**Assignment No**. 4

**Assignment Title**: Divide and Conquer (DC) technique

**Problem 1**:

Binary Search is an Example of DC technique, where we divide the input data (*given in an indexed and sorted array*) using its known size, and reduce the size of the problem.

a. Let us say you have an array that is sorted but rotated at some point, i.e., if the sorted array elements are A = [$a_1$, $a_2$, … $a_k$, $a_{k+1}$, … $a_n$] the given array is A' = [$a_{k+1}$, … $a_n$ $a_1$, $a_2$, … $a_k$}; The goal is to modify the binary search algorithm in such a way so that is can search for a key in A' in O(log n) time.
b. Now we are going to solve the same problem (find a given key) for an unknown input size. Consider a monotonically increasing function f(n), i.e., for all two given integers $n_1$ and $n_2$, if $n_1<n_2$ then f(n1) < f(n2). Assume f(n) = $x^3 – 10x^2$ - 15x – 20. We want to know for which value of **n** the function f(n) becomes positive for the first time.
**c.** **Bonus**: Consider a function f(n) which is a monotonically increasing at first, and after a certain point it is a monotonically decreasing, i.e., there exist an integer $n_0$, for which f($n_0$-1) < f($n_0$) and f($n_0$) > f($n_0$+1). Consider the Gaussian function/ bell curve as f(n); We want to find the **$n_0$** for the given f(n).

**Tasks**:

Write codes for solving both the above problems and analyze the complexity for the same.

**Problem 2**:
You remember Tower of Hanoi problem; this is another example of DC technique. You have n disks and 3 pegs; all disks are arranged from largest to smallest; you are to move the n disks from peg 1 to peg 2 using the peg 3 as intermediate, by moving one disk at a time and never putting a larger disk over a smaller one. Here also, you reduce the size of the problem from n disks to n-1 disks, and so on until you can solve the problem.

Now you are going to consider the problem again but instead of 3 pegs, you have 4 pegs. You are now moving from peg 1 to peg 2 using peg 3 and peg 4 as intermediate. We refer to the old problem as *TOH3* and this new one as *TOH4*.

a. One method will be
    - Move k (k<n) disks from peg 1 to peg 3, using peg 2 and peg 4 (*TOH4*)
    - Move another (n-k-1) disks from peg 1 to peg 4 using peg 2 (*TOH3*)
    - Move nth disk from peg 1 to peg 2
    - Move n-k-1 disks from peg 4 to peg 2 using peg 1 (*TOH3*)
    - Move n-k-1 disks from peg 4 to peg 2 using peg 1 and peg 4 (*TOH4*)
b. What value of k will be the best for the above given method? You can analyze the complexity of the method to find out?

**Tasks**:

1. Write codes for the above given method for solve TOH4. (you also need TOH3 function for this to work).
2. Analyze the complexity of the method to figure out what is the best value of k for the given solution.
3. **Bonus**: Can you devise better solution for solving the problem? Analyze the complexity of your solution to show how much better or worse your version is compared to the given one.