1. **[30 Points] Implementing the Floyd-Warshall Algorithm!** In this problem you will implement the Floyd-Warshall shortest path algorithm in your favorite programming language. For simplicity, you may assume that the graph will always have 5 vertices. The program will then find the shortest path between every pair of vertices. For each pair of vertices, \( i \) and \( j \), it must print the distance from \( i \) to \( j \) and then the actual shortest path from \( i \) to \( j \).

To see an example of how your program should behave, go to the directory `~hadas/public/algorithms` on turing or odin. The file `data` contains an adjacency matrix for the graph shown on page 626 of our textbook. Notice that the value 10000 is being used instead of \( \infty \). You can run Ran’s implementation, called `floyd`, by using Unix redirection. Simply type `floyd < data`.

You should turn in a printout of your source code as well a script showing your program running on the file `data` in Ran’s directory.

2. **[20 Points] Diameters.** In this problem we will consider connected undirected graphs with no cycles. There are no weights on the edges. The graphs are simply represented with adjacency lists. The distance between two vertices in such a graph is just the minimum number of edges on a path between the vertices. The diameter of the graph is the maximum distance between all pairs of vertices. Make sure that you understand this definition well before proceeding! Now, describe an algorithm for computing the diameter of a connected undirected graph with no cycles. Carefully analyze and explain the running time of your algorithm. *For full credit, your algorithm should be as fast as possible.*

3. **[10 Points] Finding a Negative Cost Cycle.** In class we showed how the Bellman-Ford Algorithm can be used to detect if a negative cost cycle exists in a directed graph. Now explain how an actual negative cost cycle can be found (that is, the vertices and edges in one such cycle can be printed out) if one exists. Your algorithm needs to only find one such negative cost cycle. Describe the algorithm and analyze its running time. Your algorithm should run in time polynomial in \( V \) and \( E \), but it’s not required to be the fastest possible algorithm for this problem.

4. **[25 points] An Algorithmic Get-Rich-Quick Scheme!** Your job at Millisoft was fun for awhile, but your reputation has gotten out and you’ve been hired back by the brokerage firm of Weil, Proffet, and Howe at a whopping salary. (We won’t be specific
about the actual amount, but let’s just say that your salary is most easily expressed using scientific notation!)

Weil, Proffet, and Howe has just entered the arbitrage business. Arbitrage is a money-making scheme involving anomalies in international currency exchange rates. For example, imagine that 1 U.S. dollar buys 0.8 Zambian kwachas, 1 Zambian kwacha buys 10 Mongolian tugriks, and 1 Mongolian tugrik buys 0.15 U.S. dollars. Then, by converting currencies, a trader can start with 1 U.S. dollar and buy $0.8 \times 10 \times 0.15 = 1.2$ U.S. dollars. By capitalizing on such anomalies quickly (before they’re detected and corrected by the markets), huge amounts of money can be made.

We’ll assume that we’re given $n$ currencies and the exchange rate between every pair of currencies. That is, we are given currencies $c_1, \ldots, c_n$ and an $n \times n$ table $R$ such that one unit of currency $c_i$ buys $R[i, j]$ units of currency $c_j$.

(a) We’ll begin with the following problem. Given the list of currencies and exchange rates and a particular currency $c_i$, determine the maximum amount of each currency that you can obtain, beginning with 1 unit of currency $c_i$. For this part of the problem assume that there exist no cycles that allow you to get arbitrarily rich via arbitrage. Describe two different algorithms for doing this - one based on a modification of Floyd-Warshall and one based on a modification of Bellman-Ford. (Your algorithms can be described in English. Pseudo-code is not necessary.) Explain briefly why each algorithm works. Finally, explain the running times of each of the two algorithms.

(b) Now assume that the exchange rates are such that it may be possible to get rich via arbitrage. That is, there may exist a cycle of currency exchanges that allows you to make more of your starting currency than you had initially. Describe how you could modify each of your algorithms to determine if it’s possible to profit from arbitrage with the given currencies and exchange rates. Prove that each of your modified algorithm works.

(c) Finally, describe and analyze the running time of an algorithm for finding the actual sequence of currencies to be exchanged that makes a profit using arbitrage, if such a cycle exists.