Assignment 4 problem 2: Goose Egging

a) The minimum number of drops should be n-1. Since the only way to get the lowest floor is drop the egg from the first floor, and increasing 1 floor each time. Floor n does not need to be drop since there must exists a floor that break the egg, so if floor n-1 does not break egg, then n must be the first floor that can break the egg.

b) If the egg from floor 4 is broken, then we need to drop it from the first floor to floor 3. And the total number of drop should be 4.
If the egg from floor 4 is not broken, then we can drop it from floor 6.
- If it breaks, then we try floor 5 (3 drops in this case)
- If it's not, then we are done since there must be a floor that can break the egg (2 drops in this case)

Therefore, the number of drops of the worst case is 4.

c) Firstly, drop an egg from floor 3
- If it breaks, then drop from floor 1
    - If it breaks, then floor 1 is the value that we want, 2 drops used
    - If it's not, then drop from floor 2
        - If it breaks, then floor 2 is what we want, 3 drops used
        - If it's not, then floor 3 is what we want, 3 drops used
- If it's not, then drop form floor 5
    - If it breaks, then drop from floor 4
        - If it breaks, then floor 4 is what we want, 3 drops used
        - If it's not, then floor 5 is what we want, 3 drops used
    - If it's not, then drop from floor 6
        - If it breaks, then floor 6 is what we want, 3 drops used
        - If it's not, then floor 7 is what we want, since there must exists a floor that can break the egg, 3 drops used

Therefore, the number of drops of the worst case is 3 drops, using 2 eggs.

d) Construct 2D-array N[ ][ ], where N[i][j] represents the number of minimum drops needed when there are i initial floors and j gooses eggs.
For drop at floor x, there are two cases:
Case1: The egg breaks, then we have one less eggs to deal with x-1 floors (the floors that are lower than x)
Case2: The egg does not break, then we have the same number of eggs to deal with (n-x) floors if there are n floors in total (the floors that are higher than x).
Therefore we can get a recursive relationship of N[i][j] by traverse each floor and find the minimum value of drops. (by finding the minimum value, we find the floor of next dropping)

The pseudo code is as follows:

$N[\,][\,]\ are\ initialized\ to\ be\ \infty$

$N[0][0] = 0$

$for\ i\ from\ 2\ to\ n\ do$

    $N[i][0] = \infty$

$for\ j\ from\ 1\ to\ k\ do$

    $N[1][j] = 0$

    $N[0][j] = 0$

$for\ i\ from\ 2\ to\ n\ do$

    $N[i][1] = i - 1$

$for\ j\ from\ 2\ to\ k\ do$

    $for\ i\ from\ 1\ to\ n\ do$

        $for\ x\ from\ 1\ to\ i\ do$

            $N[i][j] = \min(N[i][j], 1 + max(N[x-1][j-1], N[i-x][j]))$

The way we initialize the base case is because there must be an floor that can break the egg. For 0 floor in total, 0 drop needed. For 1 floor in total, 0 drop needed. For n floor with 1 egg, then n-1 drop needed (worst case). And if we have more than 1 floor, but we do not have egg, then we need infinity drops (i.e. can never get value we want.)

Focus on the recurrence relation in the pseudo code, the min() function is used due to we want to minimize the number of drops. x is the index that traverse all current floors to find the minimum drops of the worst case if we drop at floor x next. The max() function is also used since we want to find the drops of the worst case. Note that there are 2 cases for each drop, broken or not broken, and we need to find the worst case of these 2 cases. The '1+' in front of the max function represents the current dropping.

And since to determine N[i][j], all indices of values that we need are less or equal to its indices. Therefore, the induction applies and we can find the value that we are interested in which is N[n][k].

And the running time of the algorithm is dominated by the nested loops, which has the running time of $O(n^2k)$.