PART A

(PART A : TO BE REFFERED BY STUDENTS)

Experiment No.03

A.1 Aim:

Implementation of quick sort.

A.2 Prerequisite:

C programming

A.3 Outcome:

After successful completion of this experiment students will be,

Apply divide and conquer strategy to solve problems.

A.4 Theory:

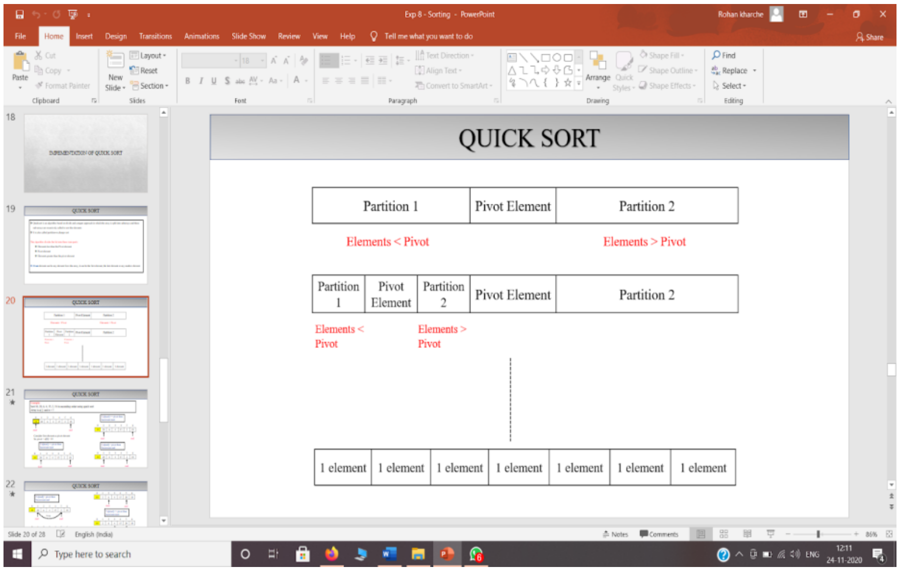
Quick sort:

Quicksort is an algorithm based on divide and conquer approach in which the array is split into subarrays and these sub-arrays are recursively called to sort the elements. It is also called partition-exchange sort.

This algorithm divides the list into three main parts:

- Elements less than the Pivot element
- Pivot element
- Elements greater than the pivot element

Pivot element can be any element from the array, it can be the first element, the last element or any random element.



QUICK SORT

Partition 1	Pivot Element	Partition 2

Elements < Pivot

Elements > Pivot

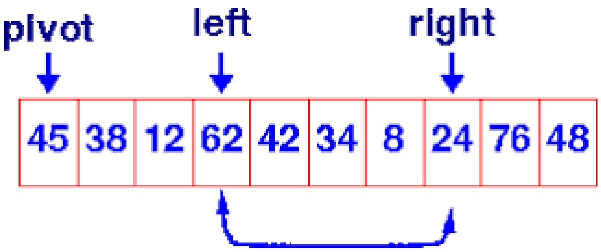
Partition 1	Pivot Element	Partition 2	Pivot Element	Partition 2

Elements < Pivot

Elements > Pivot

1 element	1 element	1 element	1 element	1 element	1 element	1 element

Example:



Algorithm:

```
Step 1: Repeat the steps 2 to 10 while low < high
Step 2:     Select Pivot = a[low]
                Pivot location p = low
Step 3:     i = low and j = high
Step 4:     Repeat the steps 5 to 7 while i < j
Step 4:         Increment index i till a[i] < pivot
Step 5:         Decrement index j till a[j] > pivot
Step 6:         Swap a[i] with a[j]
            [End of Loop]
Step 7:     if i > j then
                Swap a[p] with a[j]
Step 8:     call quicksort(a, low, j-1)
Step 9:     Call quicksort(a, j+1, high)
        [End of Loop]
Step 10: Stop
```

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

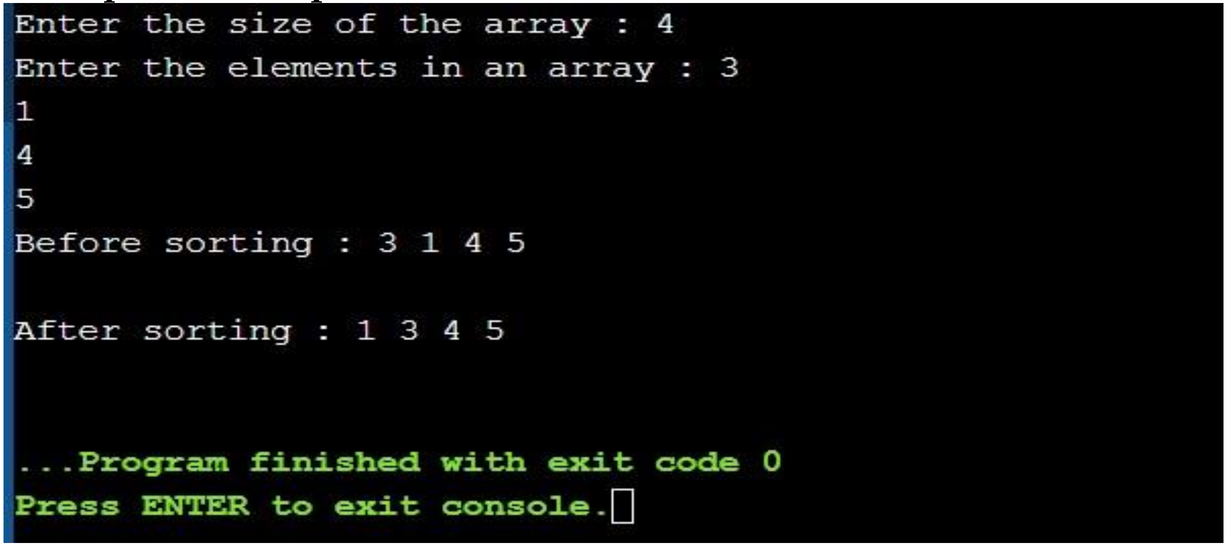
Roll. No.71	Name: Omkar Gujja
Class: 2	Batch: MTRX
Date of Experiment: 27/2/2021	Date of Submission: 6/3/2021
Grade:	

B.1 Software Code written by student:

```
#include<malloc.h>
#include <stdio.h>

void displayArray(int arr[], int len){
   for (int i = 0; i < len; i++){
      printf("%d ", arr[i]);
   }
   printf("\n");
}

void swap(int *a, int *b){
   int t = *a;
   *a = *b;
   *b = t;
}

int partition(int arr[], int l, int h){
   int pivot = arr[l];
   int start = l;
   int end = h;

   while (start<end){

      do{
         start++;
      }while(pivot>arr[start]);

      do{
         end--;
      }while(pivot<arr[end]);
      if (start<end){
      swap(&arr[start], &arr[end]);
      }
   }
```

```
    swap(&arr[end], &arr[l]);
    return end;
}

void quicksort(int arr[], int l, int h){
    if (l<h){
    int p = partition(arr, l, h);
    quicksort(arr, l, p);
    quicksort(arr, p+1, h);
    }
}

int *takeInput(){
    int size;
    printf("Enter the size of the array : ");
    scanf("%d",&size);
    int *p= malloc(sizeof(size));

    printf("Enter the elements in an array : ");
    for(int i=0;i<size;i++)
    {
        scanf("%d", &p[i]);
    }
    return p;
}

int main()
{
    int *arr = takeInput();

    int n=sizeof(*arr);


    printf("Before sorting : ");
    displayArray(arr, n);
    quicksort(arr, 0, n);
    printf("\nAfter sorting : ");
    displayArray(arr, n);
    return 0;
}
```

B.2 Input and Output:



```
Enter the size of the array : 4
Enter the elements in an array : 3
1
4
5
Before sorting : 3 1 4 5

After sorting : 1 3 4 5


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

B.3 Observations and learning:

We observe how implemation of quick sort are done.

B.4 Conclusion:

We have understood the implementation of quick sort.

B.5 Question of Curiosity

Q.1 what is the time complexity of quick sort?

Soln.

To sort an array of n distinct elements, quicksort takes O(n log n) time in expectation, averaged over all n! permutations of n elements with equal probability.

Best-case performance
Worst-case space complexity
Worst-case performance
Average performance

Q.2 Explain internal and external sorting.

Soln.

Internal sorting: If the input data is such that it can be adjusted in the main memory at once, it is called internal sorting. External sorting: If the input data is such that it cannot be adjusted in the memory entirely at once, it needs to be stored in a hard disk, floppy disk, or any other storage device. This is called external sorting.

Q.3 Quick Sort follows Divide and Conquer Strategy

a) true b) false

Soln.

a) True

Explanation: In quick sort, the array is divided into sub-arrays and then it is sorted (divide-and- conquer strategy).