Experiment 5:

Write a program for analysis of quick sort by using deterministic and randomized variant

Code:

```
// C++ implementation QuickSort
// using Lomuto's partition Scheme.
#include <cstdlib>
#include <time.h>
#include <iostream>
using namespace std;

// This function takes last element
// as pivot, places
// the pivot element at its correct
// position in sorted array, and
// places all smaller (smaller than pivot)
// to left of pivot and all greater
// elements to right of pivot
int partition(int arr[], int low, int high)
{
	// pivot
	int pivot = arr[high];

	// Index of smaller element
	int i = (low - 1);

	for (int j = low; j <= high - 1; j++)
	{
		// If current element is smaller
		// than or equal to pivot
		if (arr[j] <= pivot)
		{

			// increment index of
			// smaller element
			i++;
			swap(arr[i], arr[j]);
		}
	}
	swap(arr[i + 1], arr[high]);
	return (i + 1);
}

// Generates Random Pivot, swaps pivot with
```

```
// end element and calls the partition function
int partition_r(int arr[], int low, int high)
{
	// Generate a random number in between
	// low .. high
	srand(time(NULL));
	int random = low + rand() % (high - low);

	// Swap A[random] with A[high]
	swap(arr[random], arr[high]);

	return partition(arr, low, high);
}

/* The main function that implements QuickSort
arr[] --> Array to be sorted,
low --> Starting index,
high --> Ending index */
void quickSort(int arr[], int low, int high)
{
	if (low < high)
	{

		/* pi is partitioning index,
		arr[p] is now
		at right place */
		int pi = partition_r(arr, low, high);

		// Separately sort elements before
		// partition and after partition
		quickSort(arr, low, pi - 1);
		quickSort(arr, pi + 1, high);
	}
}

/* Function to print an array */
void printArray(int arr[], int size)
{
	int i;
	for (i = 0; i < size; i++)
		cout<<arr[i]<<" ";
}

// Driver Code
int main()
{
```

```
        int arr[] = { 10, 7, 8, 9, 1, 5 };
        int n = sizeof(arr) / sizeof(arr[0]);

        quickSort(arr, 0, n - 1);
        printf("Sorted array: \n");
        printArray(arr, n);

        return 0;
}
```

Output:

Sorted array:

1 5 7 8 9 10