Problem 1. Suppose you’re running a lightweight consulting business—just you, two associates, and some rented equipment. Your clients are distributed between the East Coast and the West Coast, and this leads to the following question.

Each month, you can either run your business from an office in New York (NY) or from an office in San Francisco (SF). In month $i$, you’ll incur an operating cost of $N_i$ if you run the business out of NY; you’ll incur an operating cost of $S_i$ if you run the business out of SF. (It depends on the distribution of client demands for that month.)

However, if you run the business out of one city in month $i$, and then out of the other city in month $i+1$, then you incur a fixed moving cost of $M$ to switch base offices.

Given a sequence of $n$ months, a plan is a sequence of $n$ locations - each one equal to either NY or SF - such that the $i$th location indicates the city in which you will be based in the $i$th month. The cost of a plan is the sum of the operating costs for each of the $n$ months, plus a moving cost of $M$ for each time you switch cities. The plan can begin in either city.

Problem. Given a value for the moving cost $M$, and sequences of operating costs $N_1, \ldots, N_n$ and $S_1, \ldots, S_n$, find a plan of minimum cost. (Such a plan will be called optimal.)

(a) Show that the following algorithm does not correctly solve this problem, by giving an instance on which it does not return the correct answer.

```
1: for $i = 1$ to $n$ do
2:     if $N_i < S_i$ then
3:         Output “NY in month $i$”
4:     else
5:         Output “SF in month $i$”
6: end if
7: end for
```

(b) Give an efficient algorithm that takes values for $n$, $M$, and sequences of operating costs $N_1, \ldots, N_n$ and $S_1, \ldots, S_n$ and returns the cost of an optimal plan. (Hint: Think about a recurrence relation that describes the cost of a plan that ends in NY, and another one that described the cost of a plan that ends in SF.)

Problem 2. Let $G = (V, E)$ be a directed graph with nodes $v_1, \ldots, v_n$. We say that $G$ is an ordered graph if it has the following properties.

- Each edge goes from a node with a lower index to a node with a higher index. That is, every directed edge has the form $(v_i, v_j)$ with $i < j$. 


• Each node except $v_n$ has at least one edge leaving it. That is, for every node $v_i, i = 1, 2, \ldots, n - 1$, there is at least one edge of the form $(v_i, v_j)$.

The length of a path is the number of edges in it. The goal in this question is to solve the following problem: \textit{Given an ordered graph $G$, find the length of the longest path that begins at $v_1$ and ends at $v_n$.}

(a) Show that the following algorithm does not correctly solve this problem, by giving an example of an ordered graph on which it does not return the correct answer. Say what the correct answer is for your example and also what the algorithm finds.

\begin{verbatim}
1: function Longest-Path(G)
2:    w ← v_1
3:    L ← 0
4:   while there is an edge out of the node w do
5:      Choose the edge $(w, v_j)$ for the smallest possible $j$
6:        w ← v_j
7:        L ← L + 1
8:    end while
9:    return L
10: end function
\end{verbatim}

(b) Give an efficient algorithm that takes an ordered graph $G$ and returns the length of the longest path that begins at $v_1$ and ends at $v_n$. (Again, the length of a path is the number of edges in the path.)

\textbf{Problem 3.} Use the Bellman-Ford algorithm to find the shortest path from $s$ to $t$.

(a) Draw the table generated by the algorithm, and complete it as instructed.

(b) What does each entry on the table represent?

(c) Explain how you can modify the algorithm, or use the table, to find the shortest path itself.
Problem 4. Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

Let $G$ be an arbitrary flow network, with a source $s$, a sink $t$, and a positive integer capacity $c_e$ on every edge $e$. If $f$ is a maximum $s-t$ flow in $G$, then $f$ saturates every edge out of $s$ with flow (i.e., for all edges $e$ out of $s$, we have $f(e) = c_e$).

Problem 5. Network flow issues come up in dealing with natural disasters and other crises, since major unexpected events often require the movement and evacuation of large numbers of people in a short amount of time.

Consider the following scenario. Due to large-scale flooding in a region, paramedics have identified a set of $n$ injured people distributed across the region who need to be rushed to hospitals. There are $k$ hospitals in the region, and each of the $n$ people needs to be brought to a hospital that is within a half-hour’s driving time of their current location (so different people will have different options for hospitals, depending on where they are right now). At the same time, one doesn’t want to overload any one of the hospitals by sending too many patients its way. The paramedics are in touch by cell phone, and they want to collectively work out whether they can choose a hospital for each of the injured people in such a way that the load on the hospitals is balanced: Each hospital receives at most $\lceil n/k \rceil$ people.

Give a polynomial-time algorithm that takes the given information about the people’s locations and determines whether this is possible.