Experiment 5

WAP to Perform the Comparative analysis of Quick sort and Randomized Quick.

<u>Quick sort</u>

Program:-

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
int partition(int a[], int first, int last)
{
 int i, j, pivot, temp;
 if (first < last)
 {
  pivot = first;
  i = first;
  j = last;
  while (i < j)
  {
   while (a[i] <= a[pivot] && i < last)
    i++;
   while (a[j] > a[pivot])
    j--;
   if (i < j)
   {
    temp = a[i];
    a[i] = a[j];
    a[j] = temp;
   }
  }
  temp = a[pivot];
  a[pivot] = a[j];
```

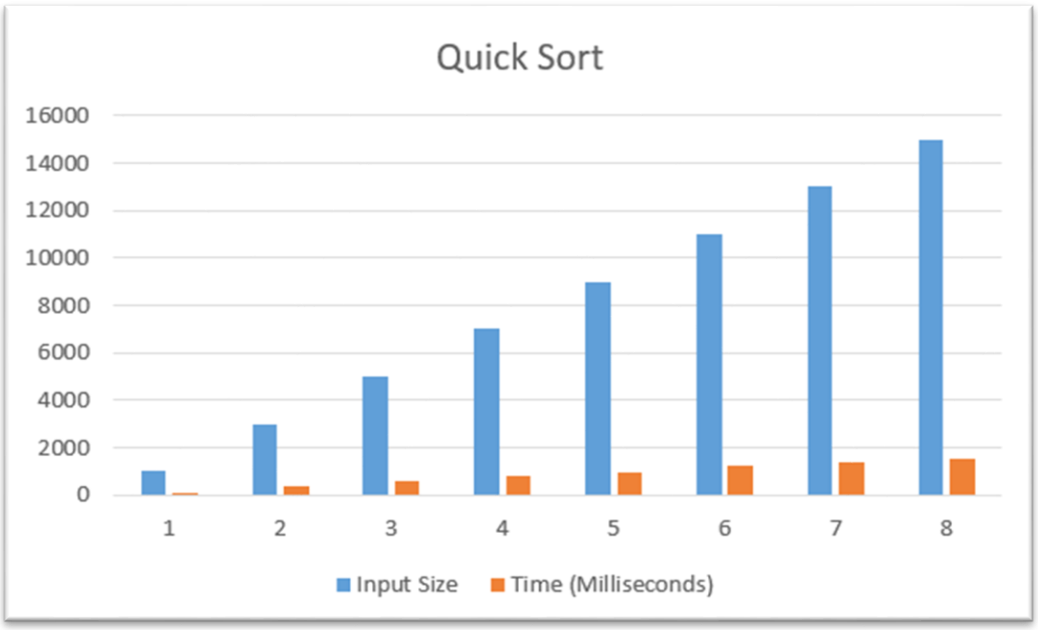
```
    a[j] = temp;
    return j;
  }
}

void quicksort(int a[], int first, int last)
{
  int p;
  if (first < last)
  {
    p = partition(a, first, last);
    quicksort(a, first, p - 1);
    quicksort(a, p + 1, last);
  }
}
int main()
{
  int a[50000], n, i, randNum;
  double time;
  clock_t start, end;
  printf("Enter the total input size:");
  scanf("%d", &n);
  start = clock();
  for (i = 0; i < n; i++)
  {
    randNum = rand() % 10000;
    a[i] = randNum;
    printf("%d\t", a[i]);
  }
  quicksort(a, 0, n - 1);
  printf("\nSorted list:");
  for (i = 0; i < n; i++)
  {
    printf("%d \t", a[i]);
  }
  end = clock();
  time = ((double)(end - start) * 1000) / CLOCKS_PER_SEC;
  printf("\nTime=%lf millseconds", time);
  return 0;
}
```

Result Analysis and Discussion:

This experiment has been conducted in a 64-bit system with 16 GB RAM and Processor 12th Gen Intel(R) Core (TM) i5-12500H 3.10 GHz. The algorithm is implemented in C programming language in Visual Studio Code 1.83.1 Code Editor. In this experiment the algorithm to sort an array of size "n" using Quick Sort Technique has been implemented and executed for different value of n. During this experiment for different value of n, the time taken by the algorithm has been measured and tabulated as shown in table below.

Input Size	**Time (Milliseconds)**
1000	120
3000	405
5000	585
7000	816
9000	981
11000	1235
13000	1387
15000	1546

The graph shown below is the plot of input n and the time in milliseconds taken by the algorithm while running on a system recorded in table above.



Based on the above table and graph it is clearly seen that the size of array n has linear relationship with the time taken by the system to sort an array.

<u>Randomized Quick sort</u>

Program:-

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
int partition(int a[], int first, int last);
int randpartition(int a[], int first, int last)
{
  int k, temp;
  k = (rand()%(last - first)) + first;
  //swapping
  temp=a[first];
  a[first]=a[k];
  a[k]=temp;
  return partition(a, first, last);
}
int partition(int a[], int first, int last)
{
  int i, j, pivot, temp;
  if(first<last) {
    pivot=first;
    i=first;
    j=last;
    while(i<j)
    {
      while(a[i]<=a[pivot]&&i<last)
        i++;
      while(a[j]>a[pivot])
        j--;
      if(i<j)
      {
        temp=a[i];
        a[i]=a[j];
        a[j]=temp;
      }
    }
    temp=a[pivot];
    a[pivot]=a[j];
    a[j]=temp;
    return j;
  }
  return 0;
}
void randquicksort(int a[], int first, int last) {
  int p;
  if (first<last) {
    p=randpartition(a, first, last);
    randquicksort(a, first,p-1);
    randquicksort(a,p+1,last);
  }
}
int main() {
  int a[50000],n,i,randNum;
  double time;
  clock_t start, end;
  printf("Enter the total input size:");
  scanf("%d",&n);
  start = clock();
  for (i=0;i<n;i++)
```

```
 {
   randNum=(rand()%10000);
   a[i]=randNum;
   printf("%d\t", a[i]);
 }
 //quick sort
 randquicksort(a,0,n-1);
 //printing sorted List
 printf("\n Sorted list:\n");
 for(i=0;i<n;i++)
   printf("%d \t", a[i]);
 //end clock
 end = clock();
 time = ((double) (end-start)*1000)/CLOCKS_PER_SEC;
 printf("\n Time Taken = %lf milliseconds", time);
 return 0;
}
```

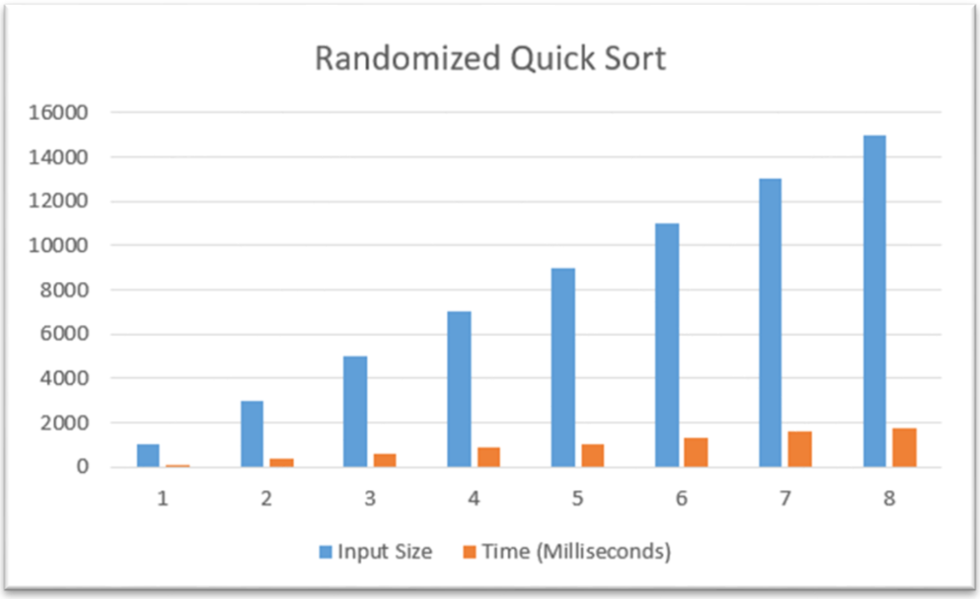
Result Analysis and Discussion:

This experiment has been conducted in a 64-bit system with 16 GB RAM and Processor 12th Gen Intel(R) Core (TM) i5-12500H 3.10 GHz. The algorithm is implemented in C programming language in Visual Studio Code 1.83.1 Code Editor. In this experiment the algorithm to sort an array of size "n" using Randomized Quick Sort Technique has been implemented and executed for different value of n. During this experiment for different value of n, the time taken by the algorithm has been measured and tabulated as shown in table below.

Input Size	**Time (Milliseconds)**
1000	119
3000	395
5000	575
7000	867

9000	1055
11000	1296
13000	1611
15000	171

The graph shown below is the plot of input n and the time in milliseconds taken by the algorithm while running on a system recorded in table above.



Based on the above table and graph it is clearly seen that the size of array n has linear relationship with the time taken by the system to sort an array.

Conclusion:

In this experiment, it has been found that the size of input “n” has linear relationship with the time taken by the system to sort the array .So, Quick sort is better than randomized quick sort algorithm to sort an array.