for printing array
{
    int i;
    for (i = 0; i < n; i++)
    {
        printf( "%d \t", array[i]);
    }
}
```

OUTPUT :


Program on QUICK SORT !
Enter for how many elements you need : 5
8
4
5
2
9
Unsorted array :
8 4 5 2 9
Sorted array:
2 4 5 8 9

ASSIGNMENT 4 (MERGE SORT)

CODE :

```
//Merwin Pinto
//202100102
//Div D
//Merge sort
#include <stdio.h>
void merge(int array[], int First, int middle, int last) ;
void mergeSort(int array[], int First, int last);
void display(int array[], int n);
int main()
{
   printf("program for MERGE SORT !\n\n");
   int n,i, array[100];
   printf("Enter for how many elements you need : ");  //no of elements
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      scanf("%d",&array[i]);                    //array of elements to be sorted
   }
   printf("\nUnsorted Array : \n");              //Unsorted elements
   display(array, n);

   printf("\nSorted Array \n");             //sorted elements
   mergeSort(array, 0, n - 1);
   display(array, n);
   return 0;
}
void merge(int array[], int First, int middle, int last)
{
```

```
int i, j, k;
int n1 = middle - First + 1;
int n2 = last - middle;

int LeftArray[n1], RightArray[n2];

for (i = 0; i < n1; i++)
{
   LeftArray[i] = array[First + i];      //left array distribution
}

for (j = 0; j < n2; j++)
{
   RightArray[j] = array[middle + 1 + j];   //right array distribution
}



i = 0;      // first sub-array
j = 0;      // second sub-array
k = First;  // sub-array

while (i < n1 && j < n2)
{
   if(LeftArray[i] <= RightArray[j])         //comparing elemts of left and right array
   {
      array[k] = LeftArray[i];
      i++;
   }
   else
   {
      array[k] = RightArray[j];
```

```
        j++;
      }
      k++;
   }

   while (i<n1)
   {
      array[k] = LeftArray[i];
      i++;
      k++;
   }

   while (j<n2)
   {
      array[k] = RightArray[j];
      j++;
      k++;
   }
}

void mergeSort(int array[], int First, int last)  //Merge sort Function holding algorithm
{
   int middle;
   if (First < last)
   {
      middle = (First + last) / 2;
      mergeSort(array, First, middle);
      mergeSort(array, middle + 1, last);
      merge(array, First, middle, last);
   }
}
```

```
void display(int array[], int n)    //function for printing array to be sorted
{
   int i;
   for (i = 0; i < n; i++)
   {
      printf("%d ", array[i]);
   }
   printf("\n");
}
```

OUTPUT :