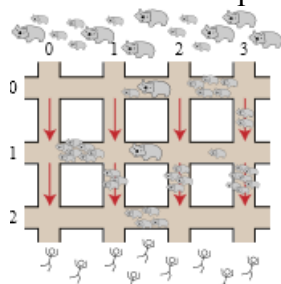
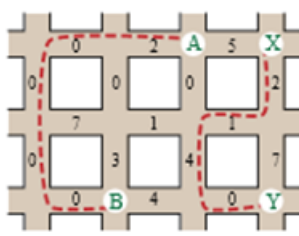


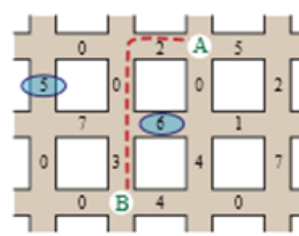
Бризбейн е нападен от огромни мутиралаи уомбати (двуутробни животни, подобни на мечета) и трябва да изведе жителите на града на безопасно място. Улиците на града образуват голяма мрежа от  $R$  хоризонтални улици, водещи от изток на запад, номерирани с  $0, 1, \dots, (R - 1)$ , започвайки от най-северната и  $C$  вертикални улици, водещи от север на юг, номерирани с  $0, 1, \dots, (C - 1)$ , започвайки от най-западната, както е показано на Фиг. 1. Уомбатите нападат от север, а гражданите бягат от тях на юг, т.е. могат да се движат по хоризонталните улици както поискат но по вертикалните *само в южна посока*, където е спасението.



Фиг. 1



Фиг. 2



Фиг. 3

Кръстовището, на което се пресичат хоризонталната улица  $P$  и вертикалната  $Q$ , означаваме с  $(P, Q)$ . На всяка отсечка между две съседни кръстовища се намират определен брой уомбати, който може да се променя. Задачата е, да се преведе гражданин, намиращ се на някое от кръстовищата на най-северната хоризонтална улица (номер 0) до някое кръстовище на най-южната хоризонтална улица (номер  $R - 1$ ) по маршрут, по който ще се срещнат минимален брой уомбати.

В началото се задават броят на улиците на мрежата и броят на уомбатите във всяка отсечка. След това – редица от  $E$  събития, които могат да бъдат:

- *смяна*, при която се променя броят на уомбатите на някоя от отсечките, или
- *извеждане*, при което гражданин се появява на някое от кръстовищата на най-северната улица и трябва да бъде изведен до кръстовище на най-южната улица.

За да обработите последователност от такива събития ще трябва да напишете подпрограмите `init()`, `changeH()`, `changeV()` и `escape()` специфицирани по-нататък.

### Пример

Нека  $R=3$ ,  $C=4$ , а броят на уомбатите за всяка отсечка е изписан върху нея. Да разгледаме следната последователност от събития:

- гражданин се появява в кръстовището  $A=(0,2)$  и трябва да бъде изведен до кръстовището  $B=(2,1)$ . Най-малкият брой уомбати в този случай е 2, по маршрута показан на Фиг. 2 с пунктир;
- друг гражданин се появява в кръстовището  $X=(0,3)$  и трябва да бъде изведен до кръстовището  $Y=(2,3)$ . Най-малкият брой уомбати в този случай е седем, по маршрута показан на Фиг. 2;
- броят на уомбатите на най-северната отсечка на вертикалния път 0 се променя от 0 на 5, а на уомбатите на средната отсечка на хоризонталния път 1 се променя от 1 на 6 (числата, показани в елипси на фигурата);

- трети гражданин се появява на кръстовището  $A=(0,2)$  и трябва да бъде изведен до кръстовището  $B=(2,1)$ . Най-малкият брой уомбати в този случай е 5, по маршрута показан на фигурата с пунктир.

За оценяването изпратете файл с функциите `init()`, `changeH()`, `changeV()` и `escape()` както следва:

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

Задава началното положение на мрежата и дава възможност да инициализирате глобални променливи и структури от данни. Ще бъде извикана само веднъж, преди всяко друго извикване.  $R$  е броят на хоризонталните, а  $C$  – на вертикалните улици.  $H$  е двумерен масив с размери  $R \times (C-1)$ , като  $H[P][Q]$  съдържа броя на уомбатите в отсечката от  $(P, Q)$  до  $(P, Q+1)$ .  $V$  е двумерен масив с размери  $(R-1) \times C$ , като  $V[P][Q]$  съдържа броя на уомбатите в отсечката от  $(P, Q)$  до  $(P+1, Q)$ .

```
void changeH(int P, int Q, int W)
```

Ще бъде извикана, когато броят на уомбатите в отсечката  $(P, Q)$  до  $(P, Q+1)$  стане  $W$ ,  $0 \leq P \leq R-1$ ,  $0 \leq Q \leq C-2$ ,  $0 \leq W \leq 1000$ .

```
void changeV(int P, int Q, int W)
```

Ще бъде извикана, когато броят на уомбатите в отсечката  $(P, Q)$  до  $(P+1, Q)$  стане  $W$ ,  $0 \leq P \leq R-2$ ,  $0 \leq Q \leq C-1$ ,  $0 \leq W \leq 1000$ .

```
int escape(int V1, int V2)
```

Ще бъде извикана да намери маршрут от  $(0, V1)$  до  $(R-1, V2)$  по който има най-малко уомбати,  $0 \leq V1 \leq C-1$ ,  $0 \leq V2 \leq C-1$ . Функцията връща броя на уомбатите по намерения маршрут

### Примерна сесия

Следната последователност от извиквания съответства на примера даден по-горе:

Извикване	Връща
<code>init(3, 4, [[0, 2, 5], [7, 1, 1], [0, 4, 0]], [[0, 0, 0, 2], [0, 3, 4, 7]])</code>	
<code>escape(2, 1)</code>	2
<code>escape(3, 3)</code>	7
<code>changeV(0, 0, 5)</code>	
<code>changeH(1, 1, 6)</code>	
<code>escape(2, 1)</code>	5

### Ограничения

- Ограничение за време: 15 сек.
- Ограничения за памет: 256 MiB
- $2 \leq R \leq 5,000$
- $1 \leq C \leq 200$
- Най-много 500 извиквания на `changeH()` или `changeV()`
- Най-много 200,000 извиквания на `escape()`
- Най-много 1,000 уомбата на всяка отсечка по всяко време

---

## Подзадачи

---

### Подзадача Точки

### Допълнителни ограничения

1	9	$C = 1$
2	12	$R, C \leq 20$ и няма извиквания на <code>changeH()</code> или <code>changeV()</code>
3	16	$R, C \leq 100$ и най-много 100 извиквания на <code>escape()</code>
4	18	$C = 2$
5	21	$C \leq 100$
6	24	Няма ограничения

---

## Експерименти

Упростен грейдер, намиращ се на Вашия компютър, чете вход от файла с име `wombats.in`, със следния формат:

- Ред 1:  $R \ C$
- Ред 2:  $H[0][0] \dots H[0][C-2]$
- ...
- Ред  $(R + 1)$ :  $H[R-1][0] \dots H[R-1][C-2]$
- Ред  $(R + 2)$ :  $V[0][0] \dots V[0][C-1]$
- ...
- line  $(2R)$ :  $V[R-2][0] \dots V[R-2][C-1]$
- следващ ред:  $E$
- следват  $E$  реда с по едно събитие на ред, в последователността в която са се случили.

Ако  $C = 1$ , не е необходимо да се добавят празни линии за броя на уомбатите в хоризонталните улици (редове от 2 до  $R + 1$ ).

Събитията се задават както следва:

- извикването `changeH(P, Q, W)` с 1  $P \ Q \ W$
- извикването `changeV(P, Q, W)` с 2  $P \ Q \ W$
- извикването `escape(V1, V2)` с 3  $V1 \ V2$

За да изпратите за експеримент дадения по-горе пример трябва да създадете следния файл

```
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
```

## Забележки

C/C++    Добавете във файла с вашите функции `#include "wombats.h"`.

Примерен файл с решение ще намерите на вашия компютър.