William Kelley
Assignment 5 - Arbitrage
Source: http://algo2.iti.kit.edu/sanders/courses/algdat03/sol3.pdf

```
procedure negCyc((V,E,c) : WeightedGraph) : List of List of Node
    n := card(V )
    distance : Array [0,...n][0,...n] of Real
    // This array tells where to go next in order to go from
    // node i to node j nextNode : Array[0,...n − 1][0,...n − 1] of Node

    // Fill up matrices with default values
    for i:=0 to n − 1 do
        for j:=0 to n − 1 do
            if (i, j) ∈ E then
                distance[i][j]:=c(i,j)
                nextNode[i][j]:=j
            else
                distance[i][j]:=+oo
                nextNode[i][j]:=nil

    // Loop all nodes
    for k:=0 to n − 1 do
        // Loop all pairs of nodes
        for i:=0 to n − 1 do
            for j:=0 to n − 1 do
                // Can node k help in constructing the path from i to j ?
                if distance[i][j] > distance[i][k]+distance[k,j] then
                    distance[i][j] := distance[i][k]+distance[k][j]
                    nextNode[i][j] := nextNode[i][k]
    // Construct result
    result : List of List of Node := ∅
    // Check out negative values at position (i,i)
    for i:=0 to n − 1 do
        if distance[i][i]<0 then
            negcyc : List of Node := < i >
            // Follow nextNodes until i has been reached again
            runnode := i
            repeat
                runnode := nextNode[runnode][i]
                negcyc.pushBack(runnode)
            until runnode==i
                result.pushFront(negcyc)

    return result
```