EXPERIMENT-3

Program:
WAP to implement Quick sort using c/c++ and write the time complexity.

Psuedo Code:

```
quickSort(array, leftmostIndex, rightmostIndex)
  if (leftmostIndex < rightmostIndex)
    pivotIndex <- partition(array,leftmostIndex, rightmostIndex)
    quickSort(array, leftmostIndex, pivotIndex - 1)
    quickSort(array, pivotIndex, rightmostIndex)
partition(array, leftmostIndex, rightmostIndex)
  set rightmostIndex as pivotIndex
  storeIndex <- leftmostIndex - 1
  for i <- leftmostIndex + 1 to rightmostIndex
  if element[i] < pivotElement
    swap element[i] and element[storeIndex]
    storeIndex++
  swap pivotElement and element[storeIndex+1]
return storeIndex + 1
```

Time Complexity:

Best Case: omega (N * log N)

Average Case: Theta (N * log N)

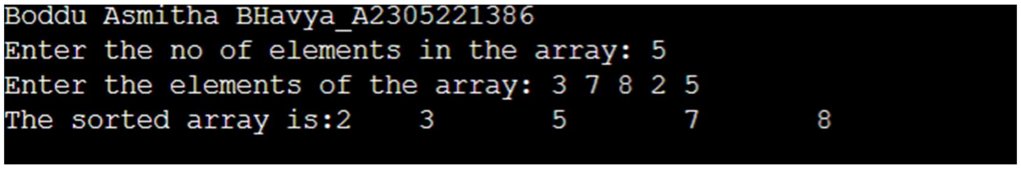
Worst Case: O(N2)

Input:

```
#include<stdio.h>
#define n 100
void exchange(int*a, int*b)
{    int c;
    c=*a;
    *a=*b;
    *b=c;
}
```

```
int partitian(int a[],int p, int r)
{
    int x=a[r];
    int i=p-1;
    for(int j=p;j<=r-1;j++)
    {
        if (a[j]<=x)
        {
           i++;
           exchange(&a[i],&a[j]);
        }
    }
    exchange(&a[i+1],&a[r]);
    return (i+1);
}
void sort(int a[],int p,int r)
{
    if(p<r)
    {
        int part=partitian(a,p,r);
        sort(a,p,part-1);
        sort(a,part+1,r);
    }
}
int main()
{
    printf("Boddu Asmitha BHavya_A2305221386");
    int a[n] ,i, N;
    printf("\nEnter the no of elements in the array: ");
    scanf("%d", &N);
    printf("Enter the elements of the array: ");
    for(i=0; i<N; i++){
    scanf("%d",&a[i]);
    }
    sort(a,0,N-1);
    printf("The sorted array is:");
    for(int i=0;i<N;i++)
    {
        printf("%d\t ",a[i]);
    }
    return 0;
}
```

Output:



```
Boddu Asmitha BHavya_A2305221386
Enter the no of elements in the array: 5
Enter the elements of the array: 3 7 8 2 5
The sorted array is:2    3       5       7       8
```

EXPERIMENT-4

Program:

WAP to implement the Merge Sort using c/c++ and write the complexity.

Pseudo Code:

```
Declare left variable to 0 and right variable to n-1
Find mid by medium formula. mid = (left+right)/2
Call merge sort on (left,mid)
Call merge sort on (mid+1,rear)
Continue till left is less than right
Then call merge function to perform merge sort.

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

Start
Declare an array and left, right, mid variable
Perform merge function.
    mergesort(array,left,right)
    mergesort (array, left, right)
    if left > right
    return
    mid= (left+right)/2
    mergesort(array, left, mid)
    mergesort(array, mid+1, right)
    merge(array, left, mid, right)
```

Stop

Time Complexity:

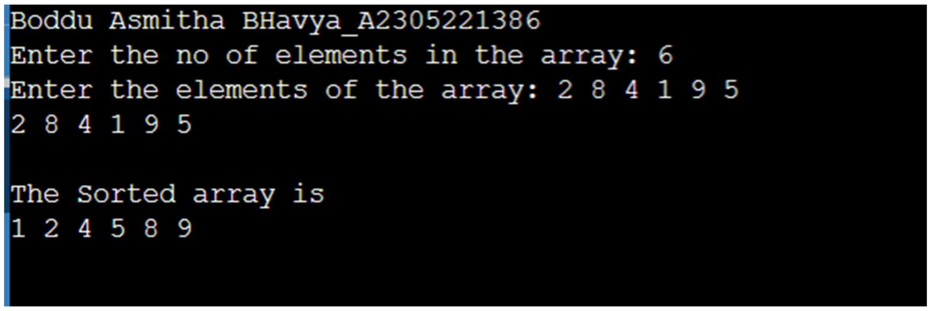
T(n) = 2T(n/2) + θ(n)

After solving
T(n)=O(nlog(n))

Input:

```
#include <stdio.h>
#include <stdlib.h>
#define n 100
void merge(int arr[], int l, int m, int r)
{       int i, j, k;
        int n1 = m - l + 1;
        int n2 = r - m;
        int L[n1], R[n2];
        for (i = 0; i < n1; i++)
                L[i] = arr[l + i];
        for (j = 0; j < n2; j++)
                R[j] = arr[m + 1 + j];
        i = 0;
        j = 0;
        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 mergeSort(int arr[], int l, int r)
{
        if (l < r) {
          int m = l + (r - l) / 2;
                mergeSort(arr, l, m);
                mergeSort(arr, m + 1, r);
                merge(arr, l, m, r);
        }
}
void printArray(int A[], int size)
{
        int i;
        for (i = 0; i < size; i++)
        printf("%d ", A[i]);
        printf("\n");
}
int main()
{
   printf("Boddu Asmitha BHavya_A2305221386");
   int a[n] ,i, N;
   printf("\nEnter the no of elements in the array: ");
   scanf("%d", &N);
   printf("Enter the elements of the array: ");
   for(i=0; i<N; i++){
    scanf("%d",&a[i]);
    }
    printArray(a, N);
            mergeSort(a, 0, N - 1);
            printf("\nThe Sorted array is \n");
            printArray(a, N);
            return 0;
}
```

Output:



```
Boddu Asmitha BHavya_A2305221386
Enter the no of elements in the array: 6
Enter the elements of the array: 2 8 4 1 9 5
2 8 4 1 9 5

The Sorted array is
1 2 4 5 8 9
```

INDEX

S.no	Experiment	Date of Allotment	Date of Evaluation	Sign
