**Experiment-4**

**IMPLEMENTATION OF QUICK SORT, MERGE SORT.**

There are multiple ways by which data can be sorted in the [R language](https://www.geeksforgeeks.org/introduction-to-r-programming-language/). There are 5 sorting techniques used they are:

* Bubble Sort
* Insertion Sort
* Selection Sort
* Merge Sort
* Quick Sort

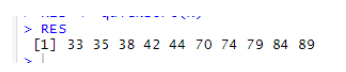
**QUICK SORT**

Quicksort algorithm works in like divide and rule. The random element is selected as a pivot in an array and then all other elements except pivot are divided into two partitions. In the next step, all the elements which are less than and greater than the pivot are divided into two different partitions. Finally, the elements are sorted using recursion.

# Quick sort algorithm:  
quickSort <- function(arr) {  
# Pick a number at random.  
mid <- sample(arr, 1)  
# Place-holders for left and right values.  
left <- c()  
right <- c()

# Move all the smaller values to the left, bigger values to the right.  
lapply(arr[arr != mid], function(d) {  
if (d < mid) {  
left <<- c(left, d)  
}  
else {  
right <<- c(right, d)  
}  
})  
if (length(left) > 1) {  
left <- quickSort(left)  
}  
if (length(right) > 1) {  
right <- quickSort(right)  
}  
# Finally, return the sorted values.  
c(left, mid, right)  
}  
x <-sample(1:100,10)  
RES <- quickSort(x)  
RES

**Output**



**Merge Sort**

Merge sort is very similar to quicksort however, here the array is divided into two equal halves. Merge sort algorithm has been divided into two parts a merge and a sort function. In merge sort, a list is broken down into multiple sub-lists till every sub-list consists of an individual element. Merging those sub-lists results is a sorted list.

mmerge<-function(a,b) {  
r<-numeric(length(a)+length(b))  
ai<-1; bi<-1; j<-1;  
for(j in 1:length(r)) {  
if((ai<=length(a) && a[ai]<b[bi]) || bi>length(b)) {  
r[j] <- a[ai]

ai <- ai+1  
} else {  
r[j] <- b[bi]

bi <- bi+1  
 }  
 }  
 r  
 }  
 mmergesort<-function(A) {  
 if(length(A)>1) {  
q <- ceiling(length(A)/2)  
a <- mmergesort(A[1:q])  
b <- mmergesort(A[(q+1):length(A)])  
mmerge(a,b)  
} else {  
A  
}

}  
 x <-sample(1:100,10)  
RES <- mmergesort(x)  
RES

**Output**

