Q1) 1) $T(n) = T(n-3) + 3\log n$ $\quad T(n) = O(n \log n)$ $\quad c > 0$

$T(n) = T(n-3) + 3\log n \leq c(n-3)\log(n-3) + 3\log n$

$= c(n-3)\log n + 3\log n$ $\quad \because \log n$ is monotonically inc for $n > 0$

$\therefore \log n > \log(n-3)$

$\therefore cn\log n - 3\log n + 3\log n$

$\therefore \leq cn\log n$

Hence $T(n) \leq cn\log n$

2) $T(n) = 4T(\frac{n}{3}) + n$ $\quad$ ~~$T(n) \leq O(n^{\log_3 4})$~~ $\quad T(n) = O(n^{\log_3 4})$

$T(n) = 4T(\frac{n}{3}) + n \leq 4c\left(\frac{n}{3}\right)^{\log_3 4} + n$

$= \frac{4}{3^{\log_3 4}}$ ~~$cn\log$~~ $cn^{\log_3 4} + n$

$= \frac{4}{4} cn^{\log_3 4} + n$

$< cn^{\log_3 4} + n \quad \& \quad n^{\log_3 4} > n$

$\therefore T(n) \leq cn^{\log_3 4}$

3) $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{8}) + n$ $\quad T(n) = O(n)$

$T(n) = T(n/2) + T(n/4) + T(n/8) + n \leq c(n/2) + c(n/4) + c(n/8) + n$

$= (c/2 + c/4 + c/8 + 1)n = c'n$

Let $c = 8$ $\quad \therefore c' = (4 + 2 + 1 + 1) = 8$ $\quad \because c > 0$

Hence $T(n) \leq cn$

4) $T(n) = 4T(\frac{n}{2}) + n^2$ $\quad T(n) = O(n^2)$

$T(n) = 4T(n/2) + n^2 \leq 4c\left(\frac{n}{2}\right)^2 + n^2$

$= (c+1)n^2$

let $c' = (c+1)$

$\therefore T(n) \leq c' n^2$

Q.2) 1) $T(n) = 3T(\frac{n}{2}) + n^2$

$a = 3$, $b = 2$, $k = 2$

$a < b^k$ $\therefore$ case (iii)

Since $p \geq 0$ $\therefore T(n) = \Theta(n^k \log^p n)$

$\therefore T(n) = \Theta(n^2)$

2) $T(n) = 2^n T(\frac{n}{2}) + n^2$

Can not be solved by masters as it is not in recurrence form $(a != 2^n)$

3) $T(n) = 3T(\frac{n}{4}) + n\log n$

$a = 3$, $b = 4$, $k = 1$, $p = 1$.

$a < b^k$ $\therefore$ case (iii) since $p \geq 0$ $T(n) = \Theta(n^k \log^p n)$

$\therefore T(n) = \Theta(n \log n)$.

4) $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$

$a=2,\ b=2,\ K=1,\ P=-1.$

$a = b^K$ ∴ case (ii) $P=-1$

(According to professors notes it can not be solved when $P < 0$).

$T(n) = \theta(n^{\log_b a} \log\log n)$

∴ $T(n) = \theta(n \log_2 \log n)$

5.) $T(n) = 0.5\,T\left(\frac{n}{2}\right) + \frac{1}{n}$

It can not be solved by masters theorem as ~~$a \geq 1$~~ $a \geq 1$ condition is not met here since $a = 0.5$.

Q3.) i) a d-ary heap can be represented as an array A[1...n]
The children of A[i] are A[2], A[3] ... A[d+1] and so on
For A[2] are A[d+2], A[d+3] ... A[2d+1],
children(i) = {di - d + 2, di - d + 3, ... di + 1}.

2) The number of nodes at level h is at most d^h.
The total nodes at height h is $1 + d + \dots + d^h = \Theta(d^h)$
$d^h = n$ ∴ height is $\Theta(\log_d n)$

3) Extract_max is the same for binary heaps.
Heapify operation on d-ary :

Heapify (A, i)

Largest = max {A[i], children(A[i])}
if L ≠ i then swap A[i] ↔ A[largest]
Heapify (A, i)

∴ Running time is $\Theta(d \log_d n)$. The d term is because at each iteration a node compares its value and the values of its d children to find the max which takes O(d) time.

4.)

```
insert (heap, n, item)
if heap is full
    return false
else
    n : n+1
    for i is n , i>1 , # set i=i.
        if item <= heap[i/2]
            heap[i] = heap[i/2]
        heap[i] = item
```

- The running time is $\Theta(\text{height}) = \Theta(\log_d n)$

c.) Running time is $O(\log_2 n)$ for $A[i] < k$.

```
Heap_Increase_Key(A, i, k)
if A[i] < k
    A[i] = k
    while i > 1 and A[Parent(i)] < A[i] do
        swap A[i] and A[Parent(i)]
        i = Parent(i)
```

8.4.) i) Since all elements are equal, Randomized-Quicksort will always return $q = R$. We have recurrence

$T(n) = T(n-1) + \Theta(n) = \Theta(n^2)$

2)

```
Partition(A, p, r)
    key = A[p]
    low = p
    High = p
    for j = p + 1 till r
        if A[j] < key
            swap A[j] - A[high + 1]
            A[High + 1] = A[Low]
            A[low] = old A[j]
            low ++, high ++
        else if A[j] == key
            swap A[High + 1] with A[j]
            High ++
    return [low, high]
```

3) Quicksort (A, p, r)

if p < r

(low, high) = Randomized-Partition (A, p, r)

Quicksort (A, p, low-1)

Quicksort (A, high+1, r)