**Assignment-66**
**Divide and Conquer**

1. We are given a sorted array A[] of n elements. We need to find if x is present in A or not. In binary search we always used the middle element, here we will randomly pick one element in a given range.
2. Given an array of integers. Find the Inversion Count in the array. Inversion Count: For an array, inversion count indicates how far (or close) the array is from being sorted. If array is already sorted then the inversion count is 0. If an array is sorted in the reverse order then the inversion count is the maximum. Formally, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j.

   **Example 1:**
   Input: N = 5, arr[] = {2, 4, 1, 3, 5}
   Output: 3
   Explanation: The sequence 2, 4, 1, 3, 5
   has three inversions (2, 1), (4, 1), (4, 3).
   **Example 2**:
   Input: N = 5
   arr[] = {2, 3, 4, 5, 6}
   Output: 0
   Explanation: As the sequence is already
   sorted so there is no inversion count.
   **Example 3**:
   Input: N = 3, arr[] = {10, 10, 10}
   Output: 0
   Explanation: As all the elements of array
   are the same, so there is no inversion count.

3. Given a sorted array arr[] of size N. Find the element that appears only once in the array. All other elements appear exactly twice.

   Example 1:
   Input:
   N = 11
   arr[] = {1, 1, 2, 2, 3, 3, 4, 50, 50, 65, 65}
   Output: 4
   Explanation: 4 is the only element that
   appears exactly once.

4. There are Infinite People Standing in a row, indexed from 1. A person having index 'i' has strength of (i*i). You have Strength 'P'. You need to tell what is the maximum number of People You can Kill With your Strength P. You can only Kill a person with strength 'X' if P >= 'X' and after killing him, Your Strength decreases by 'X'.
5. Given a sorted and rotated array A of N distinct elements which are rotated at some point, and given an element K. The task is to find the index of the given element K in array A.
6. Given a positive integer N, find the last digit of the Nth term from the Fibonacci series.
   Note: For N=0 you have to return 0.
   Expected Time Complexity: O(N)