Problem 5

Let G be the graph with n nodes and m edges

Initialize a list, **order**, to contain nodes in topological order

```
def topological_sort(G, order):
    IF G is empty:
        Return order
    initialize a dictionary, incoming, keyed by vertices to store number of incoming edges of each node. Initially all values will be zero.
    FOR every edge u, v in G:
        Incoming[v] = Incoming[v] + 1
    WHILE (incoming is not empty):
        Initialize a list, zero, to contain nodes with no incoming edges
        FOR every k in incoming:
            IF incoming[k] == 0:
                add k to list zero
                delete incoming[k]
                add k to order
                delete k from G
        IF zero is empty:
            Return G
        topological_sort(G, order)
```

Time Complexity:

The first for loop will run for at most m times so O(m)

The while loop with a for loop will run for at most O(n+m)

So total time = O(m+n)