Aquecimento para Maratona de Programação - 2007

Universidade de São Paulo

Caderno de Problemas

Departamento de Ciência da Computação IME-USP

Sábado, 26 de maio de 2007

Problema A: Harmonia!

`Arquivo: harmonia.[c|cpp|java]`

Depois de diversos anos de pesquisa, em que dezenas de milhares de músicas foram estudadas, Pil Gruberoff, pesquisador da Universidade da Transilvânia do Sul [1] descobriu que o sucesso de uma música estava intimamente ligado ao que ele denominou de **harmonia melódica**. Gruberoff anotou as freqüências das notas musicais e ao verificar as diferenças entre notas musicais consecutivas nas músicas conseguiu provar que seu sucesso era maior quando essa diferença era *menor* por toda a música. Seus estudos tiveram enorme sucesso e hoje Gruberoff trabalha como consultor de várias bandas de sucesso na Transilvânia, Azerbaijão, Uzbequistão e outras regiões européias. Ele até se mudou para New Jersey e procura convencer os americanos de suas descobertas. Sua tarefa neste problema é ajudar o Professor Gruberoff na tarefa de determinar a harmonia melódica de uma música.

O método usado pelo Professor Gruberoff é o seguinte: Dada uma seqüência de freqüências de notas músicais. Determine a maior diferença entre duas notas consecutivas. Tal diferença é exibida aos músicos para que sejam feitos novos arranjos. Afinal de contas o Professor Gruberoff só entende de melodias de músicas Punk e Hardcore!

Entrada

A entrada é composta de diversas instâncias. Cada instância começa com um inteiro n, onde $2 \leq n \leq 1000$, indicando a quantidade de inteiros da seqüência que deve ser analisada. A próxima linha contém uma seqüência de n inteiros: $a_1, a_2, ..., a_n$, onde $-2^{30} \leq a_i \leq 2^{30}$ para $1 \leq i \leq n$. O programa deve parar de processar a entrada quando $n = 0$.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte você deve imprimir a maior diferença entre duas notas musicais consecutivas.

Após cada instância, seu programa deve imprimir uma linha em branco.

Exemplo de entrada

```
4
1 2 3 4 5
3
3 3 4 5
0
```

Exemplo de saída

```
Instancia 1
1

Instancia 2
1
```

[1] Importante região européia produtora de vodka

Problema B: Quadrados latinos

`Arquivo: latino.[c|cpp|java]`

Um quadrado quadrado latino é uma matriz $n \times n$, tal que cada entrada (símbolos), ocorre no máximo uma vez em cada linha e em cada coluna.

O estudo de quadrados latinos remonta a períodos antigos, e suas aplicações eram inúmeras. Evidências arqueológicas recentemente encontradas nas proximidades de Roma relatam do famoso General Armandus que costuma utilizar quadrados latinos para codificar as mensagens enviadas aos comandantes dos seus exércitos. Gal. Armandus, famoso por correr diariamente mais de 40km, mandava seus mensageiros correndo através dos campos de batalha com a mensagem codificada. Os quadrados latinos eram combinados previamente com os comandantes, e trocados todos os dias, quando o Gal. Armandus dava sua corridinha e informava aos comandantes o quadrado que seria usado no dia seguinte. Na manhã seguinte o mensageiro de Gal. Armandus levava diversos quadrados e somente um destes era um quadrado latino.

Os arqueológos encontraram muitos quadrados, e gostariam da sua ajuda para determinar quais destes quadrados são quadrados latinos.

Entrada

A entrada é composta de diversas instâncias. Cada instância começa com um inteiro n, onde $1 \leq n \leq 100$, indicando as dimensões do quadrado que deve ser analisado. Nas próximas n linhas, são dados n símbolos. Cada símbolo é representado por um inteiro no intervalo de 1 a n. O programa deve parar de processar a entrada quando $n = 0$.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte, seu programa deve imprimir `sim` se o quadrado é um quadrado latino, e `nao` caso contrário.

Após cada instância, seu programa deve imprimir uma linha em branco.

Exemplo de entrada

```
3
1 2 3
3 1 2
2 3 1
4
1 2 1 3
2 3 4 1
3 1 2 4
4 4 2 1
0
```

Exemplo de saída

```
Instancia 1
sim

Instancia 2
nao
```

Problema C: Fortuna × Demasi: O confronto do século!

`Arquivo: desafio.[c|cpp|java]`

Fortuna e Demasi gostam de competir. Qualquer jogo ou esporte é motivo para que uma disputa se realize. Até mesmo o enfadonho jogo de damas é motivo para discórdia entre os dois. Como cada um mora em um país diferente, atualmente Fortuna mora em *New York - Brazil*, e Demasi mora no *Rio de Janeiro - Brasil*, dificilmente eles conseguem jogar ao vivo. A saída é disputar a partida usando a Internet. A cada email um movimento é realizado. Entretanto, como os dois são desconfiados (e ao mesmo tempo começaram a tentar roubar o adversário) eles resolveram verificar se o movimento enviado pelo oponente de fato corresponde a um movimento válido do jogo de damas.

Cada casa do tabuleiro é nomeada com um par de coordenadas, começando a contagem do canto esquerdo com as letras para as colunas (horizontal), e números nas linhas (vertical).

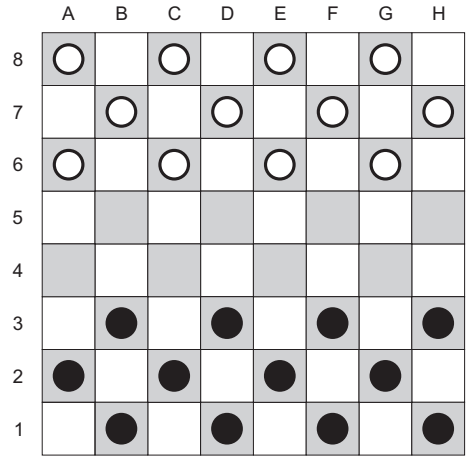

Figura 1: Tabuleiro no estado inicial

As regras do jogo de dama são as seguintes:

1. As peças do Fortuna (Brancas) podem ser movidas para as diagonais adjacentes inferiores. Somente quando a posição estiver vazia.
2. As peças do Demasi (Rubro-negras) podem ser movidas para as diagonais adjacentes superiores. Somente quando a posição estiver vazia.
3. Um jogador pode "comer" uma peça do oponente se e somente se a peça do oponente está em uma das suas diagonais adjacentes (inferior no caso do Fortuna, e superior no caso do Demasi), e existe uma posição vazia na diagonal adjacente da peças que vai ser "comida". *Veja a figura abaixo.*

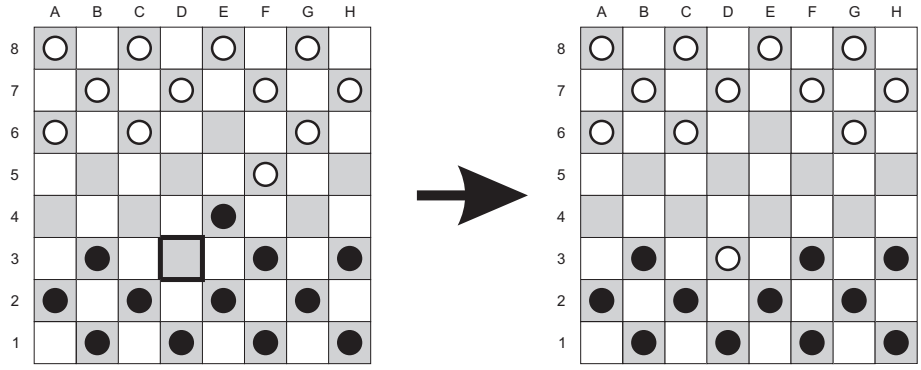

Figura 2: Tabuleiro no estado inicial

4. As peças não podem sair do tabuleiro.

5. Cada jogador pode mover somente as suas peças.

Entrada

A entrada começa com um inteiro t, onde $1 \leq t \leq 1000$, indicando a quantidade de partidas que vão ser analisadas.

Cada partida consiste em um tabuleiro inicial, que é dado em 8 linhas contendo 8 caracteres cada uma. As posições no tabuleiro são representadas da seguinte forma:

`*` indica uma posição vazia.

`B` indica uma peça do Fortuna.

`P` indica uma peça do Demasi.

A próxima linha contém um inteiro n, onde $1 \leq n \leq 1000$, indicando quantas jogadas foram feitas. Cada jogada consiste em um movimento do Fortuna, seguido de um movimento do Demasi. Dessa forma, em cada uma das $2n$ linhas seguintes é informado o movimento feito por um dos jogadores, usando a notação definida anteriormente para representar cada casa do tabuleiro.

Saída

No início de cada instância imprima a linha `Instancia k`, onde k é o número da instância atual. Em seguida, para cada movimento feito imprima o tabuleiro resultante. Se algum movimento quebrar alguma das regras, então a mensagem "`Movimento invalido!`" deve ser impressa ao invés de imprimir o tabuleiro resultante, e mais nenhuma impressão deve ser feita para esta partida.

Após cada instância, seu programa deve imprimir uma linha em branco.

Exemplo de entrada

```
2
B*B*B*B*
*B*B*B*B
B*B*B*B*
********
********
*P*P*P*P
P*P*P*P*
*P*P*P*P
1
C6 D5
D3 E4
B*B*B*B*
*B*B*B*B
B*B*B*B*
********
********
*P*P*P*P
P*P*P*P*
*P*P*P*P
2
C6 D5
D3 F5
E6 F5
E4 D5
```

Exemplo de saída

```
Instancia 1
B*B*B*B*
*B*B*B*B
B***B*B*
***B****
********
*P*P*P*P
P*P*P*P*
*P*P*P*P

B*B*B*B*
*B*B*B*B
B***B*B*
***B****
****P***
*P***P*P
P*P*P*P*
*P*P*P*P

Instancia 2
B*B*B*B*
*B*B*B*B
B***B*B*
***B****
********
*P*P*P*P
P*P*P*P*
*P*P*P*P

Movimento invalido!
```

Problema D: Trigran

`Arquivo: trigran.[c|cpp|java]`

Dono de uma fábrica de brinquedos educativos o Sr. Cardonha teve uma brilhante idéia: o Trigran[2]. Muita gente conhece o **Tangran**, quebra-cabeça chinês formado por sete peças, 5 triângulos, um quadrado e um paralelograma. O nome "tangran" significa "7 tábuas da sabedoria". A idéia do Sr. Cardonha era fazer quebra-cabeças semelhantes apenas com peças triangulares. Para construir um quebra-cabeça o Sr. Cardonha construía inicialmente uma figura de madeira que seria a figura a ser montada com as peças. Com medo de fazer fazer quebra-cabeças muito complicados (o Sr. Cardonha queria vender seu produto em lojas de brinquedos para crianças de 5 a 10 anos) as figuras usadas são todas convexas.

A produção dos quebra-cabeças foi feita na marcenaria do Sr. Magal, amigo de longa data do Sr. Cardonha. Apesar de amigo o Sr. Magal cobrava bem pelo uso de seus equipamentos na hora de cortar a figura em peças triangulares. Tentando minimizar os custos de produção o Sr. Cardonha resolveu recorrer a vocês para fazer um programa que, dada uma peça inicial, determina como ela deve ser recortada em triângulos de forma a minimizar a área a ser cortada pela serra do Sr. Magal.

Entrada

A entrada é composta de diversas instâncias. Cada instância começa com um inteiro n, onde $3 \leq n \leq 100$, indicando o número de vértices da figura. Cada uma das próximas n linhas, contém dois inteiros x e y que representam a localização do vértice, onde $0 \leq x, y \leq 10000$. O programa deve parar de processar a entrada quando $n = 0$.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte você deve imprimir o tamanho total de área a ser serrada. Os números deverão ser impressos com 2 casas decimais de precisão.

Após cada instância, seu programa deve imprimir uma linha em branco.

Exemplo de entrada

```
4
0 3
0 0
2 2
2 3
0
```

Exemplo de saída

```
Instancia 1
2.24
```

[2]Diga rápido: 1 trigran, 2 trigans, 3 trigrans.

Problema E: Casa do terror

`Arquivo: terror.[c|cpp|java]`

O sonho do Prof. Eselkopf era ter uma casa diferente. Ele sonhava com uma casa cheia de cômodos, cômodos cheios de portas, vários corredores, ambientes, becos e esconderijos. Ele vivia imaginando como seria sua casa ideal. O primeiro problema que ele tinha para realizar seu sonho foi resolvido com uma invenção genial: a porta de mão única. Cansado de portas que abriam em ambas as direções, tornando enfadonho o passeio entre cômodos vizinhos de uma casa, Prof. Eselkopf inventou esta porta que funciona como um sistema de válvula, permitindo que uma pessoa a atravesse num sentido mas não possa fazê-lo no sentido contrário. O sucesso da porta "só vai" (nome que deu ao seu fabuloso invento, "Einerichtungnurtür" em sua língua original) no mercado, entretanto, não foi o que Prof. Eselkopf esperava. As pessoas não viam a beleza de andar entre os cômodos seguindo um certo fluxo, e as diversas vantagens práticas da invenção. Inconformado, Prof. Eselkopf resolveu que iria se isolar no castelo de Igelstein que ele herdou da família e trabalhar em solidão em outros inventos fantásticos.

Antes da mudança, entretanto, Prof. Eselkopf trocou todas as portas do castelo por portas "só vai". Como os trabalhadores não entendiam direito o que era aquela porta estranha acabaram instalando-as aleatoriamente, causando certa preocupação ao Prof. Eselkopf. Preocupado ele estudou a planta do castelo e determinou, para cada cômodo, quantas portas de entrada e quantas portas de saída ele devia ter. Portas excedentes, se existissem, seriam emparedadas.

Sua tarefa neste problema é dada a planta do castelo de Igelstein com as portas "só vai" instaladas pelos trabalhadores, e os desejos do Prof. Eselkopf, determinar se é possível emparedar algumas portas de forma a satisfazer o seu projeto.

Entrada

A entrada começa com um inteiro t, onde $1 \leq t \leq 1000$, indicando a quantidade de plantas que serão analisadas.

Cada instância começa com dois inteiros n e m, onde $1 \leq n \leq 100$ e $1 \leq m \leq 10000$, indicando o número de quartos e o número de portas, respectivamente. Cada uma das próximas m linhas, contém dois inteiros u e v, onde $1 \leq u, v \leq n$, indicando que existe uma porta "só vai" do quarto u para o quarto v. Cada uma das próximas n linhas, contém dois inteiros x e y, onde $1 \leq x, y \leq 100$. Seja x_i e y_i os inteiros da i-ésima linha. Então x_i indica o número de portas "só vai" que abrem para dentro da sala i, e y_i indica o número de portas "só vai" da sala i que abrem para outras salas do castelo.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte você deve imprimir `sim` se for possível emparedar algumas portas para que seja respeitado os limites de portas que permitem entrar e sair de cada quarto. Caso contrário você deve imprimir `nao`.

Após cada instância, seu programa deve imprimir uma linha em branco.

Exemplo de Entrada

```
2
4 6
1 2
1 3
2 3
3 2
2 4
3 4
0 1
1 1
2 2
1 0
```

```
4 6
1 2
1 3
2 3
3 2
2 4
3 4
0 1
2 2
2 2
1 0
```

Exemplo de Saída

```
Instancia 1
sim

Instancia 2
nao
```

Problema F: Hoje tem prova da Cris!

`Arquivo: prova.[c|cpp|java]`

A Profa. Cristina é muito exigente. Sua fama de maquiavélica já é conhecida fora da universidade e ela até gosta disso... Mas, parece que agora ela exagerou... Na última prova de sua disciplina a professora exigiu que os alunos formassem fila indiana para entrar na sala de aula. Eles formaram. Quando eles iam entrar ela gritou: "Mas, vocês não estão em ordem lexicográfica crescente!!!???"

Os alunos revoltados formaram uma comissão e foram conversar com a professora dizendo que aquilo já era demais. Para não parecerem intransigentes eles disseram que permitiriam no máximo um número k de trocas de posição entre pessoas consecutivas na fila. A professora gostou da idéia, e colocou como um problema extra da prova. Dada uma seqüência de nomes e um inteiro k devolver a menor seqüência (em ordem lexicográfica) que pode ser obtida a partir da original com no máximo k trocas de elementos vizinhos. Sua tarefa é resolver este exercício para os alunos da Profa. Cristina a fim de que eles consigam entrar na sala e começar a prova.

Entrada

A entrada é composta de diversas instâncias. Cada instância começa com dois inteiros n e m, onde $1 \leq n \leq 100$ e $0 \leq k \leq n$, indicando a quantidade nomes e a quantidade máxima de trocas, respectivamente. A próxima linha contém uma seqüência de n nomes. Cada nome possui tamanho máximo de 20 caracteres, e letras de `'a'` a `'z'`. O programa deve parar de processar a entrada quando $n = m = 0$.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte você deve imprimir a lista de nomes resultante do processo.

Após cada instância, seu programa deve imprimir uma linha em branco.

Exemplo de entrada

```
3 0
wanderley thadeu chegado
3 1
wanderley thadeu chegado
3 2
wanderley thadeu chegado
3 3
wanderley thadeu chegado
0 0
```

Exemplo de saída

```
Instancia 1
wanderley thadeu chegado

Instancia 2
thadeu wanderley chegado

Instancia 3
chegado wanderley thadeu

Instancia 4
chegado thadeu wanderley
```

Problema G: Telefone sem fio

`Arquivo: telefone.[c|cpp|java]`

Toda criança já brincou de "telefone sem fio". Joãozinho inventou uma variação da brincadeira. O grupo de crianças é dividido em dois times. Os times se organizam como na brincadeira original, em que cada um repete o que lhe foi falado para o seguinte, até que o último diz o que chegou até ele. No caso da brincadeira de Joãozinho será falada uma frase com n caracteres (contando letras, espaços, sinais de pontuação, etc). Todos sabem que a frase tem este comprimento. A frase é falada pelo juiz ao primeiro competidor de cada time que a repete para o segundo, e este para o terceiro e assim sucessivamente, até que o último competidor de cada time escreve a frase final (garantindo que n caracteres sejam escritos) e a entrega para o juiz. A equipe vencedora é aquela cuja frase final seja mais próxima da frase original. Para calcular a semelhança entre duas frases de mesmo comprimento você deve contar o número de vezes em que o caractere da frase do time coincide com o caractere da frase original. Ganha o time para o qual o número de coincidências seja máximo. Se os dois times empataram neste critério, a primeira vez que um dos times acertou e o outro errou decide.

Exemplo: Se a frase original foi "O rato roeu a roupa do rei.", o primeiro time escreveu "O ator morreu, garoupa rei." e o segundo time escreveu "O pato moeu garoupa dorlei." O segundo time ganhou pois teve 21 coincidências contra 8 coincidências do primeiro:

O		R	A	T	O		R	O	E	U		A		R	O	U	P	A		D	O		R	E	I	.
O		A	T	O	R		M	O	R	R	E	U	,		G	A	R	O	U	P	A		R	E	I	.
O		P	A	T	O		M	O	E	U		G	A	R	O	U	P	A		D	O	R	L	E	I	.

Assim como os juízes da Maratona de Programação, estes juízes são muito preguiçosos. Logo, pediram para você escrever um programa que resolve este problema.

Entrada

A entrada começa com um inteiro t, onde $1 \leq t \leq 1000$, indicando o número de instâncias que seu programa deve analisar.

Cada instância é composta por três linhas, na primeira a frase correta, na segunda a frase do primeiro time e na terceira a frase do segundo time. Cada frase tem no máximo 100 caracteres, e as frases possuem sempre o mesmo tamanho.

Saída

Para cada instância, você deverá imprimir um identificador `Instancia k`, onde k é o número da instância atual. Na linha seguinte você deve imprimir qual dos times foi o vencedor ou se houve empate.

Após cada instância, seu programa deve imprimir uma linha em branco.

Exemplo de entrada

```
2
O RATO ROEU A ROUPA DO REI.
O ATOR MORREU, GAROUPA REI.
O PATO MOEU GAROUPA DORLEI.
IH EMPATOU!
IH EMPATOU!
IH EMPATOU!
```

Exemplo de saída

```
Instancia 1
time 2

Instancia 2
empate
```