# Упражнения: Многомерни масиви

Тествайте решението в Judge: <https://judge.softuni.bg/Contests/3173/Multidimentional-Arrays>

## Сума на елементите от матрица

Напишете програма, която чете от входа матрица и отпечатва:

* Броя на **редиците**
* Броя на **колоните**
* Сумите на всички **елементи от матрицата**

На първия ред ще получите размера на матрицата в следния формат: **“ред, колона”**. На следващите редове ще прочетете клетките на матрицата.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3, 6  7, 1, 3, 3, 2, 1 1, 3, 9, 8, 5, 6 4, 6, 7, 9, 1, 0 | 3  6  76 |

### Насоки

* Опитайте се да използвате **foreach-цикъл**

## Сума на колоните на матрица

Напишете програма, която **чете матрица** от входа на конзолата и отпечатва сумата на всички колони. На първия ред ще получите размерите на матрицата в следния формат: **"редици , колони",** разделени с запетая и интервал. На следващите **редове** ще получите елементите за всяка колона разделени с интервал.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3, 6  7 1 3 3 2 1  1 3 9 8 5 6  4 6 7 9 1 0 | 12  10  19  20  8  7 |
| 3, 3  1 2 3  4 5 6  7 8 9 | 12  15  18 |

### Насоки

* Прочете размера на матрицата.
* На следващите редове ще получите колоните.
* Обходете през матрицата и съберете сумата на всички елементи във всяка колона.
* Отпечатайте сумата и продължете с другите колони.

## **Квадрат с най-голяма сума**

Напишете програма, която **чете матрица** от конзолата. Трябва да намерите сумата от квадрат с **дължина 2x2** от матрицата и да го отпечатате.

На първия ред ще получите размерите в следния формат **"редици, колони".**

На следващите **редове** ще получите елементите за всяка **колона** разделени с запетая и интервал .

Отпечатайте **най-големия** квадрат и неговата сума.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| 3, 6  7, 1, 3, 3, 2, 1 1, 3, 9, 8, 5, 6 4, 6, 7, 9, 1, 0 | 9 8  7 9  33 | 7, 1, 3, 3, 2, 1 1, 3, **9, 8**, 5, 6 4, 6, **7, 9**, 1, 0 |
| 2, 4  10, 11, 12, 13  14, 15, 16, 17 | 12 13  16 17  58 | 10, 11, **12, 13**  14, 15, **16, 17** |

### Насоки

* Помислете за **IndexOutOfRangeException()**
* Ако намерете повече от един квадрата с най-голяма сума, отпечатайте този в най-горе ляво

## **Модификация на назъбен масив**

Напишете програма, която **чете матрица** от конзолата и чете команди. На първия ред ще получите **редиците**. На следващите **редове** ще получите елементите за всяка **колона** разделени с **интервал**. Командите ще бъдат в следния формат:

* **Add {редица} {колона} {стойност}** – **Увеличава** числото на дадените **координати** със **стойността.**
* **Subtract {редица} {колона} {стойност}** – **Намалява** числото на дадените **координати** със **стойността.**

Координатите може и да не бъдат валидни. В такъв случай отпечатайте "**Invalid coordinates**". Когато получите "**END**", програмата спира и трябва да отпечатате матрицата.

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 3  **1** 2 3  4 5 **6**  7 8 9  Add 0 0 5  Subtract 1 2 1  END | **6** 2 3  4 **5** **5**  7 8 9 |
| 4  1 2 3 4  5 6 7 8  8 7 6 5  4 3 2 1  Add 4 4 100  Add 3 3 100  Subtract -1 -1 42  Subtract 0 0 42  END | Invalid coordinates  Invalid coordinates  -41 2 3 4  5 6 7 8  8 7 6 5  4 3 2 101 |

## **Триъгълника на Паскал**

Триъгълника може да бъде създаден по следния начин: В **редица 0** (най- горната) има **уникално число** със стойност 1. Всяка стойност от всяка следваща редица се създава чрез **добавяне** на числото отгоре отляво с числото отгоре отдясно, третирайки празните места с 0. Например първоначалното число в първата (или която и да е друга) редица е 1 (сумата от 0 и 1), докато числата 1 и 3 в третата редица се добавят, за да се получи числото 4 в четвъртата редица.

За повече информация научете на: <https://en.wikipedia.org/wiki/Pascal's_triangle>

Отпечатайте **елементите** на всяка редица разделени с **интервал.**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 4 | 1  1 1  1 2 1  1 3 3 1 |
| 13 | 1  1 1  1 2 1  1 3 3 1  1 4 6 4 1  1 5 10 10 5 1  1 6 15 20 15 6 1  1 7 21 35 35 21 7 1  1 8 28 56 70 56 28 8 1  1 9 36 84 126 126 84 36 9 1  1 10 45 120 210 252 210 120 45 10 1  1 11 55 165 330 462 462 330 165 55 11 1  1 12 66 220 495 792 924 792 495 220 66 12 1 |

### Насоки

* Входа **n** винаги ще бъде: **1 <= n <= 60**
* Помислете от какъв **тип данни** ще бъде масива
* Не се страхувайте да **използвате повече от един масив**

## 2X2 Квадрат

Намерете броя на квадратите от **еднакви букви** с дължина **2x2**.

### Вход

* На **първия ред** ще ви бъде даден броя на **редиците** и броя на **колоните** - измеренията на матриците
* На следващите редове ще ви бъдат дадени **редиците** на матриците. Броя на редовете ще бъде равен на броя на редиците

### Изход

* Трябва да отпечатате **броя** на еднаквите квадрати от матрицата.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| 3 4  A B B D  E B B B  I J B B | 2 | Две 2x2 квадрата с еднакви клетки:  A **B B** D A B B D  E **B B** B E B **B B**  I J B B I J **B B** |
| 2 2  a b  c d | 0 | Няма съществуващи 2x2 квадрата с еднакви клетки. |

## Максимална сума 3X3

Напишете програма, която чете правоъгълна матрица с размери **N** и **M**. Трябва да намерите **квадрата** с най-голяма сума, с **размери 3x3**.

### Вход

* На първия ред ще получите **N** редици и **M** колони
* На следващите **N** реда ще получите **редиците**

### Изход

* Отпечатайте **елементите** на квадрата от матрицата с дължина 3x3 и тяхната **сума**.

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Матрица** | **Изход** |
| 4 5  1 5 5 2 4  2 1 4 14 3  3 7 11 2 8  4 8 12 16 4 |  | Sum = 75  1 4 14  7 11 2  8 12 16 |

## Змия

Разхождате се в парка. Срещу вас стой змия. Уплашени започвате да бягате на зиг-заг. Змията започва да ви следва.

Вашата задача е да визуализирате пътя на змията в квадратна форма. **Змията** е представена като **низ**. Движи се по матрица с дължина **NxM.** Змията започва от **горният ляв ъгъл** като се плъзга надолу. Първата клетка е запълнена с първия символ на змията, втората клетка е запълнена с втория символ на змията и т.н. Змията е толкова дълга, **колкото е необходимо**, за да запълни редата изцяло – ако стигнете до края на низа, змията, започва отново от началото. След като попълните матрицата с пътя на змията, трябва да я отпечатате.

### Вход

* На първия ред ще получите измеренията в следния формат: **"N M"**, където N са редиците, а M са колоните. Ще ги получите разделени с интервал
* На втория ред ще получите низ - **змията**

### Изход

* Изхода трябва да отпечата на конзолата **N реда.**
* Всеки ред трябва да съдържа низ, който представлява съответния ред от матрицата

### Бележки

* **Измеренията** N и M ще бъдат в обхвата [1 … 12]
* **Змията** ще бъде низ с дължина [1 … 20] и **няма да съдържа интервал**

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 5 6  SoftUni | SoftUn  UtfoSi  niSoft  foSinU  tUniSo |

## Манипулиране на назъбен масив

Създайте програма, която манипулира елементите от матрица с различни дължини на редиците.

Първо прочетете **числото N**, което е равно на **номера на редиците** в матрицата.

На **следващите N редове** ще ви бъде **дадени редици от числа (разделена с интервал) на матрицата**.

След попълването на матрицата, започнете да я анализирате. Ако сегашната редицата и редицата под нея имат еднаква дължина, **умножете всяко** число по 2. В **противен** случай ги **разделете** на 2.

След това ще получите командите. Има три възможни команди:

* "**Add {ред} {колона} {стойност}**" - Добавя **{стойност}** на елемента на дадени индекси, ако индексите са валидни
* "**Subtract {ред} {колона} {стойност}**" - Изважда **{стойност}** от елемента на дадени индекси, ако са валидни
* "**End**" - принтира крайната матрица (всички елементи са разделени с интервал) и спира програмата

### Вход

* На първия ред ще получите **броя на редиците** на матрицата - число **N**
* На следващите **N редове** ще получите **редиците на матрицата** - редици от числа разделени с интервал
* Ще получавате команди докато не получите **"End"**

### Изход

* Изходът трябва да бъде принтиран на конзолата. Трябва да съдържа **N реда**
* Всеки ред трябва да съдържа низ - **редица** на матрицата. Елементите на всяка редица трябва да бъдат разделени с **интервал**

### Бележки

* **Броя** на редиците е число в обхвата на [2 … 12]
* **Входа** винаги ще бъде във **формата** по-горе
* **Помислете** какъв тип данни ще използвате

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 5  10 20 30  1 2 3  2  2  10 10  End | 20 40 60 1 2 3 2 2 5 5 |
| 5  10 20 30  1 2 3  2  2  10 10  Add 0 10 10  Add 0 0 10  Subtract -3 0 10  Subtract 3 0 10  End | 30 40 60  1 2 3  2  -8  5 5 |

## Играта на коня

Шахът е най-старата игра, но все още е популярна. В задачата ще използваме само една фигура - **Конят.**

Конят се движи до **най-близкото** поле, **но не на същия ред, колона** или **диагонал**. (Има два вида движение. Първото е да се премести две хоризонтално и след това едно квадратче вертикално. Другото е едно квадратче хоризонтално и след това две квадратчета вертикално - т. е. в модел **"L"**.)

Играта на коня се играе на дъска с измерение NxN и много коне: **0 <= K <= N2**.

Ще получите дъска със символи. **'K'** означава, че на това място име кон. '**0'** означава е клетката е празна. Вашата задача е да премахнете минималния брой коне, така че да няма останали коне за атака на другия кон.

### Вход

* На първия ред ще получите числото **N** - размера на дъската.
* На следващите N реда ще получите низ от **'K'** или **'0'**.

### Изход

Принтирайте минималния брой коне, който трябва да се премахнат

### Бележки

* Размерът на дъската ще бъде от 0 до 30

### Примери

|  |  |
| --- | --- |
| **Вход** | **Изход** |
| 5  0K0K0  K000K  00K00  K000K  0K0K0 | 1 |
| 2  KK  KK | 0 |
| 8  0K0KKK00  0K00KKKK  00K0000K  KKKKKK0K  K0K0000K  KK00000K  00K0K000  000K00KK | 12 |

## \*Бомба

Ще ви бъде дадена квадратна матрица от числа. Всяко число ще бъде разделено **с интервал**, а всяка редица на нов ред. На последният ред ще получите индекси- координати на няколко клетки разделени **с интервал** в следният формат: **редица1,колана1 редица2,колона2 редица3,колона3...**

На тези клетки има бомби. Трябва да минете през **всички бомби** както са дадени в подред. Когато бомбата експлодира нанася щети **равни на стойността** на бомбата на всички клетки около нея(във всички посоки и диагонали). Една бомба не може да се взриви повече от един път. След експлозията получава стойност **0**. Когато клетката е **по-малка или равна** на 0, тя **умира**. Умрелите клетки не могат **експлодират**.

Вие трябва да отпечатате **броя на живите клетки** и **тяхната сума**. След това принтирайте всички клетки (включително умрелите).

### Вход

* На първия ред ще получите цяло число N - размерът на квадратната матрица
* На следващите N реда ще получите стойностите за всяка редица - N числа разделени с интервал
* На последния ред ще получите координати на клетките с бомби

### Изход

* На първия ред отпечатайте броя на оцелеете клетки в следния формат:

“**Alive cells: {aliveCells}**”

* На втория ред отпечатайте сумата от оцелелите числа

“**Sum: {sumOfCells}**”

* Накрая отпечатайте матрицата. Клетките трябва да бъдат **разделени с интервал**.

### Бележки

* Размерът на матрицата ще бъде в обхвата **[0…1000].**
* Координатите на бомбата **винаги** ще бъдат в матрицата.
* Стойностите на бомбата винаги ще са **по-големи от 0.**
* Числата в матрицата ще бъдат в обхвата **[1…10000].**

### Примери

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Коментари** |
| 4  8 3 2 5  6 4 7 9  9 9 3 6  6 8 1 2  1,2 2,1 2,0 | Alive cells: 3  Sum: 12  8 -4 -5 -2  -3 -3 0 2  0 0 -4 -1  -3 -1 -1 2 | Първо бомбата със стойност **7** ще експлодира и ще намали стойностите на клетките около нея. След това бомбата с координати **2,1** и стойност **2** (първоначално 9-7) ще експлодира и ще намали съседните си клетки. В крайна сметка бомбата с координати 2.0 и стойност **7** (първоначално 9-2) ще избухне. След това трябва да отпечатате броя на живите клетки, който е 3, а сумата им е 12. Отпечатайте матрицата след експлозиите. |
| 3  7 8 4  3 1 5  6 4 9  0,2 1,0 2,2 | Alive cells: 3  Sum: 8  4 1 0  0 -3 -8  3 -8 0 |  |

## \*Гадни зайци

Разглеждате си GitHub, откривате някаква стара js игра. Става въпрос за гадни зайци, който се размножават много бързо. Има и играч, който трябва да избягва от тяхната бърлога. Много ви допада тази игра и решавате да я направите на C#, защото това е езикът, който сте избрали. Алгоритъмът на играта трябва да реше дали играчът ще успее да избяга.

Първо ще трябва да получите числата **N** и **M**, който са колоните и редовете на бърлога. След това ще получете **N** низове, като съдържат символите **".", "B", "P".** Зайците са маркирани с **"B"**. **Играчът** е маркиран с **"P"**. **Всичко друго** е запълнено с '.'. От тези символи е изграден първоначалното състояние на бърлогата. Винаги ще има само един играч. След това ще получите **низ** с команди като **LLRRUUDD -** всеки символ означава на къде да се **премести** играча (L-наляво, R-надясно, U-нагоре, D-надолу).

**След** всяка стъпка, зайците ще опитват да го спрат като се разпространяват наляво, надясно, нагоре, надолу (съседната клетка от '.' се **променя** на 'B'). Ако играчът се **премести** на клетка, където има **заек** или някой заек **хване** играчът, то той умира. Ако **излезе** без да бъде хванат печели играта.

Когато играчът **спечели или умре**, играта свършва. Няма да има такива примери, в които играчът да няма ходове преди да умре или да избяга.

Накрая трябва да отпечатате всяка редица на нов ред. На последният ред, отпечатайте **“dead: {ред} {колона}”** или **“won: {ред} {колона}”.** Редът и колоната са координатите на клетката на играчът ако е умрял или последната клетка, на която е бил преди да избяга.

### Вход

* На първият ред ще получите числата **N** и **M** - номерата на **редиците** и **колоните** на бърлогата
* На следващите N реда ще получите редиците като низ. Низът съдържа само “.”, “B”, “P”. Всички низове ще имат еднаква големина. Ще има само един символ от “P”
* На последният ред ще получите посоките като низ, който съдържа “R”, “L”, “U”, “D”

### Изход

* На първите N реда отпечатайте крайният резултат от бърлогата
* На последния ред отпечатайте – “won:” or “dead:” + {row} {col}

### Бележки

* Директорията е низ в обхвата [1...20]

### Примери

|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 5 8  .......B  ...B....  ....B..B  ........  ..P.....  ULLL | BBBBBBBB  BBBBBBBB  BBBBBBBB  .BBBBBBB  ..BBBBBB  won: 3 0 | 4 5  .....  .....  .B...  ...P.  LLLLLLLL | .B...  BBB..  BBBB.  BBB..  dead: 3 1 |