Problem 1.  30 points
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 Ni if you run the business out of NY; you’ll incur an operating cost of Si 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 ith location indicates the city in which you will be based in the ith 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.

The problem. Given a value for the moving cost M, and sequences of operating costs N1, . . . , Nn and S1, . . . , Sn, 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.

```
For i = 1 to n
  If Ni < Si then
    Output "NY in Month i"
  Else
    Output "SF in Month i"
End
```

(b) Give an efficient algorithm that takes values for n, M, and sequences of operating costs N1, . . . , Nn and S1, . . . , Sn, and returns the cost of an optimal plan.

Problem 2.  30 points
In class, we discussed a dynamic program to solve the shortest path problem in graphs with negative edge weights. Use the algorithm presented in class to find the shortest path from s to t.

Draw the table generated by the algorithm, and complete it as instructed. Then, answer the following questions:

1. How did you fill up the table? In other words, describe the order you entered values into the table.
2. What does each entry represent?

3. Explain how you can modify the algorithm, or use the table, to find the shortest path itself.

Problem 3.  [20 points]
Suppose you’re consulting for a bank that’s concerned about fraud detection, and they come to you with the following problem. They have a collection of \( n \) bank cards that they’ve confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic stripe with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, and we’ll say that two bank cards are equivalent if they correspond to the same account.

It’s very difficult to read the account number off a bank card directly, but the bank has a high-tech “equivalence tester” that takes two bank cards and, after performing some computations, determines whether they are equivalent. Their question is the following: among the collection of \( n \) cards, is there a set of more than \( n/2 \) of them that are all equivalent to one another?

Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only \( O(n \log n) \) invocations of the equivalence tester. Write the algorithm, and explain your reasoning.

Problem 4.  [20 points]
Take a look at the solution of problem 5 in assignment 7. Explain how this solution differs from your proposed solution (if it does). Be very clear in your description, it might be helpful to compare your pseudocode to the provided one in the solutions. Then, discuss the complexity of the proposed algorithm in the solutions. In other words, what is the Big-Oh of that algorithm?