Course Outcomes

At the end of the course, the student will be able to:

CO1. Identify the design technique used in an algorithm.
CO2. Analyse algorithms using quantitative evaluation.
CO3. Design and implement efficient algorithms for practical and unseen problems.
CO4. Analyse time efficiency over trading space.
CO5. Understand the limits of algorithms and the ways to cope with the limitations.

Question	Course Outcome	Topic
Q1		Merge Sort
Q2		Insertion Sort
Q3		Quick Sort

Ticking Session 5

NAME : NAGAVENI L G

SRN: PES2UG21CS315

SEC: 4F

Q1. Sort the roll numbers of n students given by the array rollnumber[] using lexicographic order, the sorting algorithm should execute in nlogn time and the order of duplicates have to be maintained. Print the sorted order.

Sample Input 1:
4
3 3 1 2
Expected Output 1:
1 2 3 3

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
4
3 3 1 2
1 2 3 3
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

Boilerplate/Skeleton Code:

```
#include <stdio.h>
```

```
#include <stdlib.h>


void solution (int rollnumber[],int n){
        //Write your solution here

}

int main()
{
    // Driver code
        int n;
        scanf("%d",&n);
        int arr[n];
        for(int i=0;i<n;i++){

                scanf("%d",&arr[i]);

        }

   solution(arr,n);

   return 0;
}
```

```c
#include <stdio.h>

void printArray(int *A, int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("%d ", A[i]);
    }
    printf("\n");
}

void merge(int A[], int mid, int low, int high)
{
    int i, j, k, B[100];
    i = low;
    j = mid + 1;
    k = low;

    while (i <= mid && j <= high)
    {
        if (A[i] < A[j])
```

```
        {
            B[k] = A[i];
            i++;
            k++;
        }
        else
        {
            B[k] = A[j];
            j++;
            k++;
        }
    }
    while (i <= mid)
    {
        B[k] = A[i];
        k++;
        i++;
    }
    while (j <= high)
    {
        B[k] = A[j];
        k++;
        j++;
    }
    for (int i = low; i <= high; i++)
    {
        A[i] = B[i];
    }

}

void solution(int rollnumber[], int low, int high){
    int mid;
    if(low<high){
        mid = (low + high) /2;
        solution(rollnumber, low, mid);
        solution(rollnumber, mid+1, high);
        merge(rollnumber, mid, low, high);
    }
}

int main()
{
    // int A[] = {9, 14, 4, 8, 7, 5, 6};
    int n;
    int arr[100];
    scanf("%d",&n);
    for(int i=0;i<n;i++){
```

```
        scanf("%d",&arr[i]);
    }

    solution(arr, 0, n-1);
    printArray(arr, n);
    return 0;
}
```

Test cases:
1.
2
1 2
Output -
1 2

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
2
1 2
1 2
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

2.
3
2 1 3

Output -
1 2 3

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
3
2 1 3
1 2 3
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

3.
4
4 4 2 3
Output -
2 3 4 4

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
4
4 4 2 3
2 3 4 4
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

Q2 . Students in a school standing in a line for the morning assembly are represented by the array studentline of size N, the values in the array studentline represents the height of each student. The instructor for the students wants to arrange their positions in the ascending order of their heights, to accomplish this he uses insertion sort. Count the number of swaps the instructor makes to accomplish his task. Assume all heights of students are unique.

Sample Input 1:

5
1 2 3 4 5

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
5
1 2 3 4 5
0
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

Expected Output 1:

0

Boilerplate/Skeleton Code:

```
#include <math.h>
#include <stdio.h>


void find_swaps(int studentline[], int N)
{
   //write your code here

}
```

```
void main()
{
   //Drivers Code
    int n;
   scanf("%d",&n);
   int arr[n];
   for(int i=0;i<n;i++){
       scanf("%d",&arr[i]);
   }
   find_swaps(arr,5);

}
```

```
#include <math.h>
#include <stdio.h>


void find_swaps(int sl[], int n)
{
    int key,j;
    int count=0;
    //write your code here
   for(int i=1;i<n;i++){
    key=sl[i];
    j=i-1;
    while(sl[j]>key){
       sl[j+1]=sl[j];
       j--;
       count++;
    }
    sl[j+1]=key;

   }
   printf("%d",count);
}




void main()
{
    //Drivers Code
     int n;
    scanf("%d",&n);
    int arr[n];
    for(int i=0;i<n;i++){
```

```
        scanf("%d",&arr[i]);
    }
    find_swaps(arr,n);

}
```

Test cases:
1.

5
5 3 1 2 4

Output -

6

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
5
5 3 1 2 4
6
```

2.

4
3 1 2 4

Output -

2

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
4
3 1 2 4
2
```

3.

4
7 1 2 3

Output -

3

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
4
7 1 2 3
3
```

Q3. A group of cars go together on a ride through the city of Hyderabad. Each car has an integer number on its number plate. The vehicles are initially in random order. The driver of the front car signals the remaining drivers behind them to overtake him and come ahead if their car number is greater than the signalling driver else they have to stay behind. The cars behind and ahead of the recent signalling driver then follow the same pattern as 2 separate groups (Leaving the signalling driver), (Assume the last number in the array to be the first car and second last to be the 2nd car and so on). Return the final sequence of car numbers when the groups are of size one. (Average case complexity - O(N log N))

Sample Input 1:
7 // Input Size
8 7 2 1 0 9 6 // Car numbers in the sequence of initial car arrangement

Expected Output 1:
0 1 2 6 7 8 9

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
7
8 7 2 1 0 9 6
0 1 2 6 7 8 9
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

Boilerplate/Skeleton Code:

```
#include <stdio.h>
```

Hint :

```
int partition(int array[], int low, int high)
{
   int pivot = array[high], i = (low - 1);

// Continue

}

void carsequence(int array[], int low, int high)
{
        // Write your code here
}

// Driver's code
int main()
{
   int n;
   scanf("%d", &n);
   int arr[n];
   for (int i = 0; i < n; i++)
      scanf("%d", &arr[i]);
   carsequence(arr, 0, n - 1);
   for (int i = 0; i < n; ++i)
   printf("%d ", arr[i]);
   printf("\n");
   return 0;
}
```

```c
#include <stdio.h>



int partition(int array[], int low, int high)
{int tmp1;
int tmp2;
    int pivot = array[low];
    int i=low;
    int j=high;
    do{
    while(array[i]<=pivot){
        i++;
    }
    while(array[j]>pivot){
        j--;
    }
    if(i<j)
```

```
    {tmp1=array[i];
    array[i]=array[j];
    array[j]=tmp1;
    }
    }while(i<j);
  //if j>i
    tmp2=array[low];
    array[low]=array[j];
    array[j]=tmp2;
// Continue
return j; //j is at correct p

}

void carsequence(int array[], int low, int high)
{int partition_val;
    // Write your code here
    if(low<high){
        partition_val=partition(array,low,high);
        carsequence(array,low,partition_val-1);
        carsequence(array,partition_val+1,high);
    }
}

// Driver's code
int main()
{
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    carsequence(arr, 0, n - 1);
    for (int i = 0; i < n; ++i)
    printf("%d ", arr[i]);
    printf("\n");
    return 0;
}
```

Test cases:
1.
3
3 2 1

Output -

1 2 3

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
3
3 2 1
1 2 3
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

2.
4
5 6 3 4

Output -
3 4 5 6

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
4
5 6 3 4
3 4 5 6
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

3.
5
7 3 5 2 1

Output -
1 2 3 5 7

```
PS C:\Users\Praka\OneDrive\Documents\DAA> ./a
5
7 3 5 2 1
1 2 3 5 7
PS C:\Users\Praka\OneDrive\Documents\DAA>
```

THANK YOU ☺