Algorithms
Dynamic Programming Group Project (75 pts)

This project requires you to first theoretically solve the dynamic programming problem below and then write a program that implements your solution. [1] **You are not allowed to use the internet or consult any references. The only people you can work with on this project are your group members.**

1. *Problem Description*: You are helping to run a big data computing system capable of processing several terabytes of data per day. For each of n days, you're presented with a quantity of data (on day i, you're presented x_i terabytes). For each terabyte you process, you receive a fixed revenue, but any unprocessed data becomes unavailable at the end of the day (i.e., you can't work on it in any future day).

   The amount of data you can process on a given day goes down with each day that passes since the most recent reboot of the system. On the first day after a reboot, you can process s_1 terabytes, on the second day after a reboot you can process s_2 terabytes, and so on, upto s_n; we assume $s_1 > s_2 > s_3 > \cdots > s_n > 0$. To get the system back to peak performance, you can choose to reboot it; but on any day you choose to reboot the system, you can't process any data at all.

   Given the amounts of available data $x_1, x_2, \ldots, x_n$ for the next n days, and given the profile of your system as expressed by $s_1, s_2, \ldots, s_n$ (and starting with a freshly rebooted system on Day 1), choose the days on which you are going to reboot so as to maximize the total amount of data you process.

   Example. Suppose $n = 4$ and the values of x_i and s_i are given by the following table:

	DAY 1	DAY 2	DAY 3	DAY 4
x	10	1	7	7
s	8	4	2	1

   The best solution would be to reboot on Day 2 only; this way you process 8 on Day 1, 0 on Day 2, 7 on Day 3 and 4 on Day 4 for a total of 19. Note that if you didn't reboot at all, you'd process $8 + 1 + 2 + 1 = 12$.

2. *Deliverables*: Please submit a **hard copy** of all of the items requested below. Please don't send this by email as I would have to print it out anyway.

   (a) [40 pts] Theory: Devise a dynamic programming algorithm that optimizes the number of terabytes processed. Demonstrate each of the following dynamic programming steps:

      i. Describe the main idea of your approach including how you break the problem into smaller recursive problems [20 pts].

      ii. Write pseudocode for your dynamic programming algorithm [10 pts].

      iii. Develop a traceback algorithm that returns the decision on each day (i.e., process vs reboot). [10 pts].

   (b) [10 pts] Theory: Derive the complexity of your algorithm in terms of n.

   (c) [15 pts] Implementation: Implement your algorithm and submit the print-out of your code. If you were unable to get your code to compile/run, please state this clearly. Although we don't plan to run everybody's code, we might choose a few groups randomly and ask them to demonstrate that their code works. (It would look pretty bad if you claim your code runs, but it doesn't!)

[1] The problem has been adapted from the text by Kleinberg and Tardos

(d) [10 pts] Implementation: Demonstrate that your code works correctly by showing its results on a small example. (Provide the example you used so that we can verify this while grading.)

(e) Submit group effort percentages and indicate who did what. These must be submitted individually by each group member. See syllabus to determine how I will use this to compute your individual grade.