Question 1

(a):

$$
\begin{aligned}
F(n) &= F(n-1) + cn + k, \quad n > 1, \quad F(1) = a \\
F(n) &= F(n-1) + cn + k = F(n-2) + c(n + (n-1)) + 2k = \ldots \\
F(n) &= F(n-(n-1)) + c\sum_{j=2}^{n} j + (n-1)k \\
F(n) &= a + c\left(\frac{n(n+1)}{2} - 1\right) + (n-1)k \\
F(n) &= \frac{c}{2}n^2 + \left(\frac{c}{2} + k\right)n + (a - c - k)
\end{aligned}
$$

(b):

$$
\begin{aligned}
M(n) &= M(n-1) + 1, \quad n > 1, \quad M(1) = 0 \\
M(n) &= M(n-1) + 1 = M(n-2) + 2 = \ldots = M(n-(n-1)) + (n-1) \\
M(n) &= M(1) + n - 1 = n - 1
\end{aligned}
$$

(c): Basic operation is multiplication.

$$M(n) = n - 1 = \Theta(n)$$

Question 2

No, we cannot definitively say which of the two will be faster for a specific case as we do not know the values of the constant terms of the expressions for computational complexity. Therefore Algorithm A, though having worse asymptotic time complexity may very well finish ahead of Algorithm B for a small number of elements, especially if the constant terms of Algorithm As complexity are much smaller than Bs. Algorithmic complexity is about how an algorithm scales, and not about absolute performance. That is, Algorithm B will eventually be faster than Algorithm A at some (large) input size.

Question 3

In a cycle, no node can be a source node, since all of the nodes have incoming edges. (Should provide example directed graph with a cycle.)

Question 4

(a)

k	$V(k)$	$E(k)$
1	2	0
2	4	4
3	6	12
4	8	24
5	10	40
6	12	60

(b)

$$V(k) = 2k$$

(c)

$$E(k) = E(k-1) + 2V(k-1), \quad k > 1 \quad E(1) = 0$$

(d)

$$E(k) = E(k-1) + 4(k-1) = E(k-2) + 4((k-1) + (k-2))$$

$$E(k) = E(k-(k-1)) + 4\sum_{j=1}^{k-1} j$$

$$E(k) = E(1) + 4\left(\frac{k(k-1)}{2}\right)$$

$$E(k) = 2k(k-1)$$

Question 5

```
function CountFeedLotPaths(G⟨V, E⟩,u,v)
    //G is the adjacency list of graph
    //V is the set of vertex
    //E is the set of edges
    //u, v the source and destination vertex

    cost[0 . . . V] = −1
    path[0 . . . V] = 0
    initialise(queue)
    inject(queue, u)
    cost[u] = 0
    path[u] = 1
    while queue is non-empty do
        x = eject(queue)
        for each edge (x, w) adjacent to x do
            if cost[w] = −1 then
                inject(queue, w)
                cost[w] = cost[x] + 1
                path[w] = path[x]
            else if cost[w] = cost[x] + 1 then
                path[w] = path[w] + path[x]
            end if
        end for
    end while
    return path[v]
end function
```

Question 6

```
function UPDATE(A[0, ..., n − 1],G,p,q)
    // Input: A an array of neural unit activation levels x_i in the neural network at time t
            G the neural unit interaction graph in adjacency matrix (n  n) format
    // Output: A an updated array containing the activation levels for each neural unit x_i
            in the neural network at time t + 1

    A_updated = array of zeros of length n
    for i = 0 to (n − 1) do
        delta_x = −A[i] + p tanh(A[i])
        for j = 0 to (n − 1) do
            if i ≠ j then
                delta_x = delta_x + qG[i][j]tanh(A[j])
            end if
        end for
        A_updated[i] = A[i] + delta_x
    end for

    A ← A_updated
    return A
end function
```