ToPAS'16
Torneio de Programação para Alunos do Secundário

Departamento de Ciência de Computadores

http://www.dcc.fc.up.pt/topas/

Conjunto de Problemas

U.PORTO

Faculdade de Ciências da Universidade do Porto

6 de maio de 2015

Este conjunto de problemas deverá conter sete (7) problemas e dezassete (17) páginas.
Se faltar algum problema, por favor avise a organização.

Edição realizada em colaboração com:

UAlg FCT UNIVERSIDADE DO ALGARVE FACULDADE DE CIÊNCIAS E TECNOLOGIA

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

FCT FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA

Departamento de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

ToPAS'16
Torneio de Programação para Alunos do Secundário

Dep. Ciência de Computadores – FCUP
6 de maio de 2016

Conteúdo

Registo de Altos e Baixos do Caminhante

Um caminhante gosta de registar a distância percorrida em cada uma das suas caminhadas. E, de forma a controlar a sua evolução, quer conhecer a diferença entre as distâncias percorridas em caminhadas consecutivas.

Tarefa

Escreva um programa que, para cada caminhada registada (exceto a primeira), indica se a distância percorrida nessa caminhada aumentou (`ALTO`), manteve-se (`PATAMAR`) ou diminuiu (`BAIXO`) em relação à distância percorrida na caminhada anterior. Quando a distância aumentou ou diminuiu, o programa calcula a diferença entre as distâncias percorridas, em quilómetros e metros. O programa termina quando a distância percorrida é 0 quilómetros e 0 metros.

Input

O input tem $n + 1$ linhas.

Cada uma das n primeiras linhas tem o registo de uma caminhada. A linha tem dois inteiros, k e m, que indicam que o caminhante percorreu k quilómetros e m metros.

A última linha tem `0 0`.

Restrições

$2 \leq n \leq 1\,000$	Número de caminhadas
$0 \leq k \leq 100$	Quilómetros percorridos numa caminhada
$0 \leq m \leq 999$	Metros percorridos numa caminhada

Output

O output tem $n - 1$ linhas: uma linha por cada duas caminhadas consecutivas. A linha correspondente às caminhadas i e $i + 1$ tem o seguinte formato:

- `ALTO` k `km` m `m`
  se a distância percorrida na caminhada $i + 1$ aumentou k quilómetros e m metros em relação à distância percorrida na caminhada i;
- `PATAMAR`
  se a distância percorrida na caminhada $i+1$ se manteve em relação à distância percorrida na caminhada i;
- `BAIXO` k `km` m `m`
  se a distância percorrida na caminhada $i + 1$ diminuiu k quilómetros e m metros em relação à distância percorrida na caminhada i.

Exemplo 1

Input

```
10 200
9 800
10 250
20 0
20 0
0 900
0 0
```

Output

```
BAIXO 0 km 400 m
ALTO 0 km 450 m
ALTO 9 km 750 m
PATAMAR
BAIXO 19 km 100 m
```

Exemplo 2

Input

```
1 205
1 205
0 0
```

Output

```
PATAMAR
```

Curso de verão

A Ana e o Carlos, que moram no Porto, ficaram entusiasmados com a ideia de em julho ir à Universidade do Algarve frequentar um curso de verão para alunos do ensino secundário. Por isso, estão já a fazer os preparativos para a viagem.

O ideal será ir de comboio, do Porto até Faro. O preço normal da viagem, em classe turística, é €51.50. Felizmente, como ambos são jovens com menos de 26 anos, têm automaticamente direito a um desconto de 25%. Ora 25% de 51.50 é 18.87.

Na verdade, o valor exato seria ligeiramente maior, mas a CP faz as contas dos descontos até aos cêntimos. Logo, o preço da viagem ficaria em €38.63. Só que a CP, para facilitar os trocos (dizem eles...) arredonda todos os preços para cima, até ao múltiplo de 50 cêntimos imediato. Ou seja, a viagem fica por 39 euros para cada um.

Já não seria mau, mas a Ana e o Carlos descobriram que se comprarem o bilhete com cinco dias de antecedência ou mais terão um desconto adicional de 40%. Ou seja, como 40% de 39 são 15.6, o preço com o desconto de 40% e o tal arredondamento aos múltiplos de 50 cêntimos fica a €23.50. Aliás, se a antecedência da compra for oito dias ou mais, o desconto sobe para 65%, o que é ainda mais vantajoso.

Mas os descontos possíveis não ficam por aqui, descobriram a Ana e o Carlos: se conseguirem trazer consigo pelo menos mais dois outros amigos, passam a ser considerados um grupo e a ter direito a um desconto adicional de 50%. Quer dizer, se todos os quatro (ou mais) tivessem decidido comprar o bilhete ao mesmo tempo, com cinco dias de antecedência, o desconto sobre 23.50 seria 11.75, pelo que, feitas as contas, e considerados os malfadados arredondamentos, a viagem para cada um dos membros do grupo ficaria por €12.00. Claro que se comprarem com antecedência de oito dias ou mais, o preço será ainda mais baixo.

Este regime de descontos aplica-se a todas as viagens de longo curso. Por isso, se a Ana e o Carlos acabarem por decidir ir fazer o curso de verão a outro sítio, por exemplo, à Universidade Nova de Lisboa, terão de fazer contas análogas, começando com o preço da viagem Porto-Lisboa.

Claro que, em vez de fazer as contas à mão, o melhor é escrever um programa que trate disso.

Tarefa

Escreva um programa que dado o preço normal, a antecedência da compra e o tamanho do grupo (todos jovens), determine o preço de cada bilhete, expresso por um número inteiro de cêntimos.

Input

Uma linha com três inteiros, representando, respetivamente, o preço normal da viagem (em cêntimos), a antecedência da compra e o tamanho do grupo de viajantes, todos jovens.

Output

Uma linha com um inteiro que representa o preço de cada bilhete, expresso por um número inteiro de cêntimos (que há de ser um múltiplo de 50, por causa daquela regra dos arredondamentos).

Exemplo 1

Input

```
5150 5 4
```

Output

```
1200
```

Exemplo 2

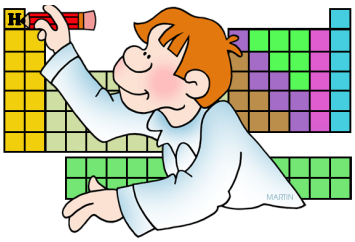
Input

```
3400 10 2
```

Output

```
900
```

Química à Toa



Quando o Rui estava a estudar Química, escrevendo fórmulas de compostos no caderno (como $CaCO_3$, Na_2CO_3 e H_2SO_4), apareceu a irmã mais nova, que ficou fascinada com aquelas sequências de letras e algarismos. O pior é que ela não se calava, exigindo que ele lhe explicasse como dividia uma sequência grande em pedacinhos. Desesperado, o Rui enunciou as regras:

- Cada sequência de letras que começa com uma letra grande (a irmã não conhecia a palavra maiúscula) representa um elemento;
- Se houver algum número a seguir ao elemento, esse é o número de átomos do elemento; se não houver, o número de átomos do elemento é 1.

Depois, exemplificou com a fórmula do carbonato de cálcio, $CaCO_3$, que tem um átomo de cálcio (Ca), um átomo de carbono (C) e três átomos de oxigénio (O).

A irmã ficou muito contente e queria aprender mais Química. Mas ele convenceu-a a praticar. Para poder regressar ao estudo rapidamente, decidiu escrever "fórmulas químicas" à toa e até inventou símbolos químicos com 1, 2, 3, 4 e 5 letras. Por exemplo, deu-lhe a fórmula $Ai_{19}Xpto_5Uus$ (que tem tamanho 12) para decompor.

Tarefa

Escreva um programa que, dada uma fórmula química inventada pelo Rui e terminada por "**.**", apresenta os elementos que ocorrem na fórmula e o número de átomos de cada um.

Input

Uma linha com uma fórmula química (inventada pelo Rui) terminada por "**.**". A fórmula é uma sequência de n letras e algarismos, que começa com uma letra maiúscula e que não tem o algarismo zero imediatamente a seguir a uma letra, nem uma letra minúscula imediatamente a seguir a um algarismo. Nenhum elemento ocorre mais do que uma vez na fórmula.

Restrições

$1 \leq n \leq 1\,000$ Tamanho da fórmula

Output

O output tem uma linha por cada elemento que ocorre na fórmula. Cada linha tem um elemento, um espaço, o número de átomos desse elemento e uma mudança de linha. Os elementos têm de aparecer pela ordem em que ocorrem na fórmula.

Exemplo 1

Input

```
CaCO3.
```

Output

```
Ca 1
C 1
O 3
```

Exemplo 2

Input

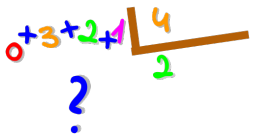
```
Ai19Xpto5Uus.
```

Output

```
Ai 19
Xpto 5
Uus 1
```

Jogo na Abobadeira

Realizam-se este ano na Abobadeira as finais nacionais das Olimpíadas da Numerologia, cujo evento principal é o Jogo do Resto. Nós vamos apoiar com a programação do quadro de resultados e precisamos da vossa colaboração. Ainda não conhecem o Jogo do Resto? É muito simples:



1. joga-se com N jogadores, do jogador 0 ao $N-1$;
2. cada jogo tem M jogadas;
3. em cada jogada, cada jogador regista um número inteiro de 0 a $N-1$;
4. somam-se os números registados por todos os jogadores, S ;
5. calcula-se o resto R da divisão inteira de S por N, com $0 \leq R \leq N-1$;
6. o jogador com o número R contabiliza mais 1 ponto, os restantes mantêm os que têm;
7. no final das M jogadas ganha quem totalizar mais pontos.

Tarefa

Agora vamos a escrever um programa para ler as várias jogadas, contabilizar os pontos dos jogadores, e indicar o vencedor.

Input

Os dados são iniciados uma linha com os inteiros N - número de jogadores, com $N < 100$, M, número de jogadas, com $N < 100$. Seguem M linhas contendo as jogadas. Cada linha com uma jogada contém N inteiros $J_0, J_1, \ldots J_i, \ldots J_{N-1}$, com $0 \leq i < N$ e $0 \leq J_i < N$.

Output

O resultado consiste em 2 grupos de linhas, cada qual com apenas um inteiro. As linhas do primeiro grupo têm o total de pontos de cada jogador, por ordem crescente do seu número. Entre o primeiro e o segundo grupo de linhas há uma linha em branco. As linhas do segundo grupo apresentam o número dos jogadores que totalizaram o maior número pontos e por isso venceram o jogo, também por ordem crescente do seu número.

Exemplo 1

Input

```
2 3
0 0
```

```
0 1
1 0
1 1
```

Output

```
2
2

0
1
```

Exemplo 2

Input

```
3 5
0 1 2
2 1 0
0 0 0
1 1 2
0 2 0
```

Output

```
3
1
1

0
```

Compras, só em moedas

Na loja do Senhor Moeda, não se aceitam notas para pagar as compras realizadas pelos clientes, nem se dá troco em notas. Tudo é realizado com as moedas de 2 e 1 euros e as moedas de 50, 20, 10, 5, 2 e 1 cêntimos. Na loja existe um stock ilimitado de moedas e, já agora, de artigos para venda.

De forma a manter os clientes satisfeitos, sempre que o Sr. Moeda dá troco, dá o menor número possível de moedas.

Tarefa

Escreva um programa que, dado o valor da compra e as moedas que o cliente deu para pagar a compra, indica as moedas dadas no troco, caso seja possível realizar a compra (ou seja, quando o valor dado pelo cliente é superior ou igual ao valor da compra). No caso contrário, o programa indica as moedas dadas pelo cliente.

Por exemplo, se o valor a pagar for 5 euros e 25 cêntimos e o cliente der 1 moeda de 2 euros, 3 moedas de 1 euro e 2 moedas de 50 cêntimos, o troco seria composto por 1 moeda de 50 cêntimos, 1 moeda de 20 cêntimos e 1 moeda de 5 cêntimos.

Input

Existem sempre 2 linhas.

A primeira linha tem dois inteiros, E e C, que indicam que o valor da compra é E euros e C cêntimos.

A segunda linha tem 8 inteiros não negativos, que indicam o número de moedas de cada tipo dadas pelo cliente para efetuar o pagamento. Estes números são dados da moeda com maior valor à moeda com menor valor. Assim, o primeiro inteiro é o número de moedas de 2 euros dadas pelo cliente e o último inteiro é o número de moedas de 1 cêntimo dadas pelo cliente.

Restrições

$0 \leq E \leq 1\,000$	Euros do valor da compra
$0 \leq C \leq 99$	Cêntimos do valor da compra

Output

Uma única linha com 8 inteiros não negativos, que indicam o número de moedas de cada tipo que serão entregues ao cliente. Caso o valor dado pelo cliente seja superior ou igual ao valor da compra, essas moedas são o troco dado pelo Sr. Moeda. No caso contrário, essas moedas foram as dadas pelo cliente ao Sr. Moeda. Tal como na entrada, o primeiro número corresponde à moeda de 2 euros, o segundo número corresponde à moeda de 1 euro, e assim sucessivamente, respeitando a ordem decrescente do valor facial das moedas.

Exemplo 1 – Venda efetuada

Input

```
5 25
1 3 2 0 0 0 0 0
```

Output

```
0 0 1 1 0 1 0 0
```

Exemplo 2 – Venda efetuada

Input

```
5 25
0 3 0 1 0 1 0 200
```

Output

```
0 0 0 0 0 0 0 0
```

Exemplo 3 – Venda não efetuada

Input

```
5 25
0 3 0 1 0 1 0 100
```

Output

```
0 3 0 1 0 1 0 100
```

Água mole em pedra dura

O provérbio "água mole me pedra dura, tanto bate até que fura" não quer dizer que alguém vai explodir se estiver continuamente sob pressão, mas sim que, com persistência, se pode vencer as dificuldades. É natural que o esforço necessário seja maior nalguns casos do que noutros.

Tarefa

Escrever um programa que analisa uma sequência de `0`'s e `1`'s e conta quantas vezes se conseguiu ultrapassar o nível de dificuldade, considerando vários cenários. Uma sequência de `1`'s adjacentes representa uma sequência de tentativas consecutivas e considera-se que se ultrapassou o nível de dificuldade do cenário se o número máximo de `1`'s consecutivos exceder o seu nível de dificuldade.

Input

Uma linha com uma sequência de `0`'s e `1`'s (sem espaços). Segue-se uma linha com um inteiro positivo n que representa o número de cenários a analisar e n linhas, cada uma com um inteiro positivo que indica o nível de dificuldade do cenário correspondente. A sequência de `0`'s e `1`'s pode ser muito grande e n também.

Restrições

$1 \leq t < 100\,000$	Número máximo de tentativas consecutivas numa sequência
$1 \leq n < 10\,000$	Número de cenários a analisar

Output

Uma linha com um inteiro não negativo que indica o número cenários em que a persistência (i.e., número máximo de `1`'s consecutivos) foi superior ao nível de dificuldade do cenário.

Exemplo

Input

```
1111000011010001111100011111011110111111111011110111111101
7
2
6
3
11
7
9
```

```
8
```

Output

```
5
```

ToPAS Parking

Normalmente, nos parques de estacionamento dos centros comerciais, os lugares de estacionamento estão organizados em filas paralelas, duas a duas, com uma faixa de circulação entre cada par de filas. Nos parques mais sofisticados, existe um sistema de semáforos para facilitar a localização de lugares livres pelos condutores que querem estacionar. É muito simples: coloca-se um semáforo por cima de cada lugar de estacionamento. Quando o lugar está livre, o semáforo fica verde; quando está ocupado, fica vermelho.

O parque de estacionamento do futuro TopaShopping de Loulé usará uma organização de lugares revolucionária, suprimindo as faixas de circulação. Quer dizer, a placa de estacionamento estará dividida em células de estacionamento retangulares, todas contíguas, cada uma com um semáforo. É verdade que, havendo condutores desrespeitosos, alguns carros podem ficar bloqueados, isto é, impossibilitados de sair do parque. Além disso, pode acontecer que certas zonas do parque fiquem vazias mas inacessíveis, porque há carros estacionados rodeando essa zona completamente. Para facilitar a vida aos clientes que querem estacionar, ajudando-os a encontrar lugar para o carro, vão usar-se semáforos com três cores — verde, vermelho e amarelo — funcionando da seguinte forma:

Em cada momento, cada uma das células de estacionamento estará ocupada, livre, bloqueante ou isolada. Estará ocupada, com o semáforo vermelho, se lá estiver um carro parado; estará livre, com o semáforo verde, se estiver vazia mas não bloqueante nem isolada; estará bloqueante, com o semáforo amarelo, se estiver vazia mas, se estivesse ocupada, pelo menos uma das células adjacentes que estejam ocupadas ficaria com o respetivo carro bloqueado; estará isolada, com o semáforo apagado, se estiver livre mas inacessível a partir do exterior do parque.

Note que um carro pode estar bloqueado mesmo sem estar entalado entre quatro carros (um à frente, outro atrás, outro à esquerda e outro à direita). Na verdade estar bloqueado significa que os lugares adjacentes à frente, atrás, à esquerda e à direita estão todos ocupados ou isolados.

Note ainda que, não sendo o parque de estacionamento vedado, os carros que estacionarem nas filas das pontas nunca ficarão bloqueados. Além disso, pode-se entrar na superfície do parque por qualquer uma das células de estacionamento das filas das pontas, desde que estejam vazias. Logo, essas células nunca ficarão isoladas.

Tarefa

Por favor, escreva um programa que leia da consola um ficheiro descrevendo o parque e os lugares ocupados e que escreva na consola o plano dos semáforos. O parque é retangular, de dimensões W e H, sendo W medida do lado horizontal e H a medida do lado vertical, e cada lugar de estacionamento é representado por um par de coordenadas x, y, com $0 \leq x < W$ e $0 \leq y < H$. O plano dos semáforos é uma matriz com H linhas e W colunas, onde cada posição terá a cor do semáforo do lugar correspondente, com a seguinte codificação: apagado, **0**; verde, **1**; vermelho, **2**; amarelo, **3**.

Input

A primeira linha do input contém os valores de W e H, por esta ordem. As restantes linhas, em número indeterminado, contêm cada uma o par de coordenadas x, y de cada lugar de estacionamento que está correntemente ocupado, sem repetições. O fim dos dados é assinalado por uma linha contendo duas vezes o número `-1`.

Restrições

$1 \leq W \leq 100$ Número de colunas
$1 \leq H \leq 100$ Número de linhas

Output

O output é a matriz com o plano dos semáforos, escrita linha a linha, com todos os números justapostos.

Exemplo

Input

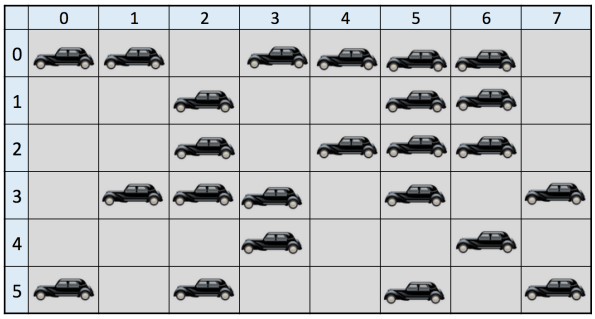
```
8 6
0 0
1 0
3 0
4 0
5 0
6 0
2 1
5 1
6 1
2 2
4 2
5 2
6 2
1 3
2 3
```

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

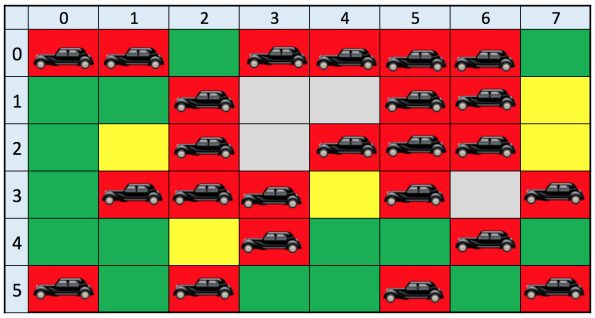
Output

```
22122221
11200223
13202223
12223202
11321121
21211212
```

O input corresponde à seguinte configuração, ainda sem semáforos:



O output já assinala as cores, ficando a cinzento os lugares onde o semáforo está apagado:



Note que o sistema é prático mas não é perfeito. Quer dizer, não garante que um carro que estacione num lugar com luz verde não impeça outro carro de sair do parque. O sistema apenas assinala que um carro que estacionasse num lugar com luz amarela bloquearia pelo menos um dos carros estacionados nos lugares adjacentes.