COMP 3011 Assignment 3

JAHJA Darwin, 16094501d

1.

The probability of a classroom does not enter by a student is $1 - 1/n$.

For each classroom $i \in \{1, 2, ..., n\}$, we can define a indicator random variable,

$$X_i = I\{\text{classroom } i \text{ is empty after all } m \text{ students have gone to the classrooms}\}$$

Therefore,

$$E[X_i] = Pr\{\text{classroom } i \text{ is empty after all } m \text{ students have gone to the classrooms}\} = (1 - 1/n)^m$$

Then, let X be the number of empty classrooms after all student m has gone to the classroom. By linear of expectation, $E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} (1 - 1/n)^m = n(1 - 1/n)^m$.

2.

If $k = 4$ and the initial of the counter is 0, in this case, if we decrement the counter, it would do 4 flips and the counter will turn from 0000 to 1111. Then, if we increment the counter, it would do another 4 flips and turn from 1111 to 0000. By performing such sequences of increment and decrement for $n/2$ times ($n/2$ increments + $n/2$ decrements), the total amortized cost will be 4n.

Similarly, for k-bits binary counter starting from 0, a decrement costs k flips, following up by a increment also costs k flips. Therefore, when performing such sequences of increment and decrement, the amortized cost per operation could be $\Theta(k)$.

3.

(a) l is the left boundary of the Z-box. The idea is to maintain an interval $[l, r]$ with max r so that $[l, r]$ is the prefix substring (substring which is also prefix).

(b) Pseudocode of Algorithm

Input: A string R and a string S.

Output: The longest suffix of S which is equal to a prefix of R.

```
Let string T = R + "$" + S
lenT = length of string T
Z = array of length lenT, with all elements initialized as 0
left = 0, right = 0

for k = 1 to lenT:
    if k > right do
        lt = rt = k
        while rt < lenT and T[right] == T[right-left]:
            right = right + 1
        Z[k] = right - left
        right = right - 1
    else
        // Operation inside the box
        k1 = k - left

        // if the value does not stretched till right bound then just copy it
        if Z[k1] < right-k+1 do
            Z[k] = Z[k1]

        else
            // Otherwise try to see if there are more matches
            left = k
            while right < lenT and T[right] == T[right-left] do
                right = right + 1
            Z[k] = right - left
            right = right - 1

    // Check whether it is a suffix
    if k + Z[k] == lenT:
        return R[0:Z[k]], which is the prefix of R equal to longest suffix of S

return None, which there is no suffix of S equal to prefix of R
```