EE 360C — Algorithms — Summer 2013
Homework #5

Due: July 24, 2013 11:30am (in class)

Homework problems are to be done individually. You may discuss the problem and general concepts with other students, but you must write your solutions independently.

Each question is worth 10 points (1a, 1b, 2). Maximum possible score is 30.

Whenever you give an algorithm, prove that it is correct.

---

1. Suppose you're managing a consulting team of expert computer hackers, and each week you have to choose a job for them to undertake. Now, as you can well imagine, the set of possible jobs is divided into those that are *low-stress* (e.g., setting up a Web site for a class at the local elementary school) and those that are *high-stress* (e.g., protecting the nation's most valuable secrets, or helping a desperate group of Cornell students finish a project that has something to do with compilers). The basic question, each week, is whether to take on a low-stress job or a high-stress job.

   If you select a low-stress job for your team in week i, then you get a revenue of $l_i > 0$ dollars; if you select a high-stress job, you get a revenue of $h_i > 0$ dollars. The catch, however, is that in order for the team to take on a high-stress job in week i, its required that they do no job (of either type) in week $i-1$; they need a full week of prep time to get ready for the crushing stress level. On the other hand, it's okay for them to take a low-stress job in week i even if they have done a job (of either type) in week $i-1$.

   So, given a sequence of n weeks, a plan is specified by a choice of "low-stress," "high-stress," or "none" for each of the n weeks, with the property that if "high-stress" is chosen for week $i > 1$, then "none" has to be chosen for week $i-1$. (It's okay to choose a high-stress job in week 1.) The value of the plan is determined in the natural way: for each i, you add l_i to the value if you choose "low-stress" in week i, and you add h_i to the value if you choose "high-stress" in week i. (You add 0 if you choose "none" in week i.)

   The problem. Given sets of values $l_1, l_2, \ldots, l_n$ and $h_1, h_2, \ldots, h_n$,find a plan of maximum value. (Such a plan will be called *optimal*.)

   Example. Suppose $n = 4$, and the values of l_i and h_i are given by the following table. Then the plan of maximum value would be to choose "none" in week 1, a high-stress job in week 2, and low-stress jobs in weeks 3 and 4. The value of this plan would be 0 + 50 + 10 + 10 = 70.

	Week 1	Week 2	Week 3	Week 4
l	10	1	10	10
h	5	50	5	1

   (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 iterations i = 1 to n
     If h_{i+1} > l_i + l_{i+1} then
       Output "Choose no job in week i"
       Output "Choose a high-stress job in week i+1"
       Continue with iteration i+2
     Else
   ```

```
        Output "Choose a low-stress job in week i"
        Continue with iteration i+1
      Endif
    End
```

To avoid problems with overflowing array bounds, we define $h_i = l_i = 0$ when $i > n$. In your example, say what the correct answer is and also what the above algorithm finds.

(b) Give an efficient algorithm that takes values for $l_1, l_2, \ldots, l_n$ and $h_1, h_2, \ldots, h_n$ and returns the *value* of an optimal plan.

2. In a word processor, the goal of "pretty-printing" is to take text with a ragged right margin, like this,

```
Call me Ishmael.
Some years ago,
never mind how long precisely,
having little or no money in my purse,
and nothing particular to interest me on shore,
I thought I would sail about a little
and see the watery part of the world.
```

and turn it into text whose right margin is as "even" as possible, like this.

```
Call me Ishmael. Some years ago, never
mind how long precisely, having little
or no money in my purse, and nothing
particular to interest me on shore, I
thought I would sail about a little
and see the watery part of the world.
```

To make this precise enough for us to start thinking about how to write a pretty-printer for text, we need to figure out what it means for the right margins to be "even." So suppose our text consists of a sequence of words, $W = w_1, w_2, \ldots, w_n$, where w_i consists of c_i characters. We have a maximum line length of L. We will assume we have a fixed-width font and ignore issues of punctuation or hyphenation.

A formatting of W consists of a partition of the words in W into lines. In the words assigned to a single line, there should be a space after each word except the last; and so if $w_j, w_{j+1}, \ldots, w_k$ are assigned to one line, then we should have

$$\left[\sum_{i=j}^{k-1}(c_i + 1)\right] + c_k \leq L.$$

We will call an assignment of words to a line *valid* if it satisfies this inequality. The difference between the left-hand side and the right-hand side will be called the *slack* of the line—that is, the number of spaces left at the right margin.

Give an efficient algorithm to find a partition of a set of words W into valid lines, so that the sum of the *squares* of the slacks of all lines (including the last line) is minimized.