Data Structures

Experiment no. 9

Develop code to implement Merge & Quick Sort techniques

Q. WAP in C to implement Merge Sort.

Code:

```
#include<stdio.h>
#include<conio.h>
#define MAX 100

void mergeSort(int a[], int lb,int ub);
void merge(int a[], int lb,int mid,int ub);
void main()
{
        int n,i,a[MAX],lb=0,ub;
        clrscr();
        printf("Enter Number of Element to be Sort: ");
        scanf("%d",&n);
        printf("\n Enter %d Elements in Array to Sort: ",n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        printf("\n Elements Before Sorting:\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }
        ub=n-1;
        mergeSort(a,lb,ub);
        printf("\n Elements after Sorting:\n");
        for(i=0;i<n;i++)
        {
                printf("%d\t",a[i]);
        }

        getch();
}


void mergeSort(int a[], int lb,int ub)
```

```
{
        int mid;
        if(lb<ub)
        {
                mid=(lb+ub)/2;
                mergeSort(a,lb,mid);
                mergeSort(a,mid+1,ub);
                merge(a,lb,mid,ub);
        }
}

void merge(int a[],int lb,int mid,int ub)
{
        int i=lb,j=mid+1,k=lb,b[MAX];
        while(i<=mid && j<=ub)
        {
                if(a[i]<=a[j])
                {
                        b[k]=a[i];
                        i++;
                }
                else
                {
                        b[k]=a[j];
                        j++;
                }
                k++;
        }

        if(i>mid)
        {
                while(j<=ub)
                {
                        b[k]=a[j];
                        k++;j++;
                }
        }
        else
        {
                while(i<=mid)
                {
                        b[k]=a[i];
                        k++;i++;
                }
        }

        //copying sorted array elements from b[] into a[]
        for(i=lb;i<=ub;i++)
```

```
                a[i]=b[i];

}
```

Output:

```
Enter Number of Element to be Sort: 10

 Enter 10 Elements in Array to Sort: 1
2
90
120
40
32
86
51
23
47

 Elements Before Sorting:
1       2       90      120     40      32      86      51
23      47
 Elements after Sorting:
1       2       23      32      40      47      51      86
90      120
```

Q. WAP in C to implement Quick Sort.

Code:

```
#include<stdio.h>
#include<conio.h>
#define MAX 100
int partition(int a[],int lb,int ub);
void quickSort(int a[],int lb,int ub);

void main()
{
	int n,i,a[MAX],lb,ub;
	clrscr();
	printf("Enter Number of Element to be Sort: ");
	scanf("%d",&n);
	printf("\n Enter %d Elements in Array to Sort: ",n);
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
	printf("\n Array Before Sorting:\n");
	for(i=0;i<n;i++)
		printf("%d\t",a[i]);

	lb=0;
	ub=n-1;
	quickSort(a,lb,ub);
	printf("\n Array After Sorting:\n");
	for(i=0;i<n;i++)
		printf("%d\t",a[i]);

	getch();
}

void quickSort(int a[],int lb,int ub)
{	int loc;
	if(lb<ub)
	{
		loc=partition(a,lb,ub);
		quickSort(a,lb,loc-1);
		quickSort(a,loc+1,ub);
	}


}

int partition(int a[],int lb, int ub)
```

```
{
        int pivot,start,end,temp;
        pivot=a[lb];
        start=lb;
        end=ub;
        while(start<end)
        {
                while(a[start]<=pivot)
                        start++;
                while(a[end]>pivot)
                        end--;
                if(start<end)
                {
                        temp=a[start];
                        a[start]=a[end];
                        a[end]=temp;
                }
        }
        temp=a[lb];
        a[lb]=a[end];
        a[end]=temp;
        return end;
}
```

Output:

```
Enter Number of Element to be Sort: 10

 Enter 10 Elements in Array to Sort: 51
23
60
84
91
12
15
999
75
8

 Array Before Sorting:
51      23      60      84      91      12      15      999
75      8
 Array After Sorting:
8       12      15      23      51      60      75      84
91      999
```