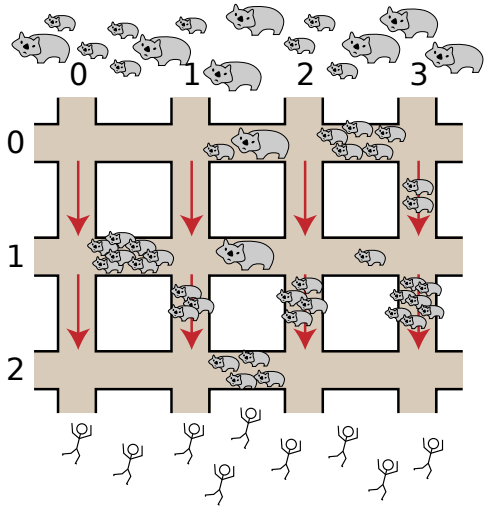
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia

wombats

French — 1.1

La ville de Brisbane a été prise d'assaut par de gros wombats mutants et vous devez guider les habitants hors du danger.

Les rues de Brisbane forment une grande grille. Il y a `R` rues horizontales qui vont d'est en ouest, numérotées de `0` à `R-1` dans l'ordre du nord au sud et `C` rues verticales qui vont du nord au sud, numérotées de `0` à `C-1` dans l'ordre d'ouest en est, comme indiqué sur l'illustration ci-dessous.



Les wombats envahissent la ville en venant du nord et la population s'échappe vers le sud, où elle sera en sécurité. Les habitants peuvent courir le long des rues horizontales dans les 2 directions, mais ils ne peuvent aller *que vers le sud* lorsqu'ils se trouvent sur des rues verticales.

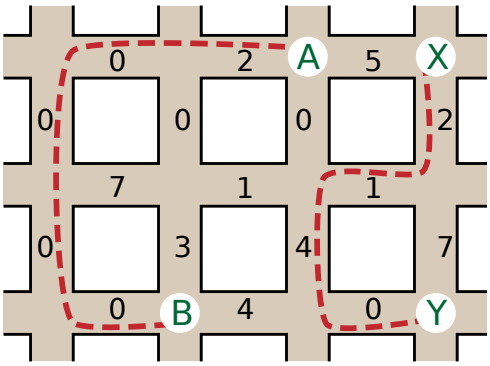
L'intersection d'une rue horizontale `P` avec une rue verticale `Q` est notée `(P, Q)`. Chaque segment de rue entre deux intersections contient un certain nombre de wombats, et ces nombres peuvent changer au fil du temps. Votre travail est de guider chaque personne depuis une certaine intersection au nord (sur la rue horizontale `0`) vers une certaine intersection au sud (sur la rue horizontale `R-1`), en les faisant parcourir un trajet qui rencontre le moins de wombats possible.

Pour commencer, on vous donnera la taille de la grille et le nombre de wombats de chaque segment de rue. On vous donnera ensuite une série de `E` événements, chacun pouvant être :

- un *changement* (change), qui modifie le nombre de wombats sur un segment de rue, ou
- une *évasion* (escape), où une personne arrive à une certaine intersection sur la route horizontale `0` pour laquelle vous devez trouver un trajet vers une intersection donnée sur la rue horizontale `R-1`, qui passe par le moins de wombats possible.

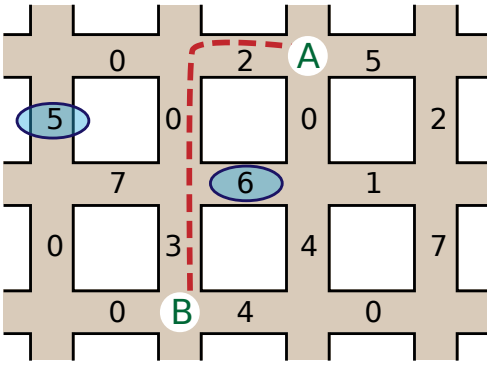
Vous devez traiter ces événements en implémentant les fonctions `init()`, `changeH()`, `changeV()` et `escape()`, comme décrit ci-dessous.

Exemples



L'illustration ci-dessus montre la carte à l'état initial. Il contient $R = 3$ rues horizontales et $C = 4$ rues verticales. Le nombre de wombats est indiqué sur chaque segment. Considérez la série d'événements suivante :

- Une personne arrive à l'intersection $A = (0, 2)$ et souhaite s'échapper vers l'intersection $B = (2, 1)$. Le plus petit nombre de wombats qu'elle peut rencontrer est 2, comme indiqué par une ligne en pointillé.
- Une autre personne arrive à l'intersection $X = (0, 3)$ et souhaite s'échapper vers l'intersection $Y = (2, 3)$. Le plus petit nombre de wombats qu'elle peut rencontrer est 7, comme indiqué là aussi par une ligne en pointillé.
- Deux changements se produisent : le nombre de wombats sur le segment du haut de la rue verticale 0 passe à 5, et le nombre de wombats du segment du milieu de la rue horizontale 1 passe à 6. Ces changements sont représentés par des nombres encerclés sur l'illustration qui suit.



- Une troisième personne arrive à l'intersection $A = (0, 2)$ et souhaite s'échapper vers l'intersection $B = (2, 1)$. Le plus petit nombre de wombats qu'elle peut désormais rencontrer est 5, comme indiqué par la nouvelle ligne en pointillé.

Implémentation

Vous devez soumettre un fichier qui implémente les fonctions `init()`, `changeH()`, `changeV()` et `escape()`, comme suit :

Votre fonction : `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

Cette fonction vous donne la structure initiale de la carte et vous permet d'initialiser toutes les variables globales et structures de données. Elle ne sera appelée qu'une fois, avant tout appel aux fonctions `changeH()`, `changeV()` ou `escape()`.

Paramètres

- `R` : le nombre de rues horizontales.
- `C` : le nombre de rues verticales.
- `H` : un tableau à deux dimensions de taille `R × (C - 1)`, où `H[P][Q]` donne le nombre de wombats du segment de rue horizontal entre les intersections `(P, Q)` et `(P, Q + 1)`.
- `V` : un tableau à deux dimensions de taille `(R - 1) × C`, où `V[P][Q]` donne le nombre de wombats du segment de rue vertical entre les intersections `(P, Q)` et `(P + 1, Q)`.

Votre fonction : `changeH()`

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

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

Description

Cette fonction sera appelée lorsque le nombre de wombats change sur le segment de rue horizontal entre les intersections (P, Q) et $(P, Q + 1)$.

Paramètres

- `P` : indique quelle rue horizontale est affectée ($0 \le P \le R - 1$).
- `Q` : indique entre quelles rues verticales le segment se trouve ($0 \le Q \le C - 2$).
- `W` : le nouveau nombre de wombats sur ce segment de rue ($0 \le W \le 1\,000$).

Votre fonction : `changeV()`

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

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

Description

Cette fonction sera appelée lorsque le nombre de wombats change sur le segment de rue vertical entre les intersections (P, Q) et $(P + 1, Q)$.

Paramètres

- `P` : indique entre quelles rues horizontales le segment se trouve ($0 \le P \le R - 2$).
- `Q` : indique quelle rue verticale est affectée ($0 \le Q \le C - 1$).
- `W` : le nouveau nombre de wombats sur ce segment de rue ($0 \le W \le 1\,000$).

Votre fonction : `escape()`

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

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

Description

Cette fonction doit calculer le plus petit nombre de wombats qu'une personne doit rencontrer pour aller de l'intersection $(0, V1)$ à l'intersection $(R-1, V2)$.

Paramètres

- `V1` : indique où la personne commence sur la rue horizontale 0 ($0 \le V1 \le C-1$).
- `V2` : indique où la personne se retrouve sur la rue horizontale $R-1$ ($0 \le V2 \le C-1$).
- *Retourne* : le plus petit nombre de wombats que la personne doit rencontrer.

Exemple de session

La session suivante décrit l'exemple ci-dessus :

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`

Contraintes

- Limite de temps : 20 secondes
- Limite de mémoire : 256 Mio
- $2 \le R \le 5\,000$
- $1 \le C \le 200$
- Au maximum 500 changements (appels soit à `changeH()` soit à `changeV()`)
- Au maximum 200 000 appels à `escape()`
- Au maximum 1 000 wombats sur un segment à un moment donné

Sous-tâches

Sous-tâche	Points	Contraintes d'entrée supplémentaires
1	9	$C = 1$
2	12	$R, C \leq 20$ et il n'y aura aucun appel à `changeH()` ou `changeV()`
3	16	$R, C \leq 100$, et il y aura au plus 100 appels à `escape()`
4	18	$C = 2$
5	21	$C \leq 100$
6	24	*(Aucune)*

Expérimentation

L'évaluateur fourni sur votre ordinateur lira l'entrée dans le fichier `wombats.in`, qui doit être au format suivant :

- ligne 1 : `R C`
- ligne 2 : `H[0][0] … H[0][C-2]`
- …
- ligne $R + 1$: `H[R-1][0] … H[R-1][C-2]`
- ligne $R + 2$: `V[0][0] … V[0][C-1]`
- …
- ligne $2R$: `V[R-2][0] … V[R-2][C-1]`
- ligne suivante : `E`
- les E lignes suivantes : un événement par ligne, dans l'ordre dans lequel les événements se produisent

Si $C = 1$, les lignes vides contenant le nombre de wombats sur les routes horizontales (lignes 2 à $R + 1$) ne sont pas nécessaires.

La ligne de chaque événement doit être dans l'un des formats suivants :

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

Par exemple, l'exemple ci-dessus doit être fourni au format suivant :

```
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
```

Remarques pour chaque langage

C/C++	Vous devez utiliser `#include "wombats.h"`.
Pascal	Vous devez utiliser l'unité `unit Wombats`. Tous les tableaux sont numérotés à partir de `0` (et non `1`).

Prenez exemple sur les modèles de solution fournis sur votre machine.