Problem A

Faith

Source file: faith.{ c | cpp | java | py }

Lino is confident he'll win the lottery soon. He has been dreaming about driving a Ferrari and traveling around the world.

His previous attempt to get rich, as a politician, was not successful. A country music singer that was a congressional candidate didn't receive as many votes as expected and ended up not favoring candidates of his party.

Despite his failure in the previous journey, he hasn't lose hope. He believes that he is special for God in a different manner than others are and that he was born to be a rich man.

One of his beliefs is that God will send him a sequence of winner lottery numbers. He has already dreamed about numbers, but was no lucky with them. Thus, he decided to follow a different strategy. The next time he dreams with numbers, let's say 2 and 4, he will bet on the second and fourth most common numbers in previous lottery drawings.

He is asking your help to write a program that, given previous lottery drawings and the numbers he dreamed about, prints the numbers he must bet on.

Input

The first line of input consists of an integer N ($1 \leqslant N \leqslant 500$), which indicates the number of lottery drawing results that will be informed. The next N lines contain six integer numbers B each ($0 < B \leqslant 60$), separated by a space, indicating a number that was drawn in a previous lottery drawing. The next line contains six integer numbers D each ($0 < D \leqslant 60$), separated by a space, indicating the numbers Lino has dreamed about. Assume that a number that Lino will dream about will never be bigger than the amount of different numbers that appeared in previous lottery drawings.

Output

Print six integer numbers, separated by a space, representing the numbers that Lino should bet on considering his strategy. If two or more numbers appeared the same amount of times in previous lottery drawings, the lower ones has precedence over the bigger ones. The output line must be ended with a line break.

Example of Input 1	Example of Output 1
5 2 4 6 8 10 12 1 4 7 9 20 30 22 33 41 2 3 7 32 35 44 60 21 12 1 2 3 4 5 6 2 5 7 1 9 10	4 6 12 2 8 9

Problema B

Um pra você, um pra mim

Arquivo fonte: pramim.{ c | cpp | java | py }

A animação Pica-Pau (*Woody Woodpecker*™, no original) é um clássico reprisado há décadas nas redes de televisão. No episódio *Esperto contra sabido* (1964), o Raposo Fink divide alguns alimentos com o protagonista, porém de forma questionável, pois no fim nada sobra para o Pica-Pau.

Após assistirem ao episódio, você e seu amigo decidiram esvaziar o cofrinho em que ambos guardam suas economias e dividi-las. Como no cofrinho há apenas notas (2, 5, 10, 20, 50 e 100 reais), a sugestão foi coletar uma nota qualquer e dar para um dos dois. Na sequência, coleta-se outra nota e dá-se a quem não ganhou a anterior, e assim sucessivamente, até acabarem as notas.

Seu amigo quer saber quanto dinheiro cada um terá no final. Você, um bom desenvolvedor, enxergou a oportunidade de automatizar o processo, construindo um programa que, dado quem ganha a primeira nota, a quantidade e a sequência de notas coletadas, exibe quanto cada um ganhou.

Entrada

A entrada é composta por um caractere, indicando quem ganhará a primeira nota, 'V' ou 'A' (você ou amigo, respectivamente); a quantidade de notas $N (1 \leqslant N \leqslant 10000)$; e N linhas, cada uma com um valor natural representando uma nota.

Saída

Imprima a frase "*VOCE: V AMIGO: A*", onde 'V' e 'A' são, respectivamente, o seu total de reais e o total de seu amigo. Finalize com uma quebra de linha.

Exemplo de Entrada 1

```
V 4
10
20
50
2
```

Exemplo de Saída 1

```
VOCE: 60 AMIGO: 22
```

Exemplo de Entrada 2

```
A 7
10
10
50
20
100
2
5
```

Exemplo de Saída 2

```
VOCE: 32 AMIGO: 165
```

Esta página foi propositadamente deixada em branco.

Problema C

A Horta do Juquinha

Arquivo fonte: horta.{ c | cpp | java | py }

Juquinha ganhou na escola 3 pacotes de sementes, que foram distribuídos em um evento sobre alimentos orgânicos promovido pela professora de Ciências. Cada pacote é de um vegetal alimentício diferente, e ele pretende plantá-los no quintal do seu avô, que prometeu ajudar na semeadura. Todo pacotinho indica a quantidade de sementes que deve ser colocada em cada cova feita no chão, além de informar os cuidados básicos a serem adotados para que os brotos vinguem. Agora o avô de Juquinha está querendo saber quantas covas ele precisa abrir na terra para que o neto possa despejar as sementes corretamente.

Sua tarefa neste problema é ajudar Juquinha a determinar a quantidade de covas que precisarão ser abertas no solo, com base nas quantidades totais de sementes disponíveis e de sementes que devem ser colocadas em cada cova.

Entrada

A entrada consiste de um único caso de teste, composto por 3 linhas, cada uma descrevendo um saquinho de sementes na forma de dois inteiros T $(1 \leqslant T \leqslant 100)$ e Q $(1 \leqslant Q \leqslant T)$, respectivamente a quantidade total de sementes contidas no saquinho e a quantidade daquelas sementes que deve ser colocada em cada cova.

Saída

Imprimir um inteiro indicando a quantidade de covas a serem abertas pelo avô de Juquinha, seguido por uma quebra de linha. Considere que apenas serão abertas covas para a quantidade correta de sementes especificada no saquinho, o que significa que poderá ocorrer alguma sobra de sementes.

Exemplo de Entrada 1	**Exemplo de Saída 1**
30 3 10 2 20 5	19

Exemplo de Entrada 2	**Exemplo de Saída 2**
25 6 20 3 10 3	13

Exemplo de Entrada 3	**Exemplo de Saída 3**
100 100 100 1 1 1	102

Esta página foi propositadamente deixada em branco.

Problema D

Impossível

Arquivo fonte: impossivel.{ c | cpp | java | py }

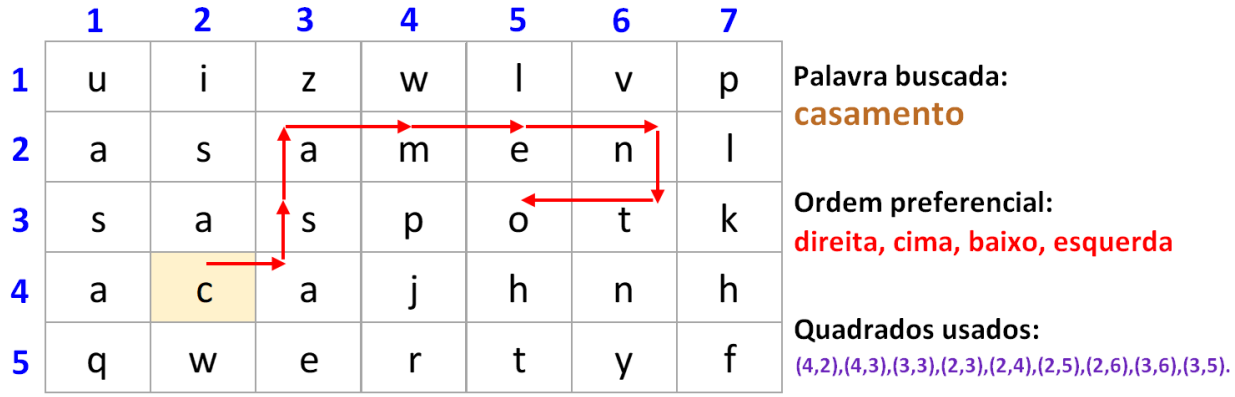
Maria Antônia elaborou um desafio interessante: dada uma folha quadriculada em que cada quadrado contém uma letra, verifica-se a possibilidade de partir de um quadrado qualquer e traçar uma seta entre ele e um quadrado adjacente, depois do adjacente escolhido até um adjacente dele, e assim por diante, tentando formar uma palavra buscada (vide Figura 1).

As regras: (a) a partir de um quadrado, é permitido traçar somente uma seta, na horizontal ou vertical, com destino a outro adjacente; (b) cada quadrado pode ser usado só uma vez para compor a palavra; (c) as setas que ligam os quadrados devem ser feitas na mesma sequência em que as letras ocorrem na palavra; (d) haverá uma ordem preferencial em que os quadrados adjacentes serão usados, a preferência será uma permutação de [*cima*, *baixo*, *esquerda*, *direita*]; (e) há no máximo uma sequência que forma a palavra.

Para exemplificar o funcionamento da ordem preferencial, suponha que a permutação seja [*direita*, *cima*, *baixo*, *esquerda*]. Sempre que for possível formar a palavra unindo o quadrado atual com o da *direita*, essa será a escolha, se não for possível, o próximo preferencial será o de *cima*, a terceira preferência será o de *baixo* e, por último, o da *esquerda* (vide Figura 1).

Cada quadrado possui uma coordenada, que é sua linha e coluna, respectivamente. Se for possível formar a palavra seguindo as regras, escreve-se as coordenadas de todos os quadrados na ordem que foram usados. Caso contrário, declara-se que a busca é impossível. Seu objetivo é automatizar esse procedimento com um programa de computador.

Figura D.1: Exemplo de uma folha com as setas formando a palavra buscada.



Entrada

A entrada é composta pela palavra buscada P $(1 \leqslant tamanho(P) \leqslant 100)$; pela ordem preferencial representada por uma permutação dos caracteres ['*c*', '*b*', '*e*', '*d*'] (*cima*, *baixo*, *esquerda* e *direita*, respectivamente); pelos números de linhas L e colunas C da folha $(1 \leqslant L, C \leqslant 100)$; por L linhas com C minúsculas cada, representando os quadrados.

Saída

Se for possível montar a palavra, imprima as coordenadas dos quadrados, separadas por vírgulas, na mesma sequência em que foram usados, e um ponto final. Caso contrário, imprima "*impossivel!*"(minúsculo, não acentuado e sem aspas). Finalize com uma quebra de linha.

Exemplo de Entrada 1 **Exemplo de Saída 1**

casamento d c b e 5 7 uizwlvp asamenl saspotk acajhnh qwertyf	(4,2),(4,3),(3,3),(2,3),(2,4),(2,5),(2,6),(3,6),(3,5).

Exemplo de Entrada 2 **Exemplo de Saída 2**

casamento c d b e 5 7 uizwlvp asamenl saspotk acajhnh qwertyf	(4,2),(3,2),(2,2),(2,3),(2,4),(2,5),(2,6),(3,6),(3,5).

Exemplo de Entrada 3 **Exemplo de Saída 3**

amazonas b c d e 5 6 rsrsrs qanoik emazkk gfdsmk lkjhhk	impossivel!

Exemplo de Entrada 4 **Exemplo de Saída 4**

fatec d e b c 3 5 cetfa aaaaa bbbbb	impossivel!

Problema E

Língua do PQR

Arquivo fonte: linguadopqr.{ c | cpp | java | py }

Maike Diordan e Coube Braian são dois meninos da comunidade de Cachoeirinha que adoram brincar com códigos de comunicação. Para que seus colegas não entendam o que eles conversam, costumam usar a língua do P. Basta colocar a letra P antes de cada sílaba. Por exemplo, a frase "Como você está?"fica "Pco Pmo Pvo Pcê Pes Ptá?", com o P sendo pronunciado como 'Pê'.

Como usam essa língua há algum tempo, seus colegas já estão especialistas em decifrar o que dizem. Por isso, eles decidiram modificá-la, criando a língua do PQR. Ela é bem similar à anterior, com a diferença de que, ao invés de usar apenas P antes das sílabas, eles usam PQ - pronunciando "PêQuê". Ainda, entre palavras, entre símbolos, e entre palavras e símbolos, eles usam R - pronunciando "Érre". Desta forma, a frase "Como você está?"fica "PQco PQmo R PQvo PQcê R PQes PQtá R ?".

Sabendo que eles não usam qualquer palavra nem qualquer abreviatura que tenha as letras PQ em sequência e também que R não é um conector de palavras, escreva um programa para ajudar seus colegas a descobrirem as frases que estão falando.

Entrada

A entrada contém uma única linha com uma sequência de caracteres C, de tamanho máximo 100, representando a frase que Maike e Coube disseram. Esta sequência tem apenas caracteres maiúsculos e minúsculos (inclusive com acentos e cedilha), espaços e os seguintes caracteres: '?', '.', ',' e '!'.

Saída

Como saída, o seu programa deverá imprimir a frase falada por Maike e Coube decodificada. Esta frase deve terminar com uma quebra de linha.

Exemplo de Entrada 1

```
PQEs PQtá R PQma PQlu PQco R , R PQMai PQke R ?
```

Exemplo de Saída 1

```
Está maluco, Maike?
```

Exemplo de Entrada 2

```
PQÔ R PQLou PQco R , R PQmeu R ! R PQEs PQse R PQfe PQra R PQa PQí R !
```

Exemplo de Saída 2

```
Ô Louco, meu! Esse fera aí!
```

Esta página foi propositadamente deixada em branco.

Problema F

Fila do Almoço

Arquivo fonte: fila.{ c | cpp | java | py }

A Câmara Municipal de Tontinho oferece almoço gratuito aos vereadores e servidores, mas alguns estão abusando e repetindo a refeição inúmeras vezes. Procurando inibir o ato, os responsáveis pelo setor de alimentação criaram senhas para o almoço. Porém, alguns vereadores montaram um esquema de duplicação e criação de senhas e ainda estão repetindo a refeição. Desesperados, alguns cidadãos procuraram o presidente da câmara, um dos poucos que não aderiram ao esquema, e solicitaram a criação de uma solução computacional para resolver este problema.

A ideia é que cada vereador e servidor receberá uma senha e, ao solicitar a refeição, deverá entregá-la. Caso a senha seja repetida ou tenha um número inválido, o funcionário não deverá entregar a refeição. No final do almoço, o presidente da câmara gostaria de saber quantos almoçaram, quantas senhas inválidas e quantas repetidas foram entregues. Note que uma senha é considerada repetida quando tem a mesma identificação de outra já entregue, não importando se ela é válida ou inválida.

Entrada

A primeira linha de cada caso de teste contém um número inteiro S $(1 \leqslant S \leqslant 200)$, que especifica o número de senhas criadas pela empresa. A próxima linha contém cada uma das senhas separas por um espaço em branco. Cada senha é composta por um número inteiro I $(1 \leqslant I \leqslant 10^5)$ e um caractere maiúsculo C. A linha seguinte contém um número inteiro F $(1 \leqslant F \leqslant 500)$, especificando a quantidade de funcionários na fila. A última linha do caso de teste contém valores separados por espaço representando cada senha com X caracteres $(1 \leqslant X \leqslant 200)$ podendo conter letras e/ou números, fornecida pelos funcionários da câmara.

Saída

Para cada caso de teste, imprima a quantidade de almoços servidos, seguido pela letra "A" representando almoço. Na próxima linha, imprima a quantidade de senhas inválidas e a letra "I" e a quantidade de senhas repetidas e a letra "R". A última linha de saída deve terminar com uma quebra de linha.

Exemplo de Entrada 1	**Exemplo de Saída 1**
3 1C 2W 300C 5 7 1C 9 1B 1C	1 A 3 I 1 R

Esta página foi propositadamente deixada em branco.

Problema G

Jogo de TV

Arquivo fonte: jogo.{ c | cpp | java | py }

Airton gosta muito de televisão e, quando assiste, não se importa com mais nada ao seu redor, principalmente quando o programa exibido é um daqueles com jogos em que o telespectador participa e concorre a um prêmio. Um desses programas consiste em exibir um painel retangular composto por quadrados, todos com uma mesma imagem, exceto um (vide Figura 1), então o telespectador entra em contato com a emissora e informa a posição (linha e coluna) do quadrado diferente e, se acertar, é premiado!

Figura G.1: Exemplo de um possível painel do jogo.

Você simulará esse jogo, construindo um programa que recebe um painel retangular, em que cada quadrado é um caractere alfanumérico, e informa a linha e a coluna do único caractere diferente.

Entrada

A entrada é composta por L linhas $(1 \leqslant L \leqslant 10^5)$ com C caracteres $(3 \leqslant C \leqslant 100)$ cada.

Saída

Imprima a frase "*LINHA L COLUNA C*", onde 'L' e 'C' são, respectivamente, os números da linha e coluna do caractere diferente. Finalize com uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
DDDDD DDBDD DDDDD	LINHA 2 COLUNA 3

Exemplo de Entrada 2	Exemplo de Saída 2
aaa aaa aaa aaa aaa aaa aaa a5a	LINHA 8 COLUNA 2

Exemplo de Entrada 3	Exemplo de Saída 3
Zzz	LINHA 1 COLUNA 1

Exemplo de Entrada 4	Exemplo de Saída 4
zZZ	LINHA 1 COLUNA 1

Exemplo de Entrada 5	Exemplo de Saída 5
MMM M7M	LINHA 2 COLUNA 2

Problema H

Poupança Multi-data

Arquivo fonte: poupanca.{ c | cpp | java | py }

Seu tio Ariovaldo Bertolino, extremamente cético com o sistema financeiro, até pouco tempo atrás guardava seu dinheiro embaixo do colchão. Apavorado com as frequentes notícias que vê na TV, onde a polícia invade as casas das pessoas e leva todo dinheiro ilícito escondido, e com medo que alguém desconfiasse que seu dinheiro fosse ilícito, resolveu colocar seu dinheiro na Poupança. Mas sempre que precisa fazer uma retirada de sua poupança multi-data, fica intrigado com a maneira que os valores são retirados dos referidos dias do mês em que possui algum saldo, achando que o sistema do Banco está incorreto na escolha dos dias, fazendo-o perder rendimentos. Por este motivo, pediu para você, seu sobrinho preferido, fazer um sistema que dado o dia, o valor da retirada e o saldo de cada dia de sua poupança, mostre quais dias deve-se retirar valores de modo que se tenha a menor perda de rendimentos.

Nas suas pesquisas sobre o funcionamento da Caderneta de Poupança, você encontrou as seguintes regras:

- Aniversário da poupança, ou data base, é o dia em que foi realizado um depósito e é neste dia que será creditado o rendimento a cada 30 dias. Por exemplo, se você fez um depósito no dia 15 deste mês, no dia 15 de cada mês subsequente será aplicado o rendimento. Se retirar antes do aniversário, perde-se o rendimento sobre o valor sacado;
- Quando o depósito é feito nos dias 29, 30 ou 31, o aniversário, ou data base, será o dia 1 do mês seguinte, pois nem todos os meses possuem esses dias.

Então, qual estratégia usar para escolher o(s) dia(s) para debitar o valor da retirada? Bom, antes de mandar um clarification, analise os exemplos.

Entrada

A primeira linha da entrada possui um valor inteiro A ($1 \leqslant A \leqslant 31$), representando o dia da retirada, seguido de um valor real R ($0.01 \leqslant R \leqslant 1000000.00$) representando o valor da retirada feita neste dia. Cada uma das próximas linhas da entrada é composta por um número inteiro D ($1 \leqslant D \leqslant 31$), representando um dia do mês em que Tio Ariovaldo possui saldo na poupança (data base), seguido de um valor real de duas casas decimais S ($0.01 \leqslant S \leqslant 100000.00$), que representa o saldo no referido dia. Os dias serão apresentados em ordem crescente e existe um espaço separando os valores.

Saída

Na saída devem ser apresentados, em ordem crescente do dia, os dias que foram afetados pela retirada com as seguintes informações: a data base, saldo remanescente neste dia após a retirada e, entre parênteses, as palavras 'resgate de' seguido do valor retirado deste dia. Um espaço separa cada informação. Devem ser mostrados somente os dias afetados pela retirada. Escreva na saída a palavra INSUFICIENTE, caso não haja saldo suficiente para a retirada. Cada linha finaliza com uma quebra de linha, inclusive a última.

Exemplo de Entrada 1

```
15 1500.00
11 1500.00
12 1000.00
16 1000.00
```

Exemplo de Saída 1

```
11 1000.00 (resgate de 500.00)
12 0.00 (resgate de 1000.00)
```

Exemplo de Entrada 2

```
8 1835.40
9 1000.00
10 1000.00
11 1000.00
```

Exemplo de Saída 2

```
10 164.60 (resgate de 835.40)
11 0.00 (resgate de 1000.00)
```

Exemplo de Entrada 3

```
9 500.00
9 500.00
10 1000.00
11 1000.00
```

Exemplo de Saída 3

```
9 0.00 (resgate de 500.00)
```

Exemplo de Entrada 4

```
9 2600.00
9 500.00
10 1000.00
11 1000.00
```

Exemplo de Saída 4

```
INSUFICIENTE
```

Exemplo de Entrada 5

```
9 2600.00
9 500.00
10 1000.00
11 1000.00
28 500.00
```

Exemplo de Saída 5

```
9 0.00 (resgate de 500.00)
10 400.00 (resgate de 600.00)
11 0.00 (resgate de 1000.00)
28 0.00 (resgate de 500.00)
```

Problema I

Boleto

Arquivo fonte: boleto.{ c | cpp | java | py }

Desde o início de 2017, o Brasil vem mudando a forma de cobrança de boletos bancários. Anteriormente, qualquer pessoa podia emitir um boleto, desde que tivesse um CNPJ e uma conta em banco. Com as novas mudanças, além do CNPJ da empresa, os bancos vêm impondo que todos os boletos devem ser registrados, ou seja, com CPF, endereço e CEP válido. A NTG Soluções em Tecnologia trabalha com desenvolvimento de sistemas e vem sofrendo demais com essas novas mudanças, visto que antigamente ela utilizava uma API de terceiro para o desenvolvimento do *layout* de boletos e agora a própria NTG vem desenvolvendo todo o algoritmo para recebimento de boletos. Para a questão do intermédio entre as empresas e os bancos, utiliza-se um arquivo de texto num padrão conhecido como CNAB, com suas variantes. Dentro deste arquivo, há 240 caracteres em cada linha (CNAB 240) onde da coluna n até n+x, está uma informação, como número do boleto, data de vencimento, valor, etc. A tabela seguinte mostra um exemplo do padrão CNAB:

Tamanho	Coluna inicial	Coluna Final	Informação
6	5	10	Vencimento (DDMMYY)
6	11	16	Valor de Pagamento inteiro
2	17	18	Valor de Pagamento Decimal
4	19	22	Número Documento
6	23	28	Data de Pagamento

No momento, a empresa está desenvolvendo o processamento do arquivo de retorno, o qual contém as informações dos boletos pagos. Sua função será fazer um programa para extrair das cadeias de caracteres do arquivo texto, o somatório recebido com adimplência e inadimplência, baseando-se na data de pagamento.

Entrada

A entrada consiste em um único caso de teste por vez, informando-se várias linhas, cada qual contendo uma cadeia com 30 caracteres com informações sobre o pagamento de um boleto.

Saída

Seu programa deverá imprimir o valor total, com duas casas decimais, seguido do termo "ADIMPLENTE"para todo os boletos pagos em dia e, na outra linha, o valor total pago de todos os boletos com atraso seguido do termo "INADIMPLENTE". Esta última linha deve terminar com uma quebra de linha.

Exemplo de Entrada 1	**Exemplo de Saída 1**
00671210180002323001231705181500380811180004400006521705181600890706170002315023561705181700632905180002323467231705181200322004180005629034111705181400	904.64-ADIMPLENTE 794.40-INADIMPLENTE

Exemplo de Entrada 2	Exemplo de Saída 2
00673110180012323001231001181500383161118012130000652270518160089170617002331502356140518170063290518099232346723230718120032300418001562903411070518140	13362.30-ADIMPLENTE 103126.74-INADIMPLENTE

Problema J

Formula Rubens

Arquivo fonte: formularubens.{ c | cpp | java | py }

Rubinho é um famoso corredor de carros que já passou por várias categorias do automobilismo profissional. Agora, ele virou personagem principal de um jogo de videogame, o *Formula Rubens*. Era para o jogo ter sido lançado em dezembro de 2017, mas houve um atraso sem explicação e o jogo só chegará ao mercado em junho de 2018 (assim esperam os investidores).

A equipe que está desenvolvendo o jogo trabalhou em jogos anteriores de estratégia e simulação, como o SimCity e o SimFarm. Talvez pela experiência com estes jogos e também pelo contexto do jogo atual, foi decidido que o jogo terá um menu de desastres. Uma das opções deste menu é o *alagamento*. Durante uma corrida, caso o jogador selecione esta opção, um trecho da pista será alagado e o jogador poderá ver os carros dos competidores balançando e, algumas vezes, até batendo por causa do alagamento causado na pista.

A pista de corrida é representada no jogo por meio de uma matriz numérica. Cada número, variando de 0 a 255, representa o tom de cinza de um pixel que compõe a pista. Quando o jogador selecionar a opção *alagamento* no menu de desastres, será sorteada uma célula desta matriz. Ela e todas as células vizinhas que tenham valor com uma diferença máxima D do valor da célula selecionada deverão ser alagadas. Como exemplo, se uma célula com valor 240 for selecionada e o valor de D for 5, ela e toda célula vizinha com valor entre 235 e 245 deverá ser alagada. Uma célula é considerada vizinha de outra se estiver acima, abaixo, à direita ou à esquerda dela.

Auxilie os desenvolvedores na implementação desta funcionalidade.

Entrada

A primeira linha da entrada contém quatro números inteiros: L ($0 < L \leqslant 100$), C ($0 < C \leqslant 100$), Y ($0 < Y \leqslant L$) e X ($0 < X \leqslant C$), separados por um espaço em branco, representando, respectivamente, a quantidade de linhas e colunas da matriz da pista, e a linha e coluna da célula na qual será aplicado o *alagamento*. A próxima linha contém um número inteiro D ($0 \leqslant D \leqslant 127$) indicando a diferença aceitável de valor entre células vizinhas. As próximas L linhas contêm C números inteiros P ($0 \leqslant P \leqslant 255$), separados por um espaço em branco, representando os valores de cada célula da matriz da pista.

Saída

Como saída, o seu programa deverá imprimir um número inteiro indicando a quantidade de células que serão alagadas pelo menu de desastre do jogo. Finalize com uma quebra de linha.

Exemplo de Entrada 1

```
5 10 3 3
7
0 1 255 3 5 1 1 1 1 1
1 1 1 1 9 10 1 1 1 1
2 3 2 32 47 96 22 1 0 2
2 2 2 2 2 2 2 2 2 2
3 3 2 2 3 3 3 3 3 3
```

Exemplo de Saída 1

```
44
```

Exemplo de Entrada 2

```
4 3 1 2
67
107 168 47
161 223 133
117 191 198
136 147 230
```

Exemplo de Saída 2

```
11
```