**Problem 3-1.** You work for a manufacturing company. The owner decides that the company is spending too much money on shipping items from the factory (where the items are produced) to the warehouse (where the items are stored). You have been asked to use your algorithms knowledge to figure out the best shipping strategy.

Every day the factory produces $n$ items and the same items are produced in the same order every day. As the items arrive at the loading dock over the course of the day they must be packaged into boxes and shipped out. Items are boxed in contiguous groups according to their arrival order; for example, items 1, 2, 3, 4 might be placed in the first box, items 5, 6, 7 in the second, items 8, 9, 10, 11, 12, 13 in the third, and so on. Alternatively, items 1, 2, 3 may be placed in the first box, 4, 5, 6, 7 in the second and so on. You may not skip items or ship them out of order. For example, items 1, 2, 5 in the first box and 3, 7, 8 in the second box, and so on, is illegal. Each item $i$ has two attributes, value $v(i)$ and weight $w(i)$, and you know the values and weights of all items.

There are two types of shipping options available to you:

- **Limited-Value Boxes:** One of your shipping companies offers insurance on boxes and hence requires that any box shipped through them must contain no more than $V$ units of value. Therefore, if you pack items into such a limited-value box, you can place as much weight in the box as you like, as long as the total value in the box is at most $V$. Each box of this type costs $a$ dollars to ship.

- **Limited-Weight Boxes:** Another of your shipping companies lacks the machinery to lift heavy boxes, and hence requires that any box shipped through them must contain no more than $W$ units of weight. Therefore, if you pack items into such a limited-weight box, you can place as much value in the box as you like, as long as the total weight inside the box is at most $W$. Each box of this type costs $b$ dollars to ship.

You may assume that every individual item has value at most $V$ and weight at most $W$. Your job is to determine the optimal way of packaging items into boxes with specified shipping options, so that shipping costs are minimized. You may choose different shipping options for different boxes.
(a) To start with, assume that \( a = b \) — that is, the cost is the same regardless of whether you use limited weight box or limited value box. Argue that this admits a greedy strategy. Give an algorithm to pack items into boxes and prove the correctness of your algorithm.

(b) Now assume that the cost of minimum weight box is different from the cost of minimum value box. Give an algorithm to find a way to pack items into boxes that costs you the minimum amount of money.

   \textit{Hint:} Use dynamic programming. It is sufficient to calculate the minimum cost, you needn’t show the back-tracing that lets you actually calculate which items go in which box.

**Problem 3-2.** You are given a set of \( n \) jobs, each of which runs in unit time. Job \( i \) has an integer-valued deadline time \( d_i \geq 0 \) and a real-valued penalty \( p_i \). Jobs may be scheduled to start at any non-negative integer time (0, 1, 2, etc), and only one job may run at a time. If job \( i \) completes at or before time \( d_i \), then it incurs no penalty; otherwise, it incurs penalty \( p_i \).

We want to design an algorithm such that we minimize the total penalty incurred.

(a) Let us consider some candidate algorithms. Lets say that you schedule the job with the earliest deadline first — meaning you consider jobs in order of their deadlines and schedule the jobs in this order in the first slot that is free. Can you generate a counter example where this greedy algorithm gives us the incorrect result — that is, a different algorithm produces smaller penalty?

(b) Say you schedule the job with the largest penalty as early as possible — again, you sort the jobs by penalty and then schedule them as early as possible. Can you generate a counter-example for this greedy strategy?

(c) It should be clear that we should consider both penalty and deadlines. In general, it seems useful to ensure that we manage to schedule the large penalty jobs, so we should consider them first in our greedy ordering. But we shouldn’t schedule them as early as possible since their deadline may be later than some lower penalty jobs. It is sufficient to schedule them so they do not miss their deadline. In particular consider the following strategy: Sort the tasks by penalty (highest to smallest) breaking ties in terms of deadlines (latest to earliest). Now consider jobs in this order, but schedule them in a slot as late as possible to still avoid penalties when possible.

Design a greedy algorithm based on this idea and provide the pseudocode and analyze its running time.

(d) Prove the correctness of the algorithm.

   \textit{Hint:} Here is a useful set up: Consider for contradiction that our algorithm doesn’t get the minimum penalty when it generates a schedule \( S \) for some set of jobs. Some
other schedules achieve a smaller total penalty. Since our penalty in $S$ must be strictly greater than the minimum, all schedules achieving the minimum penalty must disagree with their assignment of tasks to slots at some point in the greedy ordering considered by our algorithm. Let us consider, among the schedules that achieve minimum total penalty, some schedule $S'$ that schedules the longest possible prefix of jobs in our greedy ordering at the same time as in $S$. Let $i^*$ be the first index in the greedy ordering in which $S'$ makes a different choice than $S$. Now consider all cases and try to reach a contradiction.