I certify that every answer in this assignment is the result of my own work; that I have neither copied off the Internet nor from any one else's work; and I have not shared my answers or attempts at answers with anyone else.

1: Given: a directed graph G = (V, E) represented with an adjacency-matrix.
(a) Write an algorithm for determining whether or not G contains a vertex with in-edges from every other vertex but no out-edge. (Definition: A directed edge (u, v) is an out-edge for vertex u and in-edge for vertex v. In other words, an edge that leaves vertex u is an out-edge for u; one that enters vertex v is an in-edge for v.) Your algorithm must run in time O(V). Hint: Focus on the Adj matrix. You are looking for a vertex with no out-edges.
What entries would you expect in the row corresponding to such a vertex? What about the column?
(b) Outline the idea behind your algorithm in a few sentences.
(c) Explain why its time complexity is O(V)

(a)

```
not_Ans ← []
pivot ← 1
for i in V and i not in not_Ans do
    for j in V and j not in not_Ans do
      if Adj[i,j] == 1 do
        not_Ans.add(i)
      if Adj[i,j] != pivot do
        not_Ans.add(j)
        if pivot = 1
          pivot = 0
        else
          pivot = 1
for i in V do
  if i is not in not_Ans
    return i
```

(b) The not_Ans array holds the value of all the vertices that do not meet our requirements (i.e. they're blacklisted from being a viable answer). The pivot variable is being used to determine if a variable shows the alternating 1's and 0's pattern that we're looking for by row, and the Adj[i,j] == 1 is checking that the given vertex is not pointing to anything else. The we search the not_Ans array to find out which vertex is NOT in this array, which should be our answer.
(c) although we have two for loops, the black listed array guarantees that this will run less than O(V^2) and in the realm of O(V) complexity

---

2: You are given a list of n professional wrestlers. Between any pair of professional wrestlers, there may or may not be a rivalry. You are also given a list of r pairs of wrestlers for which there are rivalries. Give an algorithm that determines whether or not it is possible to partition the set of wrestlers into good guys and bad guys such that each rivalry is between a good guy and a bad guy. Your algorithm must run in O(n + r)-time.

If it is possible to perform such a partition, then your algorithm should output the designations.
(Partition implies that each wrestler has to be designated a good guy or a bad guy and no wrestler can be both a good guy and a bad guy.)
Hint: Consider Breadth-first search

First we would set up our graph G where every vertex is a wrestler (w) and every edge is $\in$ r.
We would run the BFS algorithm once in order to visit all the vertices.
We can then search through d[] for every value of w and assign wrestlers with an even distance to "good" and then odd distanced wrestlers to "bad".
We should then check to see that every edge wrestler alternates between "good" and "bad"
We can assign the "good" an "bad" roles to a corresponding array A for all n wrestlers.
Then return A
This algorithm would take at most O(n+r) times as that's what BFS runs at, to check the edges would take O(r) time, and to assign good and bad to each wrestler would take O(n). Therfore the enitre algorithm runs the same O() time as BFS, O(n+r)