1. Sort a given set of elements using the quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the 1st to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

```
import java.util.Random;
import  java.util.Scanner;
public class Quicksort
{
        static final int MAX = 10005;
        static int[] a = new int[MAX];

        public static void main(String[] args)
        {
                Scanner input = new Scanner(System.in);
                System.out.print("Enter Max array size: ");
                int n = input.nextInt();
                Random random = new Random();
                System.out.println("Enter the array elements: ");
                for (int i = 0; i < n; i++)
                        a[i] = random.nextInt(1000);
                for (int i = 0; i < n; i++)
                        System.out.print(a[i] + " ");
                long startTime = System.nanoTime();
                QuickSortAlgorithm(0, n - 1);
                long stopTime = System.nanoTime();
                long elapsedTime = stopTime - startTime;
                System.out.println("Time Complexity (ms) for n = " + n + " is : " +
                (double)elapsedTime / 1000000);
                System.out.println("Sorted Array (Quick Sort):");
                for (int i = 0; i < n; i++)
                        System.out.print(a[i] + " ");
                input.close();
        }

        public static void QuickSortAlgorithm(int p, int r)
        {
                        int i, j, temp, pivot;
                        if (p < r)
                        {
                                i = p;
                                j = r;
                                pivot = a[p];
                                while(true)
                                {
                                        i++;
                                        while (a[i] < pivot && i<r)
                                        {
```

```
                    i++;
                }
                while (a[j] > pivot)
                {
                    j--;
                }
                if (i < j)
                {
                    temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
                }
                else
                {
                    break;
                }
            }
            a[p] = a[j];
            a[j] = pivot;
            QuickSortAlgorithm(p, j - 1);
            QuickSortAlgorithm(j + 1, r);
        }
    }
}
```