Masters Theorem

Let T(n) be a monotonically increasing function that satisfies:

T(1) = c
T(n) = $aT(n/b) \ + \ f(n)$,

where,
a>=1, b>=1, c>0
n = Size of the problem
a = Number of subproblems in the recursion and a>=1
n/b = Size of each subproblem
f(n) = Cost of work done outside the recursive calls

There are 3 cases
If f(n) ∈ Θ (n^d) where d>=0, then

1. T(n) = Θ(n^d) if a < (b^d)
2. T(n) = Θ(n^dlog n) if a = (b^d)
3. T(n) = Θ($n^{(log(base\ b)\ a)}$) if a > (b^d)

Time complexities for

1) T (n) = 3T (n/2) + n

   Solution:
   T(n) = Θ($n^{(log(base\ 2)3)}$) = Θ($n^{(log\ 3)}$)
   a = 3, b = 2, d= 1

2) T (n) = 64T (n/8) − n^2(log n)

   Solution:
   Time complexity **can not be calculated**

3) T (n) = 2nT (n/2) + n^n

   Solution:
   Time complexity **can not be calculated**

4) T (n) = 3T (n/3) + n/2

Solution:
T(n) = Θ($n \log n$)
a = 3, b = 3, d = 1

5) T (n) = 7T (n/3) + n^2

Solution:
T(n) = Θ(n^2)
a = 7, b = 3, d =2