Semester Break Problem Solving Competition

Contest 2

September 17, 2012

Problem A – Hop

The cows have reverted to their childhood and are playing a game similar to human hopscotch. Their hopscotch game features a line of N (3 ≤ N ≤ 250,000) squares conveniently labeled 1..N that are chalked onto the grass.

Like any good game, this version of hopscotch has prizes! Square i is labeled with some integer monetary value Vi (-2,000,000,000 ≤ Vi ≤ 2,000,000,000). The cows play the game to see who can earn the most money.

The rules are fairly simple:

* A cow starts at square "0" (located just before square 1; it has no monetary value).

* She then executes a potentially empty sequence of jumps toward square N. Each square she lands on can be a maximum of K (2 ≤ K ≤ N) squares from its predecessor square (i.e., from square 1, she can jump outbound to squares 2 or 3 if K==2).

* Whenever she wishes, the cow turns around and jumps back towards square 0, stopping when she arrives there. In addition to the restrictions above (including the K limit), two additional restrictions apply:

* She is not allowed to land on any square she touched on her outbound trip (except square 0, of course).

* Except for square 0, the squares she lands on during the return trip must directly precede squares she landed on during the outbound trip (though she might make some larger leaps that skip potential return squares altogether).

She earns an amount of money equal to the sum of the monetary values of all the squares she jumped on. Find the largest amount of cash a cow can earn.

By way of example, consider this six-box cow-hopscotch course where K has the value 3:

```
Square Num:    0      1      2      3      4      5      6
             +---+  +---+  +---+  +---+  +---+  +---+  +---+
             |///|--|   |--|   |--|   |--|   |--|   |--|   |
             +---+  +---+  +---+  +---+  +---+  +---+  +---+
     Value:    -      0      1      2     -3      4      5
```

One (optimal) sequence Bessie could jump (shown with respective bracketed monetary values) is: 1[0], 3[2], 6[5], 5[4], 2[1], 0[0] would yield a monetary total of 0+2+5+4+1+0=12.

If Bessie jumped a sequence beginning with 0, 1, 2, 3, 4, ... then she would be unable to return since she could not legally jump back to an untouched square.

INPUT

* Line 1: Two space separated integers: N and K

* Lines 2..N+1: Line i+1 contains a single integer: V_i

OUTPUT

- Line 1: A single line with a single integer that is the maximum amount of money a cow can earn

SAMPLE IO

Input	Output

6 3
0
1
2
-3
4
5
	12

Problem B – Slowdown

Every day each of Farmer John's N (1 <= N <= 100,000) cows conveniently numbered 1..N move from the barn to her private pasture. The pastures are organized as a tree, with the barn being on pasture 1. Exactly N-1 cow unidirectional paths connect the pastures; directly connected pastures have exactly one path. Path i connects pastures A_i and B_i (1 <= A_i <= N; 1 <= B_i <= N).

Cow i has a private pasture P_i (1 <= P_i <= N). The barn's small door lets only one cow exit at a time; and the patient cows wait until their predecessor arrives at her private pasture. First cow 1 exits and moves to pasture P_1. Then cow 2 exits and goes to pasture P_2, and so on.

While cow i walks to P_i she might or might not pass through a pasture that already contains an eating cow. When a cow is present in a pasture, cow i walks slower than usual to prevent annoying her friend.

Consider the following pasture network, where the number between parentheses indicates the pastures' owner.

```
        1 (3)
       / \
  (1) 4   3 (5)
     / \
(2) 2   5 (4)
```

First, cow 1 walks to her pasture:

```
        1 (3)
       / \
  [1] 4*  3 (5)
     / \
(2) 2   5 (4)
```

When cow 2 moves to her pasture, she first passes into the barn's pasture, pasture 1. Then she sneaks around cow 1 in pasture 4 before arriving at her own pasture.

```
        1 (3)
       / \
  [1] 4*  3 (5)
     / \
[2] 2*  5 (4)
```

Cow 3 doesn't get far at all -- she lounges in the barn's pasture, #1.

```
        1* [3]
       / \
  [1] 4*  3 (5)
     / \
[2] 2*  5 (4)
```

Cow 4 must slow for pasture 1 and 4 on her way to pasture 5:

```
        1* [3]
       / \
  [1] 4*   3 (5)
     / \
[2] 2*   5* [4]
```

Cow 5 slows for cow 3 in pasture 1 and then enters her own private pasture:

```
        1* [3]
       / \
  [1] 4*   3*[5]
     / \
[2] 2*   5* [4]
```

FJ would like to know how many times each cow has to slow down.

INPUT

* Line 1: Line 1 contains a single integer: N
* Lines 2..N: Line i+1 contains two space-separated integers: A_i and B_i
* Lines N+1..N+N: line N+i contains a single integer: P_i

OUTPUT

* Lines 1..N: Line i contains the number of times cow i has to slow down.

SAMPLE IO

Input	Output
5	0
1 4	1
5 4	0
1 3	2
2 4	1
4	
2	
1	
5	
3	

Problem C – Jump

An n × n game board is populated with integers, one nonnegative integer per square. The goal is to jump along any legitimate path from the upper left corner to the lower right corner of the board. The integer in any one square dictates how large a step away from that location must be. If the step size would advance travel off the game board, then a step in that particular direction is forbidden. All steps must be either to the right or toward the bottom. Note that a 0 is a dead end which prevents any further progress.

Consider the 4 × 4 board shown in Figure 1, where the solid circle identifies the start position and the dashed circle identifies the target. Figure 2 shows the three legitimate paths from the start to the target, with the irrelevant numbers in each removed.

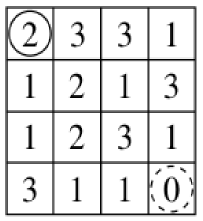

Figure 1

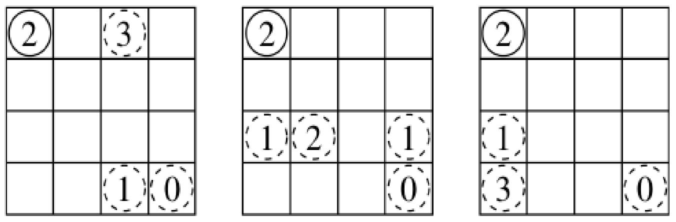

Figure 2

Your task is to write a program that determines the number of legitimate paths from the upper left corner to the lower right corner.

INPUT

The input file contains a first line with a single positive integer n, 4 <= n <= 100, which is the number of rows in this board. This is followed by n rows of data. Each row contains n integers, each one from the range 0...9.

OUTPUT

The output file jump.out should consist of a single line containing a single integer, which is the number of legitimate paths from the upper left corner to the lower right corner.

SAMPLE IO

Input	Output

4
2 3 3 1
1 2 1 3
1 2 3 1
3 1 1 0
	3

GRADING

The number of legitimate paths can be quite big. Only 70% of the score can be achieved using a 64-bit integer variable (long long int in C, Int64 in Pascal). It is guaranteed that all inputs will lead to a number of legitimate paths that can be written with no more than 100 digits.