MATH6010-M02 Assignment02
Jia Mian Tan 120033990001

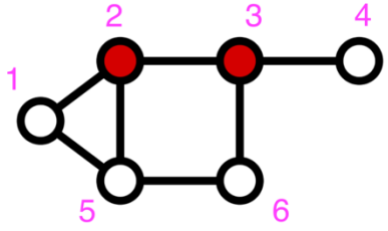
Find the dominating set using greedy approach

```
Algorithm findDominatingSetGreedy
Input: A graph G with n vertices and e edges
Output: A set U, the dominating set of vertices
1:  E ← adjacency matrix; U ←[]
2:  coveredV[1 ... n] ← array used to tell if the vertex is already covered: 1 if covered, 0 otherwise.
                        Initialized as 0 for all vertices.
3:  uncovNB[1 ... n] ← array used to store the number of uncovered neighbors for each vertex.
                       Initialized as the number of neighbors + 1(itself)
4:  while sum(coveredV) is not n do  (meaning there is still vertex(vertices) uncovered)
5:      maxV ← the vertex with largest number of uncovered neighbors
6:      add maxV to tail of U
7:      E[maxV][maxV] ← 0
8:      if maxV has never been covered
9:          uncovNB[maxV] ← uncovNB[maxV] − 1    (minus itself)
10:     for i from 1 to n
11:         if i is an uncovered neighbor of maxV
12:             remove the edge between i and maxV
13:             if maxV has never been covered
14:                 uncovNB[i] ← uncovNB[i] − 1
15:             if i has never been covered
16:                 uncovNB[maxV] ← uncovNB[maxV] − 1
17:                 for j from 1 to n
18:                     if there is still edge between i and j
19:                         uncovNB[j] ← uncovNB[j] − 1
20:                         if j has been covered before    (so now both i and j are covered)
21:                             remove the edge between i and j
22:             mark i as covered
23:     mark maxV as covered
24: return U
```

p/s: the vertices which have been covered but not yet in the dominating set are still eligible to be picked into the dominating set

<u>Driver code</u>
Sample 1: randomly generated 100x100 adjacency matrix, the number of minimum neighbor (δ) for each vertex is 3.
n = 100; δ = 3

Sample 2:
n = 6; δ = 1
Greedy Approach should be 2 → 3;
output in Python (index from 0): 1 → 2



Sample 3:
n = 10; $\delta = 3$
Greedy Approach should be 1 → 2 → 9;
output in Python (index from 0): 0 → 1 → 8