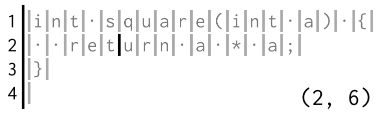
# Text editor

Robert is competing in CEOI 2024. He has almost finished his solution to the hardest problem of the day, and not just that, he's pretty sure it will get 100 points! There is just one small issue: He made a typo! What's worse, his favorite computer mouse, which he had been using since 2008, seems to have finally broken and isn't responding at all. Therefore, he will have to navigate to the typo using the arrow keys on his keyboard.

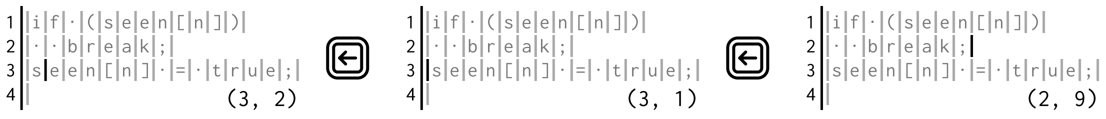
Robert's program has $N$ lines with lengths $l_1, l_2, \ldots, l_N$. Robert always ends his programs with an empty line, therefore $l_N = 0$. The cursor can be placed between two characters, as well as at the start or end of a line. As such, line $i$ has $l_i + 1$ possible cursor positions (called columns) numbered from $1$ to $l_i + 1$. For example, this is what a cursor placed on line 2 in column 6 would look like:



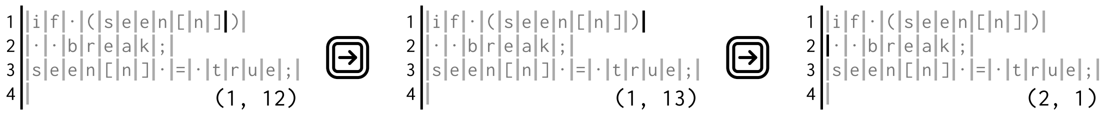
Robert wants to move his cursor from line $s_l$ column $s_c$ to line $e_l$ column $e_c$. He would like to know the minimum number of key presses needed to do so.

The horizontal arrow keys are quite simple. Pressing *left* will move the cursor to the previous column, unless the cursor was at the start of a line, in which case it will move to the end of the previous line. Similarly, pressing *right* will move the cursor to the next column, or to the start of the next line if the cursor was at the end of a line.

For example, *left* presses can look like this:



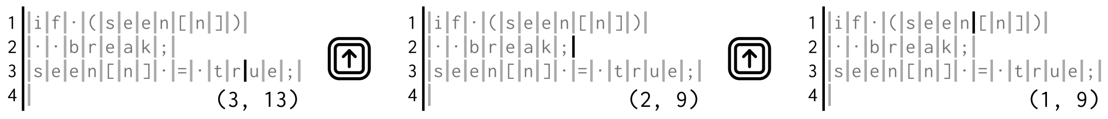
And *right* presses can look like this:



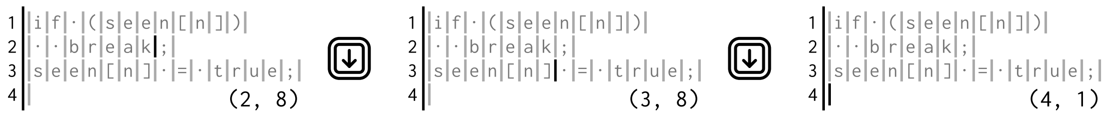
Pressing *left* at the very start of the file or pressing *right* at the very end of the file will have no effect.

The vertical arrow keys are slightly more complicated. Pressing *up* will move the cursor to the previous line and pressing *down* will move it to the next line, without changing the column number. However, if this would put the cursor past the end of the new line, the cursor will jump to the end of that line instead.

For example, *up* presses can look like this:



And *down* presses can look like this:



If pressing *up* or *down* would put the cursor on a line that doesn't exist, the cursor won't move at all.

## Input

The first line of input contains the integer $N$ — the number of lines of Robert's solution. The second line contains two integers $s_l$ and $s_c$ separated by spaces — the initial cursor position. Similarly, the third line contains two integers $e_l$ and $e_c$ — the target cursor position. The fourth line contains $N$ space-separated integers $l_1, l_2, \ldots, l_N$ — the length of each line.

## Output

Your program should output a single line containing a single integer — the minimum number of key presses to move the cursor from $(s_l, s_c)$ to $(e_l, e_c)$.

## Examples

### Example 1

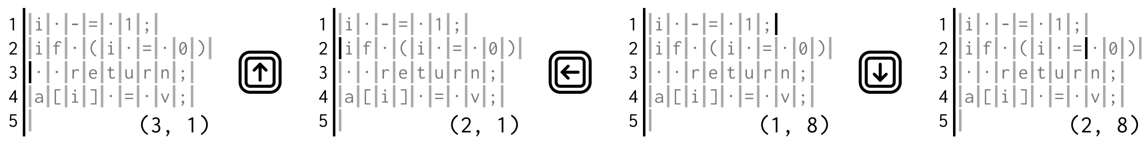
Input:

```
5
3 1
2 8
7 10 9 9 0
```

Output:

```
3
```

Robert can move the cursor to the target position using three key presses by pressing *up*, *left* and *down*, in that order:



Alternatively, he could move his cursor to the target position equally quickly by pressing *left*, *up* and *down*. It can be easily shown that it's impossible to reach the target position using at most two key presses.

### Example 2

Input:

```
5
1 20
3 25
25 10 40 35 0
```

Output:

```
16
```

The shortest possible key press sequence consists of two *down* presses followed by fourteen *right* presses.

## Constraints

- $1 \leq N \leq 10^6$
- $0 \leq l_i \leq 10^9$ (for each $i$ such that $1 \leq i \leq N$)
- $l_N = 0$
- $1 \leq s_l, e_l \leq N$
- $1 \leq s_c \leq l_{s_l} + 1$
- $1 \leq e_c \leq l_{e_l} + 1$.

## Subtasks

1. (5 points) $N \leq 2$
2. (14 points) $N \leq 1\,000$, $l_i \leq 5\,000$ (for each $i$ such that $1 \leq i \leq N$)
3. (26 points) $N \leq 1\,000$
4. (11 points) $l_i = l_j$ (for each $i, j$ such that $1 \leq i, j \leq N - 1$)
5. (44 points) *no additional constraints*