Assignment: Recursion, Recurrence Relations and Divide & Conquer

1. **Solve recurrence relation using three methods**:

   Write recurrence relation of below pseudocode that calculates x^n, and solve the recurrence relation using three methods that we have seen in the explorations.

```
power2(x,n):
    if n==0:
        return 1
    if n==1:
        return x
    if (n%2)==0:
        return power2(x, n//2) * power2(x,n//2)
    else:
        return power2(x, n//2) * power2(x,n//2) * x
```

2. **Solve recurrence relation using any one method**:

   Find the time complexity of the recurrence relations given below using any one of the three methods discussed in the module. Assume base case T(0)=1 or/and T(1) = 1.

   a) $T(n) = 4T\ (\ n/2\ ) + n$
   b) $T(n) = 2T\ (\ n/4\ ) + n^2$

3. **Implement an algorithm using divide and conquer technique**: Given two sorted arrays of size m and n respectively, find the element that would be at the k^{th} position in combined sorted array.

   a. Write a pseudocode/describe your strategy for a function kthelement(Arr1, Arr2, k) that uses the concepts mentioned in the divide and conquer technique. The function would take two sorted arrays Arr1, Arr2 and position k as input and returns the element at the k^{th} position in the combined sorted array.
   b. Implement the function kthElement(Arr1, Arr2, k) that was written in part a. Name your file **KthElement.py**

   Examples:
   Arr1 = [1,2,3,5,6] ; Arr2= [3,4,5,6,7] ; k= 5
   Returns: 4
   Explanation: 5^{th} element in the combined sorted array [1,2,3,3,4,5,5,6,6,7] is 4