## Question 1

Provide a worst-case asymptotic time complexity of the following algorithms/ codes by using a suitable asymptotic notation considering a nearest function. Assume that there are no errors/ bugs in the algorithms/ codes.

<table>
<tr><th></th><th>Algorithm/ code</th><th>Time Complexity</th></tr>
<tr><td>a)</td><td><pre>Func(n)
BEGIN
      FOR (i = 1 to n)
            FOR (j = 1 to i×i)
                  IF (j modulus i = 1) THEN
                        PRINT i
            END FOR
      END FOR
END</pre></td><td>$O(n^3)$</td></tr>
<tr><td>b)</td><td><pre>Fun(int n) {
     if (n <= 1)
          return 1;
     for (int i = 1; i <= n; i = i * 2) {
          int p = 1;
          for (int j = 1; j <= n; j = j++) {
               p = p + j;
          }
     }
     return Fun(n/10) + Fun(n/10);
}</pre>Hint: First make recurrence and then solve it</td><td>T(n)==2T(n/10) + nlogn<br><br>O(nlogn)</td></tr>
<tr><td>c)</td><td>Problem<br>Given N integers, an optimal algorithm is provided below to count the total pairs of integers that have a difference of K.<br>Inputs<br>Number of integers N, the difference K, and the list A of elements.<br>Output<br>An integer telling the number of pairs that have a difference of K.<br><br>Optimal Algorithm<br><br>Initialize count as 0<br>Sort all numbers in increasing order using a quickest sorting algorithm on random data. O(nlogn)<br>Optimally remove duplicates from the list. Let D be the new list size after the removal of duplicates. O(n)<br>Do the following for each element A[i], where i varies from 1 to D O(nlogn)<br>a) Binary Search for A[i] + K in subarray from i+1 to D.<br>b) If A[i] + K found, increment count.<br>Return count</td><td>O(nlogn)</td></tr>
</table>

## Question 2

**a)** Can the Master theorem (for solving recurrence) be applied to the following recurrence relation: Why or why not? If yes, then write the time complexity for this recurrence using Master Theorem (given on the last page).

$T(n) = 2T(n/2) + n / \lg n$

Master theorem is not applicable. n/lgn grows only logarithmically slower than n, not polynomially slower (page # 105 of the textbook).

**b)** Solve the following recurrence relation using iteration method. Show all steps.

$T(n) = 9T(n/3) + n^2$

$$
\begin{aligned}
T(n) &= 9T(\tfrac{n}{3}) + n^2 \\
&= 9\,(9T(\tfrac{n}{3^2}) + (\tfrac{n}{3})^2) + n^2 \\
&= 9^2\, T(\tfrac{n}{3^2}) + 2n^2 \\
&= 9^2\,(9T(\tfrac{n}{3^3}) + (\tfrac{n}{3^2})^2) + 2n^2 \\
&= 9^3\, T(\tfrac{n}{3^3}) + 3n^2 \\
&\ldots \\
&= 9^k\, T(\tfrac{n}{3^k}) + kn^2
\end{aligned}
$$

If $3^k = n$, then recursion the will reach base case (i.e $T(\frac{n}{3^k})$ = T(1), which is constant), so

$k = \log_3 n$

$$
\begin{aligned}
T(n) &= 9^{\log_3 n} + \log_3 n \times n^2 \\
&= n^2 + n^2 \log_3 n \\
&= \theta\,(n^2 \log_3 n)
\end{aligned}
$$

**c)** Solve the following recurrence using recursion tree method.

$T(n) = T(n/3) + T(2n/3) + \Theta(n)$

Please consult the text book (page # 99)

## Question 3

A unimodal array is an array that has a sequence of monotonically increasing integers followed by a sequence of monotonically decreasing integers. All elements in the array are unique.

An array $A[1.. . . n]$ of size $n \geq 2$ is called unimodal if there exists an integer $m$, $1 \leq m \leq n$, such that $A[i-1] < A[i]$ *for* $i = 1, \ldots, m$ and $A[i-1] > A[i]$ *for* $i = m+1, \ldots, n$. We call $m$ the ***mode*** of $A$. Note that this mode is different from the statistical mode. For example, the array

$A[] = \{1, 2, 4, 7, 11, 10, 8, 4, -9\}$

is unimodal. Its mode is m = 5 since:

i. $1 < 2 < 4 < 7 < 11$

ii. $A[5] = 11$

iii. $11 > 10 > 8 > 4 > -9$

***a.*** Write an iterative algorithm to find mode *m* of a unimodal input array *A[1...n]*. Note that the call of the function is ***naiveModeFinder(A, n).*** **[4 Marks]**

***b.*** Give an efficient recursive divide-and-conquer algorithm to compute the mode *m* of a unimodal input array *A[1...n]* in ***O(lg n)*** time. Note that the initial call of the function is ***recursiveModeFinder(A, 1, n).*** **[16 Marks]**

**Note: Assume that the input is a unimodal array.**

Solution`:

a.

```
Procedure naiveModeFinder(A, n)
BEGIN
  IF (n < 2) THEN
     return 1
  ELSE
    FOR i = 2 to n
       IF A[i] < A[i-1]
          return (i-1)
    END FOR
    return n
END
```

b.

```
Procedure recursiveModeFinder(A, p, q)
BEGIN
  IF (p=q) THEN
     return p
  ELSE
    m=(p+q)/ 2
    IF (A[m−1] < A[m]) THEN
      return recursiveModeFinder (A,m,q)
    ELSE
      return recursiveModeFinder (A,p,m)
END
```