Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time, i.e., the running time must be bounded by a polynomial function of the input size.

Some Comments on Dynamic Programming Problems. If your solution consists of a dynamic programming algorithm, you must

- begin by *clearly specifying the set of subproblems you are using – describing what they mean in English as well as any notation you define*;
- clearly specify base cases and the recurrence used to solve the subproblems; the complete algorithm description should clearly show how the subproblems are solved (using the base cases and recurrence), and how the solution to the given problem is computed from the subproblems;
- analyze the running time of the algorithm you gave;
- prove correctness by explaining why your recurrence leads to the correct solution of the subproblems (using their English language description from the first bullet).

A description of a DP algorithm consisting of a piece of pseudocode without these explanations will not get full credit.

1. **Hiring in 1-, 2-, and 3-Day-Shifts** A small business lost their only employee, and in the current job market, they are having a hard time hiring a replacement. They have decided to use a temp agency to hire an employee for the next n days. The temp agency gives the option of hiring an employee for one, two, or three consecutive days and has provided you with a list of all prices, where $c_j(i)$ is the price of hiring an employee for j days starting on day i, for $j = 1, 2, 3$ and $i = 1, 2, \ldots, n - j + 1$.

   The business wants to find the minimum cost way to cover the next n days, making sure that they have exactly one employee every day (because if they have none, the work doesn't get done, and if they have more than one, the employees just end up chatting all day and the work also doesn't get done either).

   Design a dynamic programming algorithm that finds the solution to this problem in $O(n)$ time, and implement the algorithm. We suggest using subproblem $OPT(i)$ to be the minimum cost solution for days 1 up to i **with day i being the last day for the employee working that day**. Your algorithm needs to output the total cost of the hiring plan, as well as a list that for each day specifies whether the employee working that day is doing a 1-day, 2-day or 3-day shift.

   Input / output formatting and requirements
   Your algorithm is to read data from `stdin` in the following format:

   - The first line has one integer, n, representing the number of days for which the hiring plan needs to be made.
   - Each of the following $n - 2$ lines contains 3 integers, where the three integers on the i-th line of this part of the input give the cost of an employee starting on day i and working the next one, two or three days.
   - The next line has 2 integers, representing the cost of an employee starting on day $n - 1$ and working one day or two days.
   - The last line has 1 integer, representing the cost of an employee starting on day n and working one day.

   Your algorithm should output data to `stdout` in the following format:

   - The first line contains the minimum total cost s for having exactly one employee shows up for each of the n days.
   - The second line contains a list of values $a_1, a_2, \ldots, a_k$, with values separated by a space, and each $a_i \in \{1, 2, 3\}$ indicates the shift

length for the i-th hire. The number of values depends on the solution you found, but we know that $\sum_{i=1}^{k} a_i = n$ so that all days are covered.

Example
When the input is:

```
4
5 8 16
4 12 18
3 7
6
```

The output should be

```
15
2 2
```


because the cheapest solution for covering the four days in this case has cost 15, and has someone start on day 1 and work for two days (for a cost of 8), and then hire an employee who starts on day 3 and works for two days (for a cost of 7).

When the input is:


```
4
3 9 16
4 9 10
3 7
6
```

The output should be

```
13
1 3
```

because the cheapest solution for covering the four days in this case has cost 13, and has someone start on day 1 and work for one day (for a cost of 3), and then hire an employee who starts on day 2 and works for three days (for a cost of 10).

2. **The Ferry From Húsavík.** After seeing the movie Eurovision, Song Contest: The Story of Fire Saga, you decide to move to Húsavík in Iceland. The local inhabitants are very excited to have you there, first of all because you sing beautifully, but also because they have a question for which they need your expertise. The ferry from Húsavík to Ædarfossar has a set of regular passengers who come at the same time every day over the n minutes of the day: it is known that at the start of minute i, a fixed number of passengers c_i will arrive at the Húsavík dock. The Húsavíkians (?) ask you to schedule the departure times for the ferry to minimize the total wait time across all passengers.

   Due to a large cash influx from the European Union, Húsavík has a very large ferry which is large enough to transport $\sum_{i=1}^{n} c_i$ passengers at once, so it is able to take as many passengers at the same time as you want! The ferry takes $k < n$ minutes to travel to Ædarfossar, and k minutes to get back, that is, if the ferry departs right after the passengers arrive at minute i, then all passengers currently at the dock (including the c_i new arrivals) will arrive in Ædarfossar at the start of minute $i + k$, and the ferry is back at the dock in Húsavík at the start of minute $i + 2k$.

   Give an algorithm that finds the ferry departure times that minimize the total wait time of the passengers. Your input will be n, k, and a schedule of the passenger arrival counts c_i at each minute i. (Keep in mind that the last ferry may leave Húsavík *after* time n with the last passengers of the day; because it can be the case that the ferry is still en route between Ædarfossar and Húsavík at time n.)

   Example: Suppose $k = 2, n = 8$ and you have the following schedule of passenger arrivals:

Minute	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
c_i	1	5	0	3	1	2	0	7

   Suppose we schedule the ferry at minute 1, minute 5, and minute 9 (so the ferry is constantly moving). The 2 passengers arriving right when the ferry is scheduled (at minutes 1, 5, and 9) will only wait 2 minutes to arrive inÆdarfossar; the 10 passengers arriving 1 minute before the departure (arriving at minutes 4 and 8) will wait 3 minutes each, the 7 passengers arriving 3 minutes before the scheduled departures (arriving at minutes 2 and 6) will wait 5 minutes each (and there happen to be no passengers arriving 2 minutes before a departure in

this case). Thus the total wait for the 18 passengers is

$$(2 \cdot 2) + (10 \cdot 3) + (0 \cdot 4) + (7 \cdot 5) = 69.$$

However, if we schedule the departures to be at minutes 2 and 8, we get the optimal total wait of

$$(12 \cdot 2) + (1 \cdot 3) + (2 \cdot 4) + (1 \cdot 5) + (3 \cdot 6) = 58.$$

Note that in this case, optimizing the total wait time is equivalent to optimizing the average wait time: the average is simply the total wait time divided by the total number of passengers, which is known in advance.

Your algorithm should run in polynomial time; for extra fun (but no extra points), see if you can bring the running time of your algorithm down to $O(bm)$.

3. **Counting Shortest Paths.** Consider a directed graph $G = (V, E)$ where the cost of edge e, c_e, can be negative. Given two nodes s and t, we have seen in class how to compute the length of the min-cost path in G assuming G has no negative cost cycles. In this problem, assume that all cycles have strictly positive costs.

Klaas lives in Amsterdam, and has found a job in The Hague. The job doesn't allow him to work remotely, and Klaas wants to assess how long his commute would be from his home to the company before he signs the contract. He uses the algorithm from class to measure the shortest path (in terms of duration, i.e., the travel time) from his home to the company, where his graph takes into account various modes of transportation. Because of the uncertainties of travel modes in the Netherlands (trains may not run in case of snow or too many leaves on the tracks, highways may get closed due to construction, his bike may get a flat, to name just a few), he also wants to make sure that there is more than one shortest path available to work. In this case, in addition to the duration for the shortest path from s to t, you would also like to know how many paths there are from s to t of this shortest duration.

(a) (2.5 points) Previously, we computed the length (duration) of the shortest path from v to t using *at most* i edges and stored the value in $OPT(i, v)$. Here, consider the minimum duration path using *exactly* i edges (rather than at most i edges). Give

a polynomial time algorithm to find the minimum duration path from s to t using *exactly* i edges.

(b) (5 points) In addition to the duration for the shortest path from s to t with i edges, also compute the number of different shortest paths from v to t with the same duration that use exactly i edges. (There can be exponentially many shortest paths, so don't try finding them all, because this may not be possible in polynomial time; you just need to count them.)

(c) (2.5 points) Show how to compute the total number of paths from s to t of minimum duration, this time using any number of edges.