DESIGN ANALYSIS AND ALGORITHMS

LAB ASSIGNMENT-2

NAME : PRATHAPANI SATWIKA

REG.NO. : 20BCD7160

1) Implementation of Merge sort in java

CODE :

```
package Lab2a;
class MergeSort
{
	void merge(int arr[], int l, int m, int r)
	{
		int n1 = m - l + 1;
		int n2 = r - m;
		int L[] = new int[n1];
		int R[] = new int[n2];
		for (int i = 0; i < n1; ++i)
			L[i] = arr[l + i];
		for (int j = 0; j < n2; ++j)
			R[j] = arr[m + 1 + j];
		int i = 0, j = 0;
		int k = l;
		while (i < n1 && j < n2) {
			if (L[i] <= R[j]) {
				arr[k] = L[i];
				i++;
			}
			else {
				arr[k] = R[j];
				j++;
			}
			k++;
		}
		while (i < n1) {
			arr[k] = L[i];
			i++;
			k++;
		}
		while (j < n2) {
			arr[k] = R[j];
			j++;
			k++;
		}
	}
	void sort(int arr[], int l, int r)
	{
		if (l < r) {
```

```
                int m =l+ (r-l)/2;

                sort(arr, l, m);
                sort(arr, m + 1, r);
                merge(arr, l, m, r);
            }
        }
        static void printArray(int arr[])
        {
            int n = arr.length;
            for (int i = 0; i < n; ++i)
                System.out.print(arr[i] + " ");
            System.out.println();
        }
        public static void main(String args[])
        {
            int arr[] = { 25,55,98,67,34,8 };

            System.out.println("Given Array : ");
            printArray(arr);

            MergeSort ob = new MergeSort();
            ob.sort(arr, 0, arr.length - 1);

            System.out.println("\nSorted array : ");
            printArray(arr);
        }
}
```

```
package Lab2a;
class MergeSort
{
    void merge(int arr[], int l, int m, int r)
    {
        int n1 = m - l + 1;
        int n2 = r - m;
        int L[] = new int[n1];
        int R[] = new int[n2];
        for (int i = 0; i < n1; ++i)
            L[i] = arr[l + i];
        for (int j = 0; j < n2; ++j)
            R[j] = arr[m + 1 + j];
        int i = 0, j = 0;
        int k = l;
        while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                arr[k] = L[i];
                i++;
            }
            else {
                arr[k] = R[j];
                j++;
            }
            k++;
        }
        while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
        }
        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
    }
```

```
38      void sort(int arr[], int l, int r)
39      {
40          if (l < r) {
41              int m =l+ (r-l)/2;
42
43              sort(arr, l, m);
44              sort(arr, m + 1, r);
45              merge(arr, l, m, r);
46          }
47      }
48      static void printArray(int arr[])
49      {
50          int n = arr.length;
51          for (int i = 0; i < n; ++i)
52              System.out.print(arr[i] + " ");
53          System.out.println();
54      }
55      public static void main(String args[])
56      {
57          int arr[] = { 25,55,98,67,34,8 };
58
59          System.out.println("Given Array : ");
60          printArray(arr);
61
62          MergeSort ob = new MergeSort();
63          ob.sort(arr, 0, arr.length - 1);
64
65          System.out.println("\nSorted array : ");
66          printArray(arr);
67      }
68  }
69
```

OUTPUT :

```
<terminated> MergeSort [Java Application]
Given Array :
25 55 98 67 34 8

Sorted array :
8 25 34 55 67 98
```

2) Implementation of Quick sort in java

CODE :

```
package Lab2b;
import java.util.*;
public class QuickSort {
  public static int divide(int a[],int d,int u){
    int p = a[d];
    int i = d;
    int j = u;
    while (i < j)
    {
      while (a[i] <= p && i <= u - 1)
      {
        i++  ;
      }
      while (a[j] > p && j >= d)
      {
        j-- ;
      }
      if (i < j)
      {
        int t = a[i]  ;
        a[i] = a[j]  ;
        a[j] = t ;
      }
    }
    int t;
    t=a[j]  ;
    a[j] = a[d]  ;
    a[d] = t ;

    return j ;
  }
  static void quick_sort(int a[] , int d, int u){
    if (d<u)
    {
      int p;
      p=divide(a, d, u) ;
      quick_sort(a, d, p - 1)  ;
      quick_sort(a, p + 1, u)  ;

    }

  }
  public static void main(String[] args) {
    int i;
    int n=6 ;
    int a[] = {4,6,1,5,3,2}  ;
    System.out.println("Array before using quick sort method : ");
     for(i=0;i<n;i++){
      System.out.print(a[i]+","+" ")  ;
    }
    System.out.println();
```

```
    quick_sort(a,0,n-1)  ;
    System.out.println("Array after using quick sort method : ") ;
    for(i=0;i<n;i++){
      System.out.print(a[i]+","+" ")  ;
    }
  }
}
```

```
1  package Lab2b;
2  import java.util.*;
3  public class QuickSort {
4    public static int divide(int a[],int d,int u){
5      int p = a[d];
6      int i = d;
7      int j = u;
8      while (i < j)
9      {
10       while (a[i] <= p && i <= u - 1)
11       {
12         i++  ;
13       }
14       while (a[j] > p && j >= d)
15       {
16         j-- ;
17       }
18       if (i < j)
19       {
20         int t = a[i]  ;
21         a[i] = a[j]  ;
22         a[j] = t ;
23       }
24     }
25     int t;
26     t=a[j]  ;
27     a[j] = a[d]  ;
28     a[d] = t ;
29
30     return j ;
31   }
32   static void quick_sort(int a[] , int d, int u){
33     if (d<u)
34     {
35       int p;
36       p=divide(a, d, u) ;
37       quick_sort(a, d, p - 1)  ;
38       quick_sort(a, p + 1, u)  ;
39
40     }
41
42   }
```

```
  }
  public static void main(String[] args) {
    int i;
    int n=6 ;
    int a[] = {4,6,1,5,3,2}  ;
    System.out.println("Array before using quick sort method : ");
     for(i=0;i<n;i++){
      System.out.print(a[i]+","+" ")  ;
    }
    System.out.println();
    quick_sort(a,0,n-1)  ;
    System.out.println("Array after using quick sort method : ") ;
    for(i=0;i<n;i++){
      System.out.print(a[i]+","+" ")  ;
    }
  }
}
```

OUTPUT :

```
Array before using quick sort method :
4, 6, 1, 5, 3, 2,
Array after using quick sort method :
1, 2, 3, 4, 5, 6,
```