CSE 3004 DAA LAB TASK - 2

NAME : V.SAIKRISHNA
REG. NO :19BCE7638

MERGE SORT

Merge Sort is a divide and conquer algorithm. It works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

CODE:

```
public class Main
{
	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)/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[] = {32, 5, 55, 8, 62, 14};
        System.out.println("Given Array");
        printArray(arr);

        Main ob = new Main();
        ob.sort(arr, 0, arr.length-1);
        System.out.println("\nSorted array");
        printArray(arr);
```

```
    }
}
```

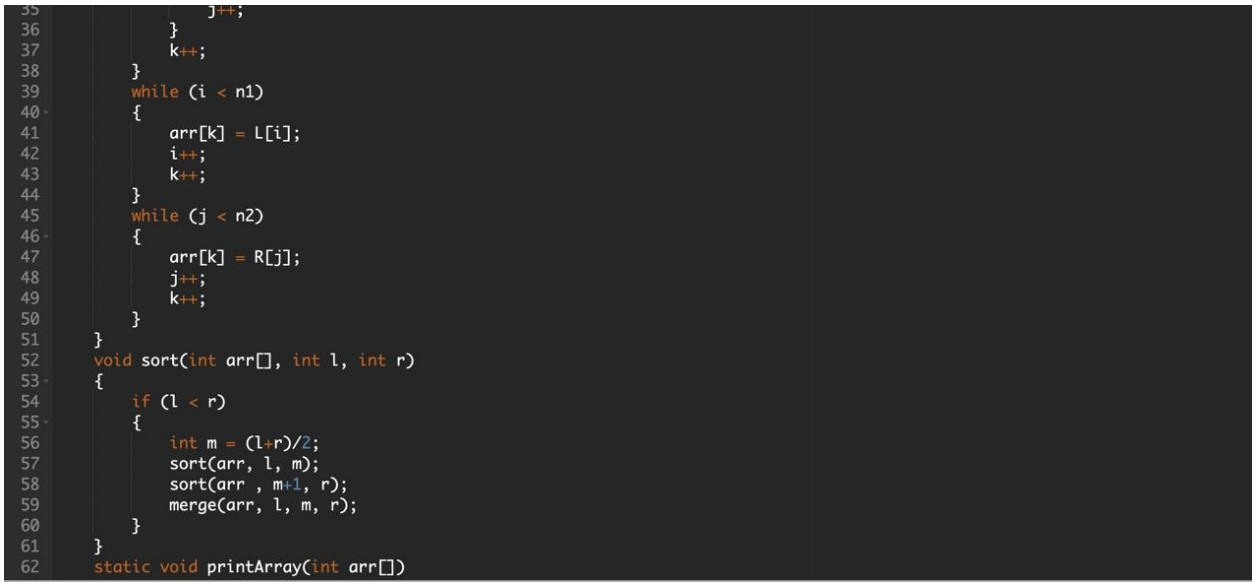
SCREENSHOTS:

```
public class Main
{
    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++;
```



```
                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)/2;
            sort(arr, l, m);
            sort(arr , m+1, r);
            merge(arr, l, m, r);
        }
    }
    static void printArray(int arr[])
```

```java
        if (l < r)
        {
            int m = (l+r)/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[] = {32, 5, 55, 8, 62, 14};
        System.out.println("Given Array");
        printArray(arr);

        Main ob = new Main();
        ob.sort(arr, 0, arr.length-1);
        System.out.println("\nSorted array");
        printArray(arr);
    }
}
```

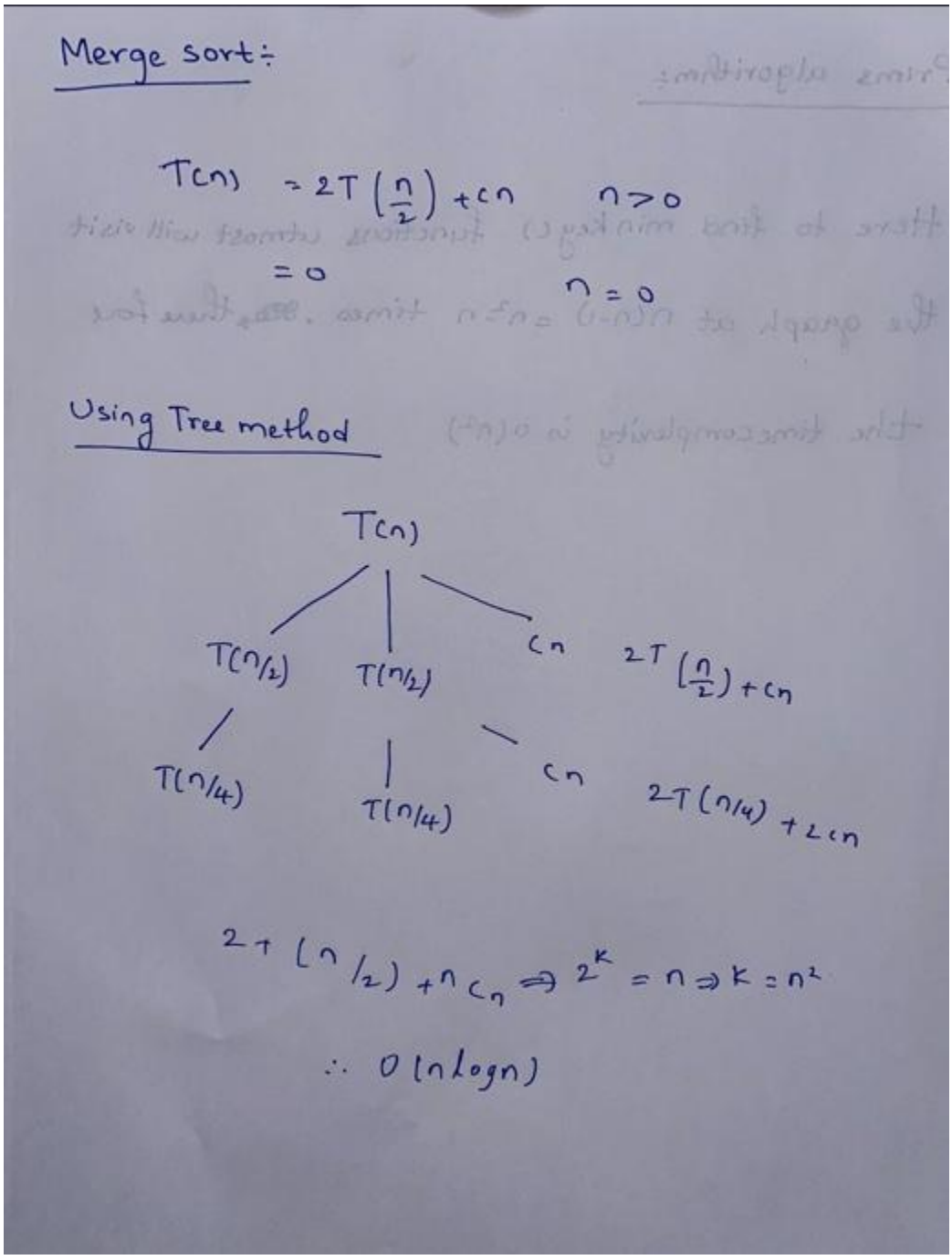
input

```
Given Array
32 5 55 8 62 14

Sorted array
5 8 14 32 55 62


...Program finished with exit code 0
Press ENTER to exit console.
```

ANALYSIS:


Merge sort:-
T(n) = 2T(n/2) + cn n>0
= 0 n = 0
Using Tree method
T(n)
T(n/2)
T(n/2)
cn
2T(n/2) + cn
T(n/4)
T(n/4)
cn
2T(n/4) + 2cn
2T(n/2) + n cn ⇒ 2^k = n ⇒ k = n^2
∴ O(nlogn)