Maratona de Programação
2000
Caderno de Problemas

Hotel

```
Arquivo:  hotel.c ou hotel.pas
Entrada:  hotel.in
Saída:  hotel.out
```

O dono de um hotel está desconfiando do gerente de seu hotel, que tem reclamado constantemente de problemas de superlotação ao mesmo tempo que repassa ao dono muito menos dinheiro do que este esperava. Entretanto, as anotações do gerente não permitem ao dono saber em que momentos seu hotel estava de fato superlotado.
Para cada hóspede (todos os quartos são simples, e portanto o número de quartos é igual ao número de hóspedes), o gerente anota o dia da chegada e o dia da partida deste. Sua tarefa é determinar em que dias do período o hotel esteve com o maior número de quartos ocupados, e quantos são estes quartos.

Entrada

São dados vários conjuntos de entradas. Cada conjunto de entradas é composto por diversas hospedagens. Na primeira linha de cada conjunto é dado o número de hóspedes no período. Para cada hospedagem é dado o dia de entrada e o dia de saída (os dias são numerados de 1 a 365, a partir de primeiro de janeiro). O hotel fecha no dia 365 (31 de dezembro), não permitindo, portanto, hospedagens para o Reveillon (desconsidere anos bissextos). Um conjunto com 0 hospedagens indica que acabaram os dados.

Saída

Para cada conjunto de hospedagens seu programa deverá imprimir a mensagem `Instancia` com o número da instância e, na linha seguinte a mensagem `O hotel atingiu a lotacao maxima de x hospedes nos dias:` onde `x` é o número máximo de hóspedes no hotel no período. Na linha seguinte deverão ser listados os dias em que o hotel atingiu lotação máxima. As respostas de cada instância devem ser separadas por uma linha em branco.

Exemplo

Entrada

```
3
3 26
23 46
18 22
2
1 20
18 24
0
```

Saída

```
Instancia 1:
O hotel atingiu a lotacao maxima de 2 hospedes nos dias:
18 19 20 21 23 24 25

Instancia 2:
O hotel atingiu a lotacao maxima de 2 hospedes nos dias:
18 19
```

O exemplo acima estava errado na prova aplicada.

PPA

```
Arquivo:  ppa.c ou ppa.pas
Entrada:  ppa.in
Saída:  ppa.out
```

Dizemos que uma seqüência $a_1, a_2, \ldots a_n$ de termos é uma ppa se a seqüência $a_2 - a_1, a_3 - a_2, \ldots, a_n - a_{n-1}$ forma uma pa. Lembre que uma pa (progressão aritmética) é uma seqüência em que, para todo par de elementos consecutivos, a diferença entre eles, chamada **razão** é constante.
Sua tarefa vai ser fazer um programa que recebe como entrada várias seqüências e determina se cada uma delas é ou não uma ppa.

A entrada

São dadas várias seqüências. Cada seqüência é dada por uma linha contendo o número $n \geq 3$ de termos da seqüência, e, na linha seguinte, os n termos. A entrada termina com um zero.

A saída

Para cada seqüência você deverá imprimir uma linha identificando a mesma (`Sequencia 1`, etc). A seguir deverá vir uma mensagem `A sequencia nao e uma ppa` ou `A sequencia e uma ppa de razao r`, conforme o caso, onde r é a razão da pa.

Exemplo

Entrada

```
12
1 3 15 7 19 11 13 15 17 19 21 23
3
3 7 15
4
2 4 8 16
0
```

Saída

```
Sequencia 1
A sequencia nao e uma ppa

Sequencia 2
A sequencia e uma ppa de razao 4

Sequencia 3
A sequencia nao e uma ppa
```

Padrão

```
Arquivo:  padrao.c ou padrao.pas
Entrada:  padrao.in
Saída:  padrao.out
```

Busca de padrões em texto é uma das tarefas mais realizadas e interessantes em computação. Esta busca, entretanto, é unidimensional, e não se aplica diretamente a outras aplicações interessantes. Neste exercício sua tarefa será fazer um programa de busca de padrões em imagens bidimensionais.
Para simplificar a tarefa, vamos supor que a imagem é em branco e preto. Ou seja, você pode imaginar a imagem dada por uma matriz de zeros e uns. O padrão procurado também é uma matriz de 0s e 1s. Sua tarefa é verificar se o padrão ocorre ou não na matriz, e, no caso em que ocorre, indicar a posição inicial do padrão na imagem (canto superior esquerdo).

Entrada

São dadas várias imagens e padrões. Para cada par, na primeira linha vem o número de linhas (você pode supor que é menor ou igual a 100) e colunas da imagem (também menor ou igual a 100), seguido pela imagem (linhas de zeros e uns). A seguir, vem o número de linhas e colunas do padrão (também menores ou iguais a 100), seguido por ele, no mesmo formato. A entrada termina por um zero.

Saída

Para cada par você deverá imprimir a palavra `Instancia i` onde i indica o número da instância tratada. Na linha seguinte você deverá imprimir a mensagem `O padrao nao ocorre na imagem` ou `O padrao ocorre na imagem na posicao l c`, dependendo do caso que se aplique, onde l é a linha e c a coluna onde ocorre o padrão. Caso o padrão ocorra mais de uma vez na imagem seu programa deverá devolver a ocorrência mais próxima do canto superior esquerdo. As instâncias devem ser separadas por uma linha em branco.

Exemplo

Entrada

```
5 5
11010
01010
11011
01010
11001
2 2
10
```

```
10
5 5
11010
01010
11011
01010
11001
2 2
11
11
0
```

Saída

```
Instancia 1
O padrao ocorre na imagem na posicao 1 2

Instancia 2
O padrao nao ocorre na imagem
```

Rato

```
Arquivo:  rato.pas e rato.c
Entrada:  rato.in
Saída:  rato.out
```

É muito comum ver experiências em laboratório em que ratos ou cobaias devem encontrar a saída de labirintos, medindo, com isso, suas habilidades, inteligência, etc.
Os ratos estão cada vez mais espertos ultimamente e resolveram aproveitar os computadores disponíveis no laboratório para pedir ajuda na tarefa de encontrar a saída. As informações do labirinto são dadas pelo rato, assim como as coordenadas de sua posição inicial e da saída. Eles desejam, então, obter instruções precisas de como proceder para achar uma saída do labirinto. Essas instruções deverão ser dadas usando as letras `D, E, C, B` indicando, respectivamente, direita, esquerda, cima, baixo, seguida pelo número de passos que o rato deverá dar naquela direção. Caso não haja saída do labirinto, seu programa deve indicar isso, a fim de que o rato fique imóvel olhando para o cientista.
Sua tarefa é, então escrever um programa de auxílio para o rato, que recebe várias configurações de labirinto, seguido das posições do rato e da saída, e determina, para cada uma delas como o rato deve proceder para achar a saída, caso isso seja possível.

Entrada

São dados vários labirintos. Cada labirinto é dado por uma primeira linha em que é dado o número de linhas $m \leq 100$ e colunas $n \leq 100$ no labirinto. A seguir são dadas m linhas, cada uma com n números. O número 0 indica uma posição vazia e o número 1 indica uma obstrução no labirinto. A seguir é dada a posição do rato e da saída, através da linhas e colunas destes (as linhas e colunas são numeradas a partir de 1). A entrada termina com um zero.

Saída

Para cada labirinto você deverá imprimir a palavra `Labirinto i` onde i indica o número da instância tratada. A seguir seu programa deverá imprimir a mensagem `Nao e possivel sair do labirinto`, ou dar uma seqüência de passos dados por uma letra e um número. Os passos devem ser seguidos pela mensagem `Numero de movimentos = k`, onde k deve ser o número total de passos dado pelo rato. A letra poderá ser `D, E, C, B` indicando, respectivamente, direita, esquerda, cima, baixo, e o número indica o número de passos a serem dados naquela direção. Não serão aceitas respostas com duas letras iguais consecutivas como, por exemplo,

```
B 1
B 1
```

A informação dos labirintos devem ser separados por uma linha em branco.

Exemplo

Entrada

```
4 5
0 0 1 0 1
0 1 0 0 1
0 0 0 1 1
1 1 0 0 0
1 1
1 4
3 4
0 0 1 0
0 1 0 0
1 0 0 0
1 1
3 4
0
```

Saída

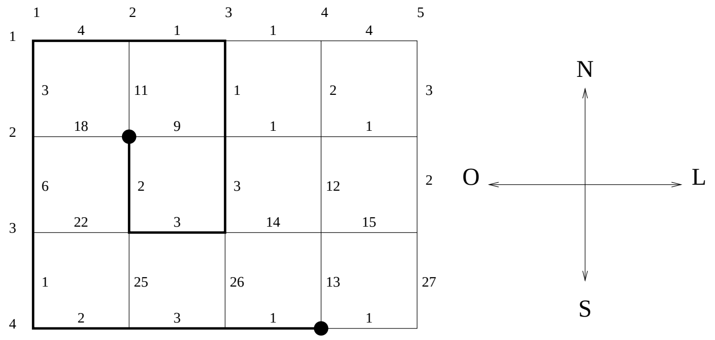
```
Labirinto 1
B 2
D 2
C 1
D 1
C 1
Numero de movimentos = 7

Labirinto 2
Nao e possivel sair do labirinto
```

Newyork

`Arquivo: newyork.c ou newyork.pas`
`Entrada: newyork.in`
`Saída: newyork.out`

Em uma cidade planejada, em que as ruas são verticais ou horizontais (como Manhatan, em Nova York), um cidadão deseja fazer um planejamento de seu caminho de tal forma que ele nunca espere a abertura de um semáforo.
A cidade pode ser vista como uma grade retangular de altura m e largura n com todas as ruas de duas mãos e com um semáforo em cada cruzamento (inclusive nos cruzamentos "em T" e "em cotovelo"). Com isso, esta cidade tem $(m-1)n+(n-1)m$ trechos de rua e mn cruzamentos. A figura abaixo mostra uma grade onde $m=4$ e $n=5$.



Os cruzamentos são indicadas por um par de números que indicam as ruas que formam este cruzamento, sendo que o primeiro número indica a rua na direção Oeste-Leste (esquerda para direita) e o segundo indica a rua na direção Norte-Sul. A fim de poder se movimentar sem parar em semáforos, este cidadão recolheu dados para cada um dos trechos de rua da cidade com o tempo que ele deve percorrer este trecho até o cruzamento de forma a encontrar o semáforo aberto. Com base nesses dados ele deseja agora, para cada rota que pretende fazer na cidade, achar o caminho mais rápido entre dois cruzamentos dados, sem ter de esperar a abertura de semáforos no caminho. A figura acima mostra, através de um traço mais grosso, o caminho mais rádido entre os cruzamentos $4,4$ e $2,2$, que gasta 30 minutos.

A entrada

Na primeira linha é dada a altura m e a largura n da grade. Nas $m+n$ linhas seguintes são dados os tempos dos trechos de rua da cidade. Em cada uma das primeiras m linhas, as distâncias dos trechos na direção Oeste-Leste (trechos da esquerda para a direita começando da rua 1 até a rua m). Nas n linhas seguintes, as distâncias dos trechos na direção Norte-Sul. Na linha seguinte vem o número k de consultas. Nas k linhas que se seguem aparecem dois pares de números indicando o cruzamento de origem e cruzamento destino de cada rota.

A saída

Para cada consulta você deve imprimir uma identificação da consulta e, na linha seguinte, a mensagem `Caminho mais rapido entre os cruzamentos x,y e z,w = t`, onde x, y e z, w indicam os dois cruzamentos e t indica o tempo gasto no caminho mais rápido entre os dois cruzamentos. As consultas devem ser separadas por uma linha em branco.

Exemplo

Entrada

```
4 5
 4 1  1  4
18 9  1  1
22 3 14 15
 2 3  1  1
 3 6  1
11 2 25
1  3 26
2 12 13
3  2 27
1
4 4  2 2
```

Saída

```
Consulta 1
Caminho mais rapido entre os cruzamentos 4,4 e 2,2 = 30
```

Amigos

```
Arquivo:  amigos.c ou amigos.pas
Entrada:  amigos.in
Saída:  amigos.out
```

Uma turma do BCC era reconhecida por ser muito "paneleira". Dizem que existiam grupos dentro da turma bastante fechados, conhecidos como "alas", que não conversavam com pessoas de outros grupos e coisas assim. Havia uma grande briga na turma para saber quem era a maior ala, mas nunca se conseguiu chegar a um consenso.
A fim de tirar a limpo esta situação, resolvemos fazer um programa para determinar a maior "panela" da turma. Uma "panela" é um grupo de alunos em que todos são amigos entre si, e não existe nenhum aluno fora da panela que seja amigo de todos da panela.
Sua tarefa é fazer um programa que recebe os pares de amigos da turma e imprime o tamanho da maior panela da turma.

Entrada

São dadas várias turmas. Para cada turma, na primeira linha é dado o número $n \leq 50$ de alunos na turma e o número m de amizades na turma. Nas m linhas seguintes são dados os pares de amigos. A entrada termina com um número 0.

Saída

Para cada turma seu programa deverá imprimir a mensagem `Turma i` identificando o número da turma, e na linha seguinte uma panela de tamanho máximo da turma, dada pelos alunos que a compõe. As turmas deverão ser separadas por uma linha em branco.

Exemplo

Entrada

```
10 13
1 2
1 3
2 3
3 7
3 9
4 5
4 6
4 7
5 6
5 7
6 7
```

```
8 9
9 10
5 5
1 4
2 3
1 3
5 3
1 5
0
```

Saída

```
Turma 1
4 5 6 7

Turma 2
1 3 5
```

Steiner

```
Arquivo:  steiner.c ou steiner.pas
Entrada:  steiner.in
Saída:  steiner.out
```

Um gerente de festas está preocupado com uma questão relacionada a um problema conhecido em matemática como "Steiner triple systems" (na verdade um caso particular do problema tem este nome). As festas deste buffet ocorrem de uma forma diferente, em vários dias. Em cada dia são chamados k convidados entre os n participantes. A restrição para formação das mesas é que um mesmo grupo de p pessoas não seja convidado em dois dias da festa.

Assim, por exemplo, se $n = 7$, $k = 3$ e $p = 2$, devemos organizar as 7 pessoas em mesas de 3 lugares de forma que cada par de pessoas apareça em uma mesma mesa exatamente uma vez. Uma solução possível para a instância dada acima seria:

Mesa 1: 1 2 3	Mesa 2: 1 4 5	Mesa 3: 1 6 7
Mesa 4: 2 4 7	Mesa 5: 2 5 6	Mesa 6: 3 4 6
Mesa 7: 3 5 7		

Verifique que todos os 21 pares de números estão juntos em alguma das mesas, e nenhum dos pares aparece repetido em duas mesas diferentes.

Na verdade o gerente já contratou uma empresa para fazer um software para resolver o problema, só que não está confiando muito na solução. Este problema será fazer um programa que verifica se a solução apresentada satisfaz a condição acima.

Sua tarefa será escrever um programa que recebe $n \leq 20$, $k \leq 7$, $p \leq 5$ e uma seqüência de k-tuplas, e verifica se cada uma das p-tuplas estão presentes em exatamente uma única k-tupla (mesa).

Entrada

A primeira linha do arquivo de entrada contém três números: o número n de pessoas, o tamanho k das mesas e p. A partir da segunda linha, cada linha contém uma k-tupla com elementos distintos, com uma disposição de mesa, até o fim do arquivo (`end-of-file`).

Saída

A saída consiste de uma mensagem:

`A disposicao das mesas atende aos requisitos`

ou

`A disposicao das mesas nao atende aos requisitos`

Exemplo 1

Entrada

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

Saída

```
A disposição das mesas atende aos requisitos
```

Exemplo 2

Entrada

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

Saída

```
A disposição das mesas nao atende aos requisitos
```