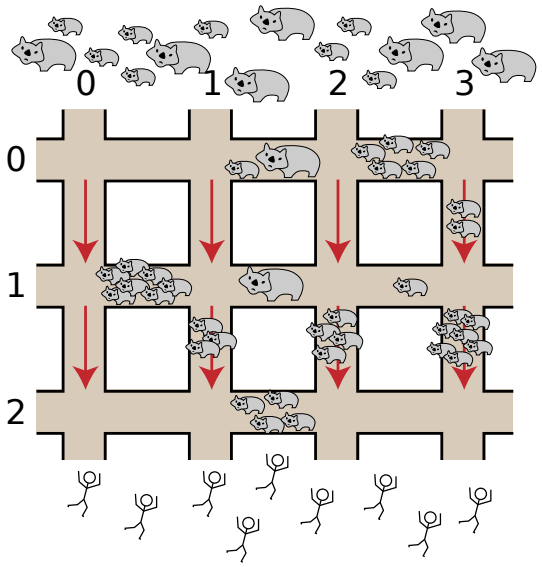
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia

wombats

English — 1.1

The city of Brisbane has been taken over by large mutated wombats, and you must lead the people to safety.

The roads in Brisbane are laid out in a large grid. There are R horizontal roads that run east-to-west, numbered $0, \ldots, (R - 1)$ in order from north to south, and C vertical roads that run north-to-south, numbered $0, \ldots, (C - 1)$ in order from west to east, as shown in the picture below.



The wombats have invaded from the north, and the people are escaping to the south. People can run along horizontal roads in either direction, but on vertical roads they will *only run towards the south*, towards safety.

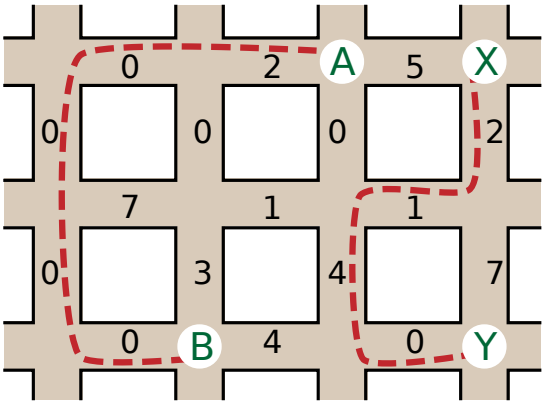
The intersection of horizontal road P with vertical road Q is denoted (P, Q). Each segment of road between two intersections contains some number of wombats, and these numbers may change over time. Your task is to guide each person from some given intersection in the north (on horizontal road 0) to some given intersection in the south (on horizontal road $R - 1$), taking them on a route that passes as few wombats as possible.

To begin, you will be given the size of the grid and the number of wombats on each road segment. Following this you will be given a series of E events, each of which is either:

- a *change*, which alters the number of wombats on some road segment; or
- an *escape*, where some person arrives at a given intersection on horizontal road `0`, and you must find a route to a given intersection on horizontal road `R - 1` that passes the fewest possible wombats.

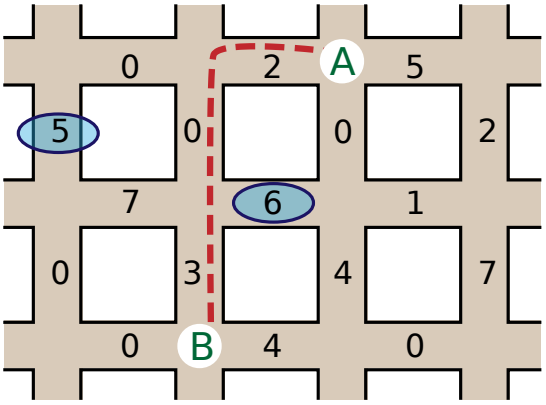
You must handle these events by implementing the routines `init()`, `changeH()`, `changeV()` and `escape()`, as described below.

Examples



The picture above shows an initial map with `R = 3` horizontal roads and `C = 4` vertical roads, with the number of wombats marked on each segment. Consider the following series of events:

- A person arrives at intersection `A = (0, 2)` and wishes to escape to intersection `B = (2, 1)`. The smallest number of wombats she can pass is `2`, as indicated by a dashed line.
- Another person arrives at intersection `X = (0, 3)` and wishes to escape to intersection `Y = (2, 3)`. The smallest number of wombats he can pass is `7`, again indicated by a dashed line.
- Two change events occur: the number of wombats on the top segment of vertical road `0` changes to `5`, and the number of wombats on the middle segment of horizontal road `1` changes to `6`. See the circled numbers in the picture below.



- A third person arrives at intersection `A = (0, 2)` and wishes to escape to intersection `B = (2, 1)`. Now the smallest number of wombats she can pass is `5`, as indicated by the new dashed line.

Implementation

You should submit a file implementing the procedures `init()`, `changeH()` and `changeV()` and the function `escape()`, as follows:

Your Procedure: `init()`

C/C++

```
void init(int R, int C, int H[5000][200], int V[5000][200]);
```

Pascal

```
type wombatsArrayType = array[0..4999, 0..199] of LongInt;

procedure init(R, C : LongInt; var H, V : wombatsArrayType);
```

Description

This procedure gives you the initial layout of the map, and allows you to initialise any global variables and data structures. It will be called only once, before any calls to `changeH()`, `changeV()` or `escape()`.

Parameters

- `R`: The number of horizontal roads.
- `C`: The number of vertical roads.
- `H`: A two-dimensional array of size `R × (C - 1)`, where `H[P][Q]` gives the number of wombats on the segment of horizontal road between intersections `(P, Q)` and `(P, Q + 1)`.

- `V`: A two-dimensional array of size `(R - 1) × C`, where `V[P][Q]` gives the number of wombats on the segment of vertical road between intersections `(P, Q)` and `(P + 1, Q)`.

Your Procedure: `changeH()`

C/C++
```
void changeH(int P, int Q, int W);
```

Pascal
```
procedure changeH(P, Q, W: LongInt);
```

Description

This procedure will be called when the number of wombats changes on the horizontal road segment between intersections `(P, Q)` and `(P, Q + 1)`.

Parameters

- `P`: Indicates which horizontal road is affected (`0 ≤ P ≤ R - 1`).
- `Q`: Indicates between which two vertical roads the segment lies (`0 ≤ Q ≤ C - 2`).
- `W`: The new number of wombats on this road segment (`0 ≤ W ≤ 1,000`).

Your Procedure: `changeV()`

C/C++
```
void changeV(int P, int Q, int W);
```

Pascal
```
procedure changeV(P, Q, W: LongInt);
```

Description

This procedure will be called when the number of wombats changes on the vertical road segment between intersections `(P, Q)` and `(P + 1, Q)`.

Parameters

- `P`: Indicates between which two horizontal roads the segment lies (`0 ≤ P ≤ R - 2`).
- `Q`: Indicates which vertical road is affected (`0 ≤ Q ≤ C - 1`).
- `W`: The new number of wombats on this road segment (`0 ≤ W ≤ 1,000`).

Your Function: `escape()`

C/C++
```
int escape(int V1, int V2);
```

Pascal
```
function escape(V1, V2 : LongInt) : LongInt;
```

Description

This function should calculate the fewest possible wombats a person must pass when travelling from intersection $(0, V1)$ to $(R-1, V2)$.

Parameters

- `V1`: Indicates where the person begins on horizontal row 0 ($0 \le V1 \le C-1$).
- `V2`: Indicates where the person ends on horizontal row $R-1$ ($0 \le V2 \le C-1$).
- *Returns*: The smallest number of wombats the person must pass.

Sample Session

The following session describes the example above:

Function Call	Returns
`init(3, 4, [[0,2,5], [7,1,1], [0,4,0]], [[0,0,0,2], [0,3,4,7]])`	
`escape(2,1)`	`2`
`escape(3,3)`	`7`
`changeV(0,0,5)`	
`changeH(1,1,6)`	
`escape(2,1)`	`5`

Constraints

- Time limit: 20 seconds
- Memory limit: 256 MiB
- $2 \le R \le 5{,}000$
- $1 \le C \le 200$
- At most 500 changes (calls to either `changeH()` or `changeV()`)

- At most 200,000 calls to `escape()`
- At most 1,000 wombats on any segment at any time

Subtasks

Subtask	Points	Additional Input Constraints
1	9	$C = 1$
2	12	$R,C \le 20$, and there will be no calls to `changeH()` or `changeV()`
3	16	$R,C \le 100$, and there will be at most 100 calls to `escape()`
4	18	$C = 2$
5	21	$C \le 100$
6	24	*(None)*

Experimentation

The sample grader on your computer will read input from the file `wombats.in`, which must be in the following format:

- line 1: `R C`
- line 2: `H[0][0] … H[0][C-2]`
- …
- line $(R + 1)$: `H[R-1][0] … H[R-1][C-2]`
- line $(R + 2)$: `V[0][0] … V[0][C-1]`
- …
- line $(2R)$: `V[R-2][0] … V[R-2][C-1]`
- next line: `E`
- next E lines: one event per line, in the order in which events occur

If $C = 1$, the empty lines containing the number of wombats on horizontal roads (lines 2 through $R + 1$) are not necessary.

The line for each event must be in one of the following formats:

- to indicate `changeH(P, Q, W)`: `1 P Q W`
- to indicate `changeV(P, Q, W)`: `2 P Q W`
- to indicate `escape(V1, V2)`: `3 V1 V2`

For instance, the example above should be provided in the following format:

```
3 4
0 2 5
7 1 1
0 4 0
0 0 0 2
0 3 4 7
5
3 2 1
3 3 3
2 0 0 5
1 1 1 6
3 2 1
```

Language Notes

C/C++	You must `#include "wombats.h"`.
Pascal	You must define the `unit Wombats`. All arrays are numbered beginning at `0` (not `1`).

See the solution templates on your machine for examples.