ToPAS'24
Torneio de Programação para Alunos do Secundário

Departamento de Informática

https://eventos.fct.unl.pt/topas-lx

Conjunto de Problemas

Faculdade de Ciências e Tecnologia | NOVA FCT

10 de maio de 2024

Este conjunto de problemas deverá conter oito (8) problemas e dezoito (18) páginas.
Se faltar algum problema, por favor avise a organização.

Edição realizada em colaboração com:

Departamento de Ciência de Computadores, Faculdade de Ciências, Universidade do Porto

Departamento de Engenharia Eletrónica e Informática, Faculdade de Ciências e Tecnologia, Universidade do Algarve

ToPAS'24
Torneio de Programação para Alunos do Secundário

Departamento de Informática – Faculdade de Ciências e Tecnologia | NOVA FCT
10 de maio de 2024

Conteúdo

Problema A: Encontros e desencontros 3
Problema B: Dias até ao Natal 5
Problema C: Melhores amigos 7
Problema D: Não metas água! 9
Problema E: Vai um? 11
Problema F: Tudo em família! 13
Problema G: Festa da empresa 15
Problema H: Zega Zinga, Zoga Zonga 17

Encontros e desencontros

O Alex e a Bia são primos direitos que gostam de atividades ao ar livre. Ultimamente, o Alex vai jogar à bola com os amigos, numa zona do parque chamada *campo*, enquanto a Bia prefere dar um grande passeio de bicicleta, passando sempre pelo campo. Por isso, às vezes encontram-se.

Se souberes o período em que o Alex está no campo e o instante em que a Bia passa por lá de bicicleta, consegues dizer se os dois primos se encontram ou se se desencontram?

Por exemplo, se o Alex estiver no campo das 15:10 às 17:15 e a Bia passar por lá às 15:20, os dois primos encontram-se. Mas, se a Bia passar pela zona onde o primo joga à bola às 15:08, já não se encontram.

Designed by brgfx / Freepik

O período em que o Alex está no campo é definido por dois instantes no mesmo dia: o instante em que chega ao campo e o instante em que sai do campo. É óbvio que a saída nunca ocorre antes da chegada. Os primos *encontram-se* se a Bia passar pelo campo enquanto o Alex lá está, considerando-se que ele está no campo nos instantes em que chega e em que sai.

Tarefa

Escreva um programa que, dados o período em que o Alex está no campo e o instante em que a Bia passa por lá, indica se os dois primos se encontram.

Input

O input tem três linhas, cada uma com dois números inteiros, h e m, que representam o instante h:m (h horas e m minutos). A primeira linha tem o instante em que o Alex chega ao campo. A segunda linha tem o instante em que o Alex sai do campo (que é igual ou posterior ao instante em que o Alex chega ao campo). A terceira linha tem o instante em que a Bia passa pelo campo.

Restrições

$0 \leq h \leq 23$ Horas de um instante

$0 \leq m \leq 59$ Minutos de um instante

Output

O output tem uma única linha, com: "`Encontram-se`", se os dois primos se encontram; "`Desencontram-se`", nos restantes casos.

Exemplo 1

Input

```
15 10
17 15
15 20
```

Output

```
Encontram-se
```

Exemplo 2

Input

```
15 10
17 15
15 8
```

Output

```
Desencontram-se
```

Exemplo 3

Input

```
9 7
9 7
9 7
```

Output

```
Encontram-se
```

Exemplo 4

Input

```
14 0
17 59
18 5
```

Output

```
Desencontram-se
```

Dias até ao Natal

O Artur é um menino que sonha com o dia de Natal durante todo o ano. Ele imagina a casa repleta de risos e conversas animadas, o cheiro de bacalhau cozido pairando no ar e os sons suaves de músicas natalícias ecoando pela sala. Mas o que ele mais quer é ver os presentes debaixo da árvore para antecipar a sensação de felicidade de os desembrulhar.

Consegues ajudar o Artur escrevendo um programa que calcula quantos dias faltam até ao dia 25 de dezembro?

Tarefa

Escreva um programa que lê uma data, representada pelo dia e pelo mês do ano, e calcula o número de dias que faltam até ao próximo 25 de dezembro.

Vamos assumir que os anos em causa **não são bissextos**, isto é, que o mês de fevereiro tem 28 dias. Logo, o número de dias de cada mês é dado pela seguinte tabela:

Mês	1	2	3	4	5	6	7	8	9	10	11	12
Número de dias	31	28	31	30	31	30	31	31	30	31	30	31

Note que, se a data for depois de 25 de dezembro, o próximo Natal será no ano seguinte!

Input

A entrada consiste numa linha com dois inteiros, D e M, separados por um espaço, que representam, respetivamente, o dia e o mês de uma data.

Restrições

$1 \leq M \leq 12$ O mês

$1 \leq D \leq 28$ O dia se $M = 2$ (fevereiro)

$1 \leq D \leq 30$ O dia se $M \in \{4, 6, 9, 11\}$ (meses com 30 dias)

$1 \leq D \leq 31$ O dia se $M \in \{1, 3, 5, 7, 8, 10, 12\}$ (meses com 31 dias)

Output

A saída deve ter um único inteiro de 0 a 364, que representa o número de dias até ao próximo Natal.

Exemplo 1

Input

```
1 12
```

Output

```
24
```

Exemplo 2

Input

```
1 1
```

Output

```
358
```

Exemplo 3

Input

```
25 12
```

Output

```
0
```

Exemplo 4

Input

```
26 12
```

Output

```
364
```

Melhores amigos

Ter amigos é muito especial. Com alguns até conseguimos imaginar o que eles estão a pensar. Para saber se um grupo de colegas da escola é um verdadeiro grupo de amigos, o João desenvolveu um jogo. A ideia é ter duas perguntas a que os amigos têm de responder separadamente. Na primeira questão é-lhes pedido para todos darem a mesma resposta e na segunda é-lhes pedido para todos darem respostas diferentes. Qualquer uma das perguntas requer que cada um dos colegas tenha uma ideia do que os outros irão responder. Para simplificar o jogo, vamos considerar que as respostas são números naturais.

O grau de amizade do grupo de colegas depende de quão bem estiveram nas duas perguntas, isto é, se todos coincidiram na primeira resposta e se todos diferiram na segunda resposta. Mais concretamente, o grau de amizade é 2 se todos responderem o mesmo à primeira pergunta e todos derem respostas diferentes à segunda pergunta. Se apenas um destes casos se verificar, o grau de amizade é 1. Se nenhum deles se verificar, o grau de amizade é 0.

Consegues ajudar o João a calcular o grau de amizade, dadas as respostas dos colegas às duas perguntas?

Tarefa

Escreva um programa que, dados o número de colegas e as respostas destes às duas questões, calcula o grau de amizade do grupo.

Input

O input tem três linhas. A primeira linha tem um inteiro, C, que representa o número de colegas do grupo. Cada uma das outras linhas tem C números inteiros, $n_1, n_2, \ldots, n_C$, separados por um espaço. Os valores da segunda linha correspondem às respostas dadas à primeira pergunta, em que lhes é pedido para indicarem o mesmo número. Os valores da terceira linha correspondem às respostas dadas à segunda pergunta, em que lhes é pedido para indicarem números diferentes.

Restrições

$2 \leq C \leq 8$ Número de colegas do grupo

$1 \leq n_i \leq 20$ Resposta de um colega a uma pergunta (para $i = 1, 2, \ldots, C$)

Output

O output tem uma linha com um inteiro, que representa o grau de amizade do grupo de colegas.

Exemplo 1

Input

```
4
1 1 1 1
3 8 6 1
```

Output

```
2
```

Exemplo 2

Input

```
6
5 5 5 5 5 5
4 3 1 1 8 2
```

Output

```
1
```

Exemplo 3

Input

```
5
5 3 5 5 5
7 5 1 1 4
```

Output

```
0
```

Exemplo 4

Input

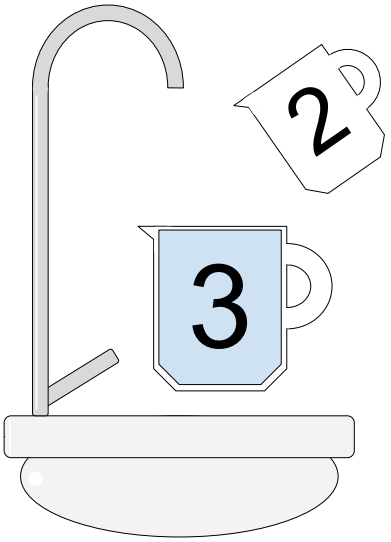
```
6
4 5 5 5 5 5
4 3 1 9 8 2
```

Output

```
1
```

Não metas água!

Imaginemos que temos de medir $1L$ (um litro) de água, mas só temos dois jarros, um de $3L$ e outro de $2L$. Uma solução será encher o jarro de $3L$ de uma torneira, depois o de $2L$ com a água do primeiro, que ficará exatamente com $1L$. Mas se tivermos um jarro de $5L$ e outro de $2L$? Podemos encher o de $5L$ da torneira, encher o de $2L$ com esta água, esvaziar esse jarro na pia, e voltar a enchê-lo com os $3L$ remanescentes do jarro maior, que fica apenas com $1L$.



Estes *puzzles*, que requerem medir volumes usando vários jarros não graduados, são um clássico em Inteligência Artificial. Podemos recorrer apenas a operações de três tipos: encher completamente um jarro a partir de uma torneira; esvaziar completamente um jarro para uma pia; ou encher um jarro com o conteúdo de outro. Neste último caso, ou o jarro que recebe a água fica completamente cheio, ou o jarro que fornece a água fica completamente vazio.

No final de uma sequência de operações destas, um dos jarros deverá ter um determinado volume alvo. Estes volumes, quer o alvo, quer a capacidade dos jarros, costumam ser valores inteiros positivos, todos na mesma unidade (litro, por exemplo).

Tarefa

Escreva um programa que recebe a descrição de um *puzzle* e uma sequência de operações, e determina se essa sequência é uma solução do *puzzle*, ou não.

Input

O input começa com a descrição do *puzzle*, que é seguida por uma sequência de operações.

A descrição do *puzzle* começa com $N+1$ linhas, cada uma com um único inteiro. As primeiras N linhas têm as capacidades C_i dos N jarros, ou seja, o número na i-ésima linha representa a capacidade do jarro número i (para $i = 1, 2, \ldots, N$). A última destas linhas tem o valor 0. Segue-se uma linha com um inteiro A, que é o alvo a atingir.

Finalmente, as operações são descritas por S linhas, cada uma com dois inteiros, F e D, em que F representa a fonte da água e D o destino dessa água, com o seguinte significado:

$$\begin{array}{rcll} & F & = & 0 \quad \text{a fonte da água é a torneira;} \\ 0 & < F & \leq & N \quad \text{a fonte da água é o jarro número } F; \\ & D & = & 0 \quad \text{o destino da água é a pia;} \\ 0 & < D & \leq & N \quad \text{o destino da água é o jarro número } D. \end{array}$$

Esta sequência termina com uma linha com dois zeros (0 0).

Restrições

$2 \leq N < 10$	Número de jarros
$1 \leq A < 10$	Alvo a atingir
$1 \leq C_i < 10$	Capacidade do jarro número i (para $i = 1, 2, \ldots, N$)
$0 \leq F \leq N$	Fonte da água numa operação
$0 \leq D \leq N$	Destino da água numa operação
$1 \leq S \leq 15$	Tamanho da sequência de operações

Output

O output deve ter uma única linha com uma única palavra, em maiúsculas: "`CERTO`", se a sequência de operações for uma solução do *puzzle*; "`ERRADO`", no caso contrário.

Exemplo 1

Input

```
2
3
0
1
0 2
2 1
0 0
```

Output

```
CERTO
```

Exemplo 2

Input

```
5
2
0
1
0 1
1 2
2 0
1 2
0 0
```

Output

```
CERTO
```

Exemplo 3

Input

```
3
6
9
0
2
0 1
1 2
2 0
0 0
```

Output

```
ERRADO
```

Vai um?

Segundo o *Programa e Metas Curriculares para Matemática do Ensino Básico*, de 2013, é fundamental que os alunos adquiram "fluência de cálculo e destreza na aplicação dos quatro algoritmos, próprios do sistema decimal", associados às quatro operações sobre os números naturais.

https://www.youtube.com/watch?v=Twe2Hmg6ghM

Aprender a somar parece ter muito que se lhe diga. As metas para o 1º Ano referem "Adições cuja soma seja inferior a 100", para o 2ºAno, "Adições cuja soma seja inferior a 1000" e, para o 3ºAno, "Algoritmos da adição e da subtração envolvendo números até um milhão".

Queremos simplesmente explorar uma das várias estratégias que se ensinam para a soma, a designada por "conta em pé". Para $28 + 45 = ?$, começa-se pelas unidades, $8 + 5 = 13$. Como excede 9, fica 3 nas unidades, e "vai 1", para a posição seguinte, ou seja, para as dezenas. Como $1 + 2 + 4 = 7$ não excede 9, já não vai nada para as centenas, e conclui-se que $28 + 45 = 73$.

Vamos agora tentar um exemplo maior, $5489649 + 759340613 = ?$, com a "conta em pé".

[0]	[0]	[1]	[0]	[1]	[1]	[1]	[0]	[1]	[0]	← o que vai?
			5	4	8	9	6	4	9	
+	7	5	9	3	4	0	6	1	3	
	7	6	4	8	3	0	2	6	2	

Por vezes, erramos a conta. Se admitirmos que nunca nos enganamos quando somamos números inferiores a 10, quantas vezes nos esquecemos do "e vai 1" no exemplo seguinte?

	[1]	[1]	[0]	[0]	[0]	[0]	[0]	[1]	[0]	← o que vai?
			5	9	7	9	6	4	9	
+		9	9	9	2	0	6	1	3	
	1	0	4	8	9	9	2	6	2	

Duas vezes: quando passamos de $6 + 6$ para $9 + 0$ e quando passamos de $9 + 9$ para $5 + 9$. Os restantes algarismos errados são consequência destes erros e, por isso, não contam.

Tarefa

Escreva um programa que, dados dois inteiros positivos x e y, e o resultado r que obtivemos para $x + y$, possivelmente errado, indique quantas vezes nos esquecemos do "e vai 1".

Input

A primeira linha tem três inteiros positivos, m, n e p, separados por um espaço, que são o número de dígitos de x, de y e de r, respetivamente. Seguem-se três linhas, com m, n e p inteiros (de 0 a 9) separados por espaços, que são os dígitos de x, y e r, lidos da esquerda para a direita, segundo a representação habitual, sem zeros à esquerda.

Restrições

$1 \leq m \leq 1000$	Número de dígitos de x
$1 \leq n \leq 1000$	Número de dígitos de y
$\max(m, n) \leq p \leq \max(m, n) + 1$	Número de dígitos de r

Output

O output tem um inteiro não negativo, que é o número de esquecimentos de "e vai 1".

Exemplo 1

Input

```
2 2 2
2 8
4 5
6 3
```

Output

```
1
```

Exemplo 2

Input

```
7 9 9
5 4 8 9 6 4 9
7 5 9 3 4 0 6 1 3
7 6 4 8 3 0 2 6 2
```

Output

```
0
```

Exemplo 3

Input

```
8 7 9
9 9 9 2 0 6 1 3
5 9 7 9 6 4 9
1 0 4 8 9 9 2 6 2
```

Output

```
2
```

Exemplo 4

Input

```
1 1 1
4
8
2
```

Output

```
1
```

Exemplo 5

Input

```
8 5 9
9 9 9 2 0 6 1 3
7 9 6 4 9
1 0 0 0 0 0 2 6 2
```

Output

```
0
```

Tudo em família!

– Algo está mal no negócio da família! Diz o padrinho da máfia local.

– Muitos querem mandar e poucos querem fazer. E temos ainda quem nos traia e fale com a polícia! Continua ele.

– Tenho de mandar tratar do assunto! Afinal, sou ou não sou o padrinho aqui?

Tarefa

Qual o desfecho de uma ordem que o padrinho dá: Será executada? Ficará por executar? Ou a polícia será informada?

Admita que a família tem m membros, além do padrinho, os quais são identificados por inteiros de 0 a $m-1$. O que o membro i da família faz quando recebe uma ordem é representado por um inteiro f_i, que pode significar uma de três coisas:

- Se $f_i < 0$ ou $f_i \geq m$, o membro i da família conta à polícia! É um *informador*.
- Se $i \neq f_i$ e $0 \leq f_i < m$, então o membro i da família gosta de dar ordens e ordena ao membro f_i que faça o que lhe foi pedido. É um *mandrião*.
- No caso que resta, quando $i = f_i$, o membro i da familia é alguém que faz o que lhe é pedido. É um *executante*, bem ao gosto do padrinho.

Pretendemos um programa que calcule o número de mandriões pelos quais é preciso passar até chegar a um executante, quando a ordem é dada a um determinado membro da família. Por exemplo, na família:

0	1	2	3	4	5	6	7	8	9
2	3	8	10	6	5	7	9	5	6

se a ordem for dada ao membro 5, que é um executante, a ordem passa por 0 (zero) mandriões. Mas, se a ordem fosse dada ao membro 0, passaria por 3 mandriões: o mandrião 0 passaria a ordem a 2, o mandrião 2 passaria a ordem a 8 e o mandrião 8 passaria a ordem a 5, que a executaria ($0 \to 2 \to 8 \to 5$).

Há situações em que o número de mandriões não está definido. Designamo-las por caso `POLICIA` e caso `INCOMPETENTE`. No primeiro caso, a ordem acaba na polícia. Por exemplo, se a ordem for dada ao membro 1, este passa-a ao informador 3. No segundo caso, a ordem não chega à polícia mas também não é executada. Por exemplo, se a ordem for dada ao membro 4, fica a circular por alguns membros da família: $4 \to 6 \to 7 \to 9 \to 6 \to 7 \to 9 \to 6 \to 7 \to \ldots$

Input

A primeira linha da entrada tem um inteiro m, que representa o número de elementos da família (para além do padrinho). Seguem-se m linhas, cada uma com um inteiro f_i, que indica

o que o membro i faz (começando pelo membro 0 e terminando no membro $m-1$). Depois, há uma linha com um inteiro, k, que indica a quem o padrinho dá a ordem.

Restrições

$0 < m \leq 1000$	Número de membros da família (para além do padrinho)
$0 \leq k < m$	Membro a quem o padrinho dá a ordem
$-3000 \leq f_i \leq 3000$	Indica o que o membro i faz (para $i = 0, 1, \ldots, m-1$)

Output

O output tem uma linha com:

- a palavra "`POLICIA`", caso a ordem acabe na polícia;
- a palavra "`INCOMPETENTE`", caso a ordem não seja executada nem acabe na polícia;
- um inteiro que representa o número de mandriões pelos quais a ordem passa, caso seja executada.

Exemplo 1

Input

```
10
2
3
8
10
6
5
7
9
5
6
4
```

Output

```
INCOMPETENTE
```

Exemplo 2

Input

```
10
2
3
8
10
6
5
7
9
5
6
0
```

Output

```
3
```

Exemplo 3

Input

```
12
6
2
8
3
10
19
8
-2
7
11
10
1
9
```

Output

```
POLICIA
```

Festa da empresa

Como o funcionário mais jovem da *Tip-Topas, Lda.*, tens a tarefa de organizar a festa de Natal da empresa. Tal como acontece com todos os organizadores de festas, o teu objetivo é, obviamente, maximizar a quantidade de diversão (medida, claro, em *Unidades Standard de Diversão* ou *USD*s).

Uma vez que algumas pessoas são mais carismáticas do que outras, estimou-se previamente a quantidade de diversão que cada funcionário trará à festa. A soma da quantidade de diversão dos funcionários presentes na festa indica a quantidade total de diversão na festa. Bem, há um pequeno problema: é um facto bem conhecido que qualquer pessoa cujo supervisor imediato também esteja na festa não se vai divertir nada. Portanto, se todos os funcionários da empresa estiverem na festa, apenas o CEO se vai divertir, e todos os outros vão ficar com riso nervoso a tentar parecer naturais enquanto procuram o seu supervisor pelo canto do olho...

O objetivo é descobrir quem convidar para a festa, de modo a maximizar a quantidade total de diversão.

Tarefa

Escreva um programa que lê a descrição da hierarquia da empresa e a quantidade de diversão de cada funcionário e calcula a maior quantidade total de diversão que é possível ter na festa. Os funcionários (incluindo o CEO) são numerados de 1 a N. Cada funcionário supervisiona exatamente dois outros ou então não supervisiona ninguém. Além disso, cada funcionário tem exatamente um supervisor, exceto o CEO, que não tem nenhum. Pode ainda assumir que os funcionários são numerados por ordem hierárquica, isto é, que se i supervisiona j, então $i < j$. Logo, o CEO terá sempre o número 1.

Input

A primeira linha da entrada tem um inteiro N, que representa o número de funcionários da empresa.

Seguem-se N linhas que descrevem a hierarquia da empresa. Cada linha tem dois inteiros, j e k, separados por um espaço. Se o funcionário i supervisionar alguém, os valores j e k da i-ésima linha são os números (distintos) dos dois funcionários supervisionados por i. Se o funcionário i não for supervisor de ninguém, os valores j e k da i-ésima linha são ambos zero.

Depois, há mais N linhas, cada uma com um inteiro D_i, que denota a quantidade de diversão do funcionário i, em USD (para $i = 1, 2, \ldots, N$).

Restrições

$3 \leq N < 100$ Número de funcionários

$0 \leq D_i \leq 100$ Diversão do funcionário i (para $i = 1, 2, \ldots, N$)

Output

A saída tem uma única linha com a forma "T `USD`", em que T é a maior quantidade total de diversão que é possível ter na festa.

Exemplo 1

Input

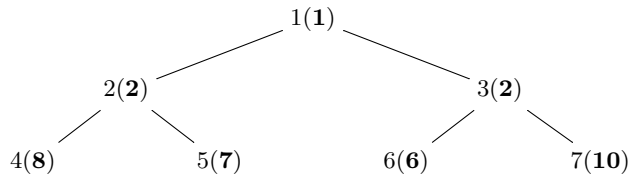
```
7
2 3
4 5
6 7
0 0
0 0
0 0
0 0
1
2
2
8
7
6
10
```

Output

```
32 USD
```

Explicação

A hierarquia da empresa está representada na seguinte figura (em que a quantidade de diversão está anotada em negrito):



A quantidade máxima de diversão obtém-se convidando os funcionários 1, 4, 5, 6 e 7 (e excluindo os funcionários 2 e 3); o seu valor é:

$$1 + 8 + 7 + 6 + 10 = 32 \text{ USD}.$$

Exemplo 2

Input

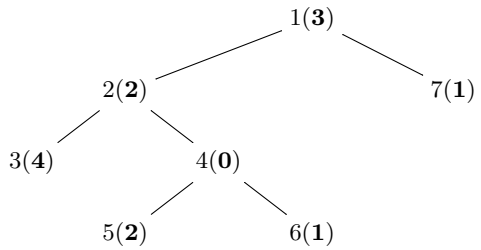
```
7
2 7
3 4
0 0
5 6
0 0
0 0
0 0
3
2
4
0
2
1
1
```

Output

```
10 USD
```

Explicação

A hierarquia da empresa pode ser esquematizada da seguinte forma:



A quantidade máxima de diversão obtém-se convidando apenas os funcionários 1, 3, 5 e 6; o seu valor é:

$$3 + 4 + 2 + 1 = 10 \text{ USD}.$$

Zega Zinga, Zoga Zonga

Zega Zinga: *Huummm... Onze.*
Zoga Zonga: *Doze!*

Zega Zinga: *Onze!!*
Zoga Zonga: *Não! Doze!!*

Zega Zinga: *A zério?*
Zoga Zonga: *Zim. Doze.*

Zega Zinga: *Zurpreendente! Como zegazte tão deprezza à zoluzão?*

Zega Zinga e Zoga Zonga, abelhas de alta competição, têm um problema: o campo de treino dos *Bem-me-queres floridos*, onde diariamente saciam o seu apetite, foi tomado pelas Vespas. Por isso, atravessar o campo requer agora outros cuidados. Quais? Escolher a flor de partida no lado do campo onde se encontram. Escolher como próxima flor uma das três vizinhas imediatamente à frente (nunca flores ao seu lado ou atrás). Repetir até chegar ao outro lado do campo, sem se esquecerem de recolher o néctar, claro. As Vespas? Certo. Convém evitar as Vespas disfarçadas de flor.

Podem ajudar Zega Zinga a calcular a maior quantidade de néctar que pode recolher numa travessia do campo, evitando, obviamente, as Vespas?

Tarefa

As flores do campo dos *Bem-me-queres floridos* estão dispostas numa grelha retangular. Uma descrição do campo especifica, para cada flor, a quantidade de néctar presente na flor ou se está uma Vespa pousada na flor.

Escreva um programa que, dada uma descrição do campo, indique a maior quantidade de néctar que pode ser recolhida pela abelha Zega Zinga numa travessia do campo (ou seja, num caminho do topo até à base do campo). Poderá ser escolhida como flor de partida qualquer uma das pertencentes à linha do topo (que corresponde à primeira linha), desde que não tenha Vespa. A deslocação terá de ser sempre entre flores vizinhas entre si, podendo ser utilizada uma única flor por cada linha. Flores com Vespas não podem fazer parte do caminho. Caso não exista nenhum caminho que atravesse o campo de um lado ao outro (que comece na primeira linha e termine na última linha), devido à presença de Vespas, deverá ser escrito o valor zero.

Input

A primeira linha do input contém dois inteiros, L e C, que representam, respetivamente, o número de linhas e o número de colunas da grelha de flores que descreve o campo.

Seguem-se L linhas, cada uma com C elementos, separados por um espaço. Cada elemento é:

- a letra "`V`", caso se trate de uma flor com uma Vespa;
- um inteiro N, que corresponde à quantidade de néctar da flor, caso a flor esteja livre de Vespas.

Restrições

$1 \leq L \leq 100$ Número de linhas do campo
$1 \leq C \leq 100$ Número de colunas do campo
$0 \leq N \leq 100$ Quantidade de néctar de uma flor

Output

O output tem uma única linha com:

- um inteiro que representa a maior quantidade de néctar que pode ser recolhida pela abelha Zega Zinga numa travessia do campo (ou seja, num caminho da primeira até à última linha do campo);
- "`0`" (zero), se não existir um caminho da primeira até à última linha do campo (conforme definido acima), devido à presença de Vespas.

Exemplo 1

Input

```
3 4
5 2 3 1
1 1 1 4
5 2 5 2
```

Output

```
12
```

Explicação

Zega Zinga pode recolher 3 unidades de néctar na primeira linha, 4 unidades na segunda linha e 5 unidades na terceira linha.

Exemplo 2

Input

```
3 4
5 2 3 1
1 1 V V
5 2 5 15
```

Output

```
11
```

Explicação

Zega Zinga pode recolher 5 unidades de néctar na primeira linha, 1 unidade na segunda linha e 5 unidades na terceira linha. Tem várias maneiras de o fazer.

Exemplo 3

Input

```
3 4
5 2 V V
V V V 4
5 2 5 15
```

Output

```
0
```

Explicação

Zega Zinga não consegue atravessar o campo, devido às Vespas.