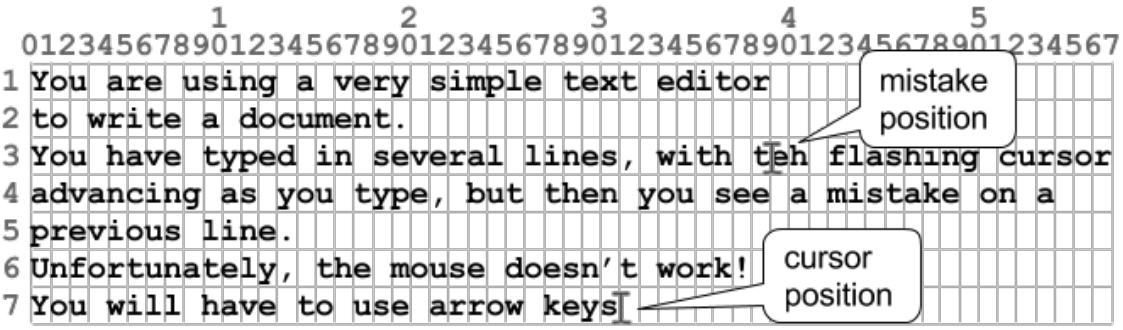
UCF Local Contest — September 2, 2017

Editor Navigation

filename: `editor`
(*Difficulty Level:* `Medium`)

You are using a very simple text editor to create a document. You have typed in several lines, with the flashing cursor advancing as you type, but then you see a mistake on a previous line. Unfortunately, the mouse doesn't work! You will have to press arrow keys to move the cursor back to the position where you can fix the mistake. Of course, you want to get to this position as quickly as possible.



This simple editor uses a monospace font, so each character is exactly the same width. The cursor can be at the beginning of the line (before the first character), end of the line (after the last character), or at a horizontal position between two characters on a given line. The following keys can be pressed to move the cursor (each keypress is independent of any preceding keypress):

← (Left arrow key)	Moves the cursor one character to the left on the same line, unless the cursor is at the beginning of the line, in which case it moves to the end of the previous line. If the cursor is at the beginning of the line and there is no previous line, the cursor does not move.
→ (Right arrow key)	Moves the cursor one character to the right on the same line, unless the cursor is at the end of the line, in which case it moves to the beginning of the next line. If the cursor is at the end of the line and there is no next line, the cursor does not move.
↑ (Up arrow key)	Moves the cursor up to the previous line; keeps the same horizontal position, unless the previous line is shorter, in which

	case the cursor goes to the end of the previous line. If there is no previous line, the cursor does not move.
↓ (Down arrow key)	Moves the cursor down to the next line; keeps the same horizontal position, unless the next line is shorter, in which case the cursor goes to the end of the next line. If there is no next line, the cursor does not move.

The Problem:

Given the line lengths of a text file that was loaded in this simple editor, along with the current cursor position and a different, desired cursor position (e.g., to fix a mistake), you are to determine the minimum number of keypresses, using arrow keys only, required to move the cursor to the desired position.

The Input:

The first input line contains a positive integer, n, indicating the number of editor navigation scenarios to process. Each scenario will occupy exactly 4 input lines. The first line of each scenario contains an integer f ($1 \leq f \leq 120$), indicating the number of lines of text in the file that is loaded in the editor. The next input line contains f integers, s_1 to s_f, where each value s_i ($0 \leq s_i \leq 80$) indicates the number of characters on line i of the file; the values will be separated by exactly one space. A value $s_i = 0$ means that there are no characters on line i. The newline character (common character indicating end of a line) does not count as a character on the line.

The third input line of each scenario will contain two integers (separated by a space) providing the current cursor position in the file: r_c ($1 \leq r_c \leq f$) and c_c ($0 \leq c_c \leq 80$), where r_c represents the line of the file, counting from 1 (as with i), and c_c represents the horizontal position of the cursor on that line, 0 for the beginning (before the first character). It is guaranteed that the cursor position is valid, so if, for instance, $r_c = 17$ and $s_{17} = 56$, then $0 \leq c_c \leq 56$; the maximum value of c_c is the end of the line. The fourth input line of each scenario is similar to the third and indicates the desired cursor position to begin fixing the mistake, i.e., this line consists of two integers separated by a space, r_m ($1 \leq r_m \leq f$) and c_m ($0 \leq c_m \leq 80$). The constraints on the values for r_c and c_c also apply to r_m and c_m.

The Output:

For each scenario in the input, output a single integer on a line by itself indicating the minimum number of keypresses needed to move the cursor from its current position (r_c, c_c) to the desired position (r_m, c_m) for fixing the mistake.

Sample Input:

```
2
7
39 20 57 54 14 38 31
7 31
3 39
3
15 30 20
1 12
3 3
```

Sample Output:

```
21
8
```

Explanation for Sample Output:

For Case #1, one possible sequence for the minimum number of keypresses to move the cursor from its current position to the desired position is: Up, Up, Right, Up, Left, Up, Left 15 times.