S.No	Date	IDX	PROGRAM
1	02-07-2015	1.1	Implement Merge Sort
		1.2	Implement Quick sort

Week 1 - Program 1.1

AIM : Implement Merge sort and observe the execution time for various input sizes (Average, Worst and Best cases).

PROGRAM :

```
#include<stdio.h>
void merge_sort(int, int);
void merge(int, int, int);
int a[10];

void main()
{
        int n,i;
        printf("\n Enter size of the array");
        scanf("%d",&n);
        printf("\n Enter elements of the array");
        for(i=1;i<=n;i++)
                scanf("%d",&a[i]);
        merge_sort(1,n);
        printf("\n After sorting the elements of the array are");
        for(i=1;i<=n;i++)
                printf("%d \t",a[i]);
}

void merge_sort(int low,int high)
 {
  int mid;
          if(low<high)
          {
                mid=(low+high)/2;
                merge_sort(low, mid);
                merge_sort(mid+1, high);
                merge(low, mid, high);
          }
 }
```

```
void merge(int low, int mid, int high)
 {
  int i, j, ,b[20],r;
        i=low;
        j=mid+1;
        k=low;
        while((i<=mid)&&(j<=high))
        {
                if(a[i]<a[j])
           {
                        b[k]=a[i];
                        i++;
                }
                else
                 {
                        b[k]=a[j];
                        j++;
                }
                 k++;
        }

        if(i<=mid)
        {
                for(r=i; r<=mid; r++)
                {
                        b[k]=a[r];
                        k++;
                 }
        }
        else
        {
                for(r=j; r<=high; r++)
                {
                        b[k]=a[r];
                        k++;
                }
        }

  for(k=low; k<=high; k++)
     a[k]=b[k];
}
```

OUTPUT:

```
Enter size of the array : 5

Enter elements of the array :12 45 32 87 65

After sorting the elements of the array are :   12      32      45      65      87
```

Week 1- Program 1.2

AIM : Implement Quick sort and observe the execution time for various input sizes (Average, Worst and Best cases).

PROGRAM:

```
#include<stdio.h>

void quick_sort(int [],int,int);
int partition(int [], int,int);

void main()
{
        int i,n,a[20];
        printf("\nEnter size of the array : ");
        scanf("%d",&n);
        printf("\nEnter elements of the array : ");
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);
        quick_sort(a,0,n-1);
        printf("\nAfter sorting elements are : ");
        for(i=0;i<n;i++)
                printf("\t%d ",a[i]);

}

void quick_sort(int a[20], int low,int high)
{
        int p;
        if(low<high)
        {
                p=partition(a,low,high);
                quick_sort(a,low,p-1);
                quick_sort(a,p+1,high);
        }
}
```

```
int partition(int a[20], int low,int high)
{
	int i,j,pivot,temp;
	pivot=a[low];
	i=low;
	j=high;
	while(i<j)
	{
		while(a[i]<=pivot)
		{
			i++;
		}
		while(a[j]>pivot)
		{
			j--;
		}
		if(i<j)
		{
			temp=a[i];
			a[i]=a[j];
			a[j]=temp;
		}
	}
	temp=a[low];
	a[low]=a[j];
	a[j]=temp;

	return j;
}
```

OUTPUT:

```
Enter size of the array : 5

Enter elements of the array : 10 50 20 40 30

After sorting elements are :    10      20      30      40      50
```