This assignment should be completed individually. You are allowed to discuss the problems together, but not discuss the answers. You must write the solutions on your own, and list the collaborators you worked with. Submission instructions. This assignment is due by 11:59pm on Friday Dec 8, as a pdf file. Submit it to the PSet 8 folder in the CS231-assignments directory. Then, print it, and bring it to class the following Monday.

**Problem 1. [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, which can also be found in page 294, to find the shortest paths from all nodes in the graph 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. The output of this algorithm is the weight of the shortest path from the nodes to the target node. Explain how you can modify the algorithm, or use the table, to find the shortest path itself.
Problem 2. [25 points]

KT Chapter 6 - Problem 1. Let \( G = (V, E) \) be an undirected graph with \( n \) nodes. A subset of the nodes is called an independent set if no two of them are joined by an edge. Finding large independent sets is difficult in general; but here well see that it can be done efficiently if the graph is simple enough.

Call a graph \( G = (V, E) \) a path if its nodes can be written as \( v_1, v_2, ..., v_n \), with an edge between \( v_i \) and \( v_j \) if and only if the numbers \( i \) and \( j \) differ by exactly 1. With each node \( v_i \), we associate a positive integer weight \( w_i \).

Consider, for example, the five-node path drawn in Figure 6.28. The weights are the numbers drawn inside the nodes. The goal in this question is to solve the following problem: Find an independent set in a path \( G \) whose total weight is as large as possible.

(a) Give an example to show that the following algorithm does not always find an independent set of maximum total weight.

```
The "heaviest-first" greedy algorithm
Start with \( S \) equal to the empty set
While some node remains in \( G \)
    Pick a node \( v_i \) of maximum weight
    Add \( v_i \) to \( S \)
    Delete \( v_i \) and its neighbors from \( G \)
Endwhile
Return \( S \)
```

(b) Give an example to show that the following algorithm also does not always find an independent set of maximum total weight.

```
Let \( S_1 \) be the set of all \( v_i \) where \( i \) is an odd number
Let \( S_2 \) be the set of all \( v_i \) where \( i \) is an even number
(Note that \( S_1 \) and \( S_2 \) are both independent sets)
Determine which of \( S_1 \) or \( S_2 \) has greater total weight,
    and return this one
```

(c) Give an algorithm that takes an \( n \)-node path \( G \) with weights and returns an independent set of maximum total weight. Analyze the algorithm. The running time should be polynomial in \( n \), independent of the values of the weights.
Problem 3.  [25 points]
KT Chapter 6 - Problem 20. Suppose its nearing the end of the semester and you’re taking \( n \) courses, each with a final project that still has to be done. Each project will be graded on the following scale: It will be assigned an integer number on a scale of 1 to \( g > 1 \), higher numbers being better grades. Your goal, of course, is to maximize your average grade on the \( n \) projects.

You have a total of \( H > n \) hours in which to work on the \( n \) projects cumulatively, and you want to decide how to divide up this time. For simplicity, assume \( H \) is a positive integer, and you’ll spend an integer number of hours on each project. To figure out how best to divide up your time, you’ve come up with a set of functions \( f_i : i = 1, 2, \ldots, n \) (rough estimates, of course) for each of your \( n \) courses; if you spend \( h \leq H \) hours on the project for course \( i \), you’ll get a grade of \( f_i(h) \). (You may assume that the functions \( f_i \) are nondecreasing: if \( h < h' \), then \( f_i(h) \leq f_i(h') \))

So the problem is: Given these functions \( f_i \), decide how many hours to spend on each project (in integer values only) so that your average grade, as computed according to the \( f_i \), is as large as possible. In order to be efficient, the running time of your algorithm should be polynomial in \( n, g, \) and \( H \); none of these quantities should appear as an exponent in your running time. **Write an algorithm** that would decide on these hours, and analyze it.

Problem 4.  [25 points]
KT Chapter 6 - Problem 22. To assess how well-connected two nodes in a directed graph are, one can not only look at the length of the shortest path between them, but can also count the number of shortest paths.

This turns out to be a problem that can be solved efficiently, subject to some restrictions on the edge costs. Suppose we are given a directed graph \( G = (V, E) \), with costs on the edges; the costs may be positive or negative, but every cycle in the graph has strictly positive cost. We are also given two nodes \( v, w \in V \). **Give an efficient algorithm** that computes the number of shortest \( v - w \) paths in \( G \). (The algorithm should not list all the paths; just the number suffices.)