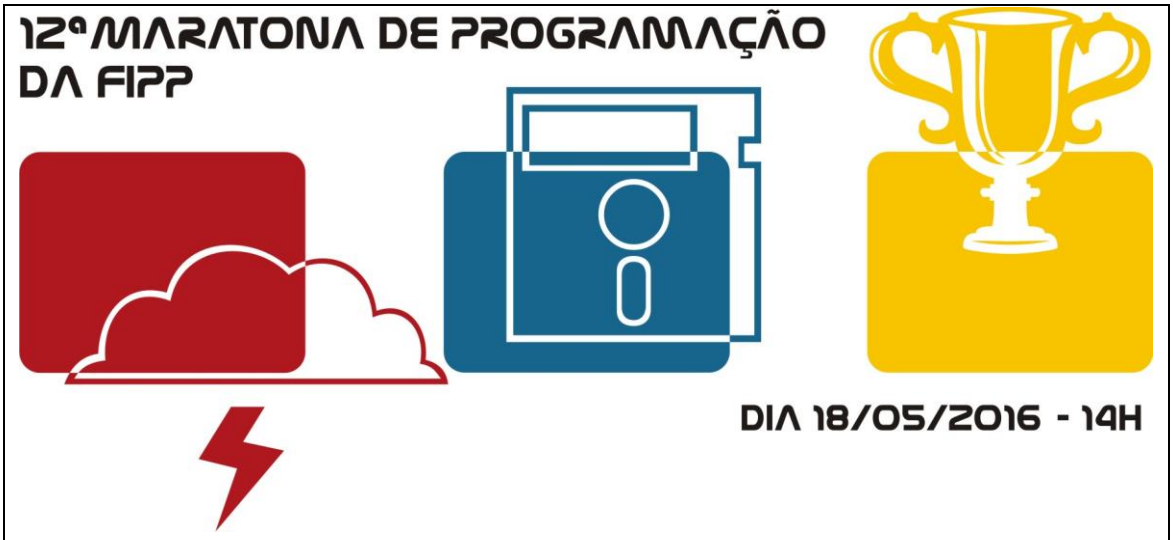

CADERNO DE PROBLEMAS

18/05/2016 – 14h às 17h30

Leia atentamente estas instruções antes de iniciar a resolução dos problemas. Este caderno é composto de 6 problemas, sendo que 2 deles estão descritos em inglês.

1. Não é permitido o uso de material impresso, livros, apostilas e dicionários. Apenas é permitido o uso de lápis, caneta, lapiseira, borracha, régua e papel para rascunho (o papel é fornecido pela comissão organizadora). O acesso à internet é bloqueado durante a realização da prova. A ajuda do ambiente de desenvolvimento como C, C++ ou Java pode ser utilizada.
2. A correção das resoluções dos problemas será automatizada, por meio do sistema de submissão eletrônica BOCA, tendo como base os resultados obtidos a partir de uma série de execuções dos algoritmos submetidos pelas equipes.
3. Siga atentamente as exigências da tarefa quanto ao formato da entrada e da saída do seu algoritmo. Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
4. Os problemas não estão ordenados por ordem de dificuldade neste caderno. A sugestão é de que procure resolver primeiro os problemas mais fáceis.
5. Utilize os nomes indicados nos problemas para nomear os seus arquivos-fonte, de acordo com a linguagem de programação utilizada.
6. Os problemas devem ser resolvidos utilizando o raciocínio entrada-processamento-saída, ou seja, não é necessário armazenar toda a saída para depois exibi-la.

Observações:

Não utilize arquivos para entrada ou saída. Todos os dados devem ser lidos da entrada padrão (teclado) e escritos na saída padrão (tela). Utilize as funções padrão para entrada e saída de dados:

- em C: `scanf`, `getchar`, `printf`, `putchar`;
- em C++: as mesmas de C ou os objetos `cin` e `cout`;
- em Java:
```
Scanner sc = new Scanner(); int i = sc.nextInt();
String s = sc.next(); float f = sc.nextFloat();
System.out.println(); System.out.printf("%d",i);
```

Em C ou C++: use `sprintf(str,"%d",num);` em vez de `itoa(num,str,10);`

Em Java: não utilize `package`.

Problema

A

Países em Guerra (vermelha)

Arquivo fonte: *paises.c*, *paises.cpp* ou *paises.java*

No ano 2050, após diversas tentativas da ONU de manter a paz no mundo, explode a terceira guerra mundial. Segredos industriais, comerciais e militares obrigaram todos os países a utilizar serviços de espionagem extremamente sofisticados, de forma que em cada cidade do mundo há ao menos um espião de cada país. Esses espiões precisam se comunicar com outros espiões, com informantes e mesmo com as suas centrais durante as suas ações. Infelizmente não existe uma forma segura de um espião se comunicar em um período de guerra, então as mensagens são sempre enviadas em código para que somente o destinatário consiga ler a mensagem e entender o seu significado.

Os espiões utilizam o único serviço que funciona no período de guerra: os correios. Cada cidade possui uma agência postal, onde as cartas são enviadas. As cartas podem ser enviadas diretamente ao seu destino ou a outras agências postais, até que a carta chegue à agência postal da cidade de destino, se isso for possível.

Uma agência postal na cidade A pode enviar diretamente uma carta impressa para a agência postal da cidade B se houver um acordo de envio de cartas, que determina o tempo, em horas, que uma carta leva para chegar da cidade A até a cidade B (e não necessariamente o contrário).

Se não houver um acordo entre as agências A e B, a agência A pode tentar enviar a carta a quantas agências forem necessárias para que a carta chegue ao seu destino, se isso for possível.

Algumas agências são interligadas por meios eletrônicos de comunicação, como satélites e fibras ópticas. Antes da guerra, essas ligações atingiam todas as agências, fazendo com que uma carta fosse enviada de forma instantânea, mas durante o período de hostilidades cada país passou a controlar a comunicação eletrônica e uma agência somente pode enviar uma carta à outra agência por meio eletrônico (ou seja, instantaneamente) se ela estiver no mesmo país.

Duas agências, A e B, estão no mesmo país se houver uma forma de uma carta impressa enviada de uma das agências ser entregue na outra agência.

O serviço de espionagem do seu país conseguiu obter o conteúdo de todos os acordos de envios de mensagens existentes no mundo e deseja descobrir o tempo mínimo para se enviar uma carta entre diversos pares de cidades. Você seria capaz de ajudá-lo?

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém dois inteiros separados por um espaço, N ($1 \leq N \leq 500$) e E ($0 \leq E \leq N^2$), indicando o número de cidades (numeradas de 1 a N) e de acordos de envio de mensagens, respectivamente. Seguem-se, então, E linhas, cada uma com três inteiros separados por espaços, X, Y e H ($1 \leq X, Y \leq N$, $1 \leq H \leq 1000$), indicando que existe um acordo para enviar uma carta impressa da cidade X à cidade Y, e que tal carta será entregue em H horas.

Em seguida, haverá uma linha com um inteiro K ($0 \leq K \leq 100$), referente ao número de consultas.

Finalmente, virão *K* linhas, cada uma representando uma consulta e contendo dois inteiros separados por um espaço, *O* e *D* ($1 \leq O, D \leq N$). Você deve determinar o tempo mínimo para se enviar uma carta da cidade *O* à cidade *D*.

O final da entrada é indicado por $N = E = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste da entrada, seu programa deve produzir *K* linhas na saída. A *I*-ésima linha deve conter um inteiro *M*, o tempo mínimo, em horas, para se enviar uma carta na *I*-ésima consulta. Se não houver meio de comunicação entre as cidades da consulta, você deve imprimir "Nao e possivel entregar a carta" (sem acentos).

Imprima uma linha em branco após cada caso de teste.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**
4 5 1 2 5 2 1 10 3 4 8 4 3 7 2 3 6 5 1 2 1 3 1 4 4 3 4 1 3 3 1 2 10 2 3 1 3 2 1 3 1 3 3 1 3 2 0 0	0 6 6 0 Nao e possivel entregar a carta 10 Nao e possivel entregar a carta 0

Problema
B

Intercursos (amarela)

Arquivo fonte: *inter.c*, *inter.cpp* ou *inter.java*

O Intercursos da FIPP é uma competição de futebol de salão que reúne alunos de todos os cursos e termos da Faculdade de Informática. Durante os jogos podemos encontrar jogadores de todos os tipos, desde aqueles que dizem que jogam muita bola, mas na hora da partida são os maiores pipoqueiros, aqueles pernas-de-pau que só servem para jogar LoL, tem também aqueles que você não dá um centavo, mas na hora do jogo desequilibram, sem falar dos goleiros, para alguns chega a ser ofensa chamá-los assim. A classificação do Intercursos é baseada no número de pontos ganhos pelos times, e a distribuição de pontos é feita da forma usual. Ou seja, quando um time ganha um jogo, ele recebe 3 pontos; se o jogo termina empatado, ambos os times recebem 1 ponto; e o perdedor não recebe nenhum ponto.

Dada a classificação atual dos times e o número de times participantes no Intercursos, sua tarefa é de determinar quantos jogos terminaram empatados até o momento.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém dois inteiros T e N, indicando respectivamente o número de times participantes ($2 \leq T \leq 200$) e o número de partidas jogadas ($0 \leq N \leq 10000$). Cada uma das T linhas seguintes contém o nome de um time (uma cadeia de no máximo 10 letras e dígitos), seguido de um espaço em branco e do número de pontos que o time obteve até o momento. O final da entrada é indicado por $T = N = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada um dos casos de teste, seu programa deve imprimir uma única linha contendo um número inteiro, representando a quantidade de jogos que terminaram empatados até o momento.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**

`3 3`
`Zuadores 3`
`CSplayer 3`
`Mascarados 3`
`3 3`
`Fippanos 5`
`PernasDePau 1`
`LoLsoccer 1`
`0 0`
	`0`
	`2`

Vowels Frequencies (azul)

Problema C

Source file: *vowels.c*, *vowels.cpp* or *vowels.java*

The English alphabet consists of 26 letters. Five of these (a, e, i, o and u) are classified as vowels, the remaining 21 as consonants. Almost every English word contains at least one vowel (rhythm is one of the few exceptions).

In this problem you will be given a number of pieces of English text. Your task is to determine the frequency of each vowel that is found in the piece, and to display the answers sorted by frequency, highest frequency first. Where two vowels are equally frequent, they are to be displayed in alphabetical order.

As you can see from the examples below, upper case and lower case letters are considered to be the same letter in this problem. Use lower case in your output. As you can see from the second example, a frequency of zero must still be displayed.

Input

Each piece of text to be analysed is on a separate line of the input file. Each line has at most 200 characters. A single '#' on a line indicates the end of input.

The input must be read from standard input.

Output

Output for a problem must be on a single line. Each vowel must be output in lower case, followed by a colon, followed by the frequency of that vowel. There must be one space before the next letter, and a dot at the end.

The output must be written to standard output.

Sample Input

```
This piece of text was written in the city of Auckland.
ACM Programming Contest.
#
```

Output for the sample Input

```
e:5 i:5 a:3 o:2 u:1.
a:2 o:2 e:1 i:1 u:0.
```

Problema

D

Clay Bully (verde)

Source file: *clay.c*, *clay.cpp* or *clay.java*

Ms. Terry is a pre-school art teacher who likes to have her students work with clay. One of her assignments is to form a lump of clay into a block and then measure the dimensions of the block. However, in every class, there is always one child who insists on taking some clay from some other child. Since Ms. Terry always gives every child in a class the same amount of clay to begin with, you can write a program that helps Ms. Terry find the bully and victim after she measures each child's finished block.

Input

There are one or more classes of students, followed by a final line containing only the value *-1*. Each class starts with a line containing an integer, *n*, which is the number of students in the class, followed by *n* lines of student information. Each line of student information consists of three positive integers, representing the dimensions of the clay block, followed by the student's first name. There can never be more than 9 students nor less than 2 students in any class. Each student's name is at most 8 characters. Ms. Terry always gives each student at most 250 cubic units of clay. There is exactly one bully and one victim in each class.

The input must be read from standard input.

Output

For each class print a single line exactly as shown in the sample output.

The output must be written to standard output.

Sample input	**Output for the sample input**
3 10 10 2 Jill 5 3 10 Will 5 5 10 Bill 4 2 4 10 Cam 4 3 7 Sam 8 11 1 Graham 6 2 7 Pam -1	Bill took clay from Will. Graham took clay from Cam.

Moedas de ouro (rosa)

Problema E

Arquivo fonte: *moedas.c*, *moedas.cpp* ou *moedas.java*

O rei paga seu cavaleiro leal em moedas de ouro. No primeiro dia de seu serviço, o cavaleiro recebe uma moeda de ouro. Em cada um dos próximos dois dias (o segundo e terceiro dias de serviço), o cavaleiro recebe duas moedas de ouro. Em cada um dos próximos três dias (o quarto, quinto e sexto dias de serviço), o cavaleiro recebe três moedas de ouro. Em cada um dos quatro dias seguintes (sétimo, oitavo, nono e décimo dias de serviço), o cavaleiro recebe quatro moedas de ouro. Esse padrão de pagamentos continuará indefinidamente: depois de receber N moedas de ouro em cada um dos N dias consecutivos, o cavaleiro vai receber $N + 1$ moedas de ouro em cada um dos próximos $N + 1$ dias consecutivos, sendo N um inteiro positivo.

Seu programa irá determinar o número total de moedas de ouro pagas ao cavaleiro em um determinado número de dias (a partir do Dia 1).

Entrada

Cada linha da entrada (exceto a última) contém dados para um caso de teste do problema, que possui, exatamente, um número inteiro (na faixa 1..10000), representando o número de dias. O final da entrada é sinalizado por uma linha contendo o número 0.

A entrada deve ser lida da entrada padrão.

Saída

Há exatamente uma linha de saída para cada caso de teste. Essa linha contém o número de dias a partir da linha correspondente de entrada, seguido por um espaço em branco e pelo número total de moedas de ouro pagas ao cavaleiro no número determinado de dias, começando com Dia 1.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**
10 6 7 11 15 16 100 10000 1000 21 22 0	10 30 6 14 7 18 11 35 15 55 16 61 100 945 10000 942820 1000 29820 21 91 22 98

Problema
F

Rota Crítica (branca)

Arquivo fonte: *rota.c*, *rota.cpp* ou *rota.java*

Uma tragédia aconteceu recentemente em sua cidade. Um paciente em condição crítica, que necessitava de tratamento urgente, morreu enquanto era transportado para um grande hospital da capital do estado. O que ocorreu foi que a ambulância ficou presa no trânsito, devido a uma rocha que deslizou na estrada. A população reclamou com o governador, que agora deseja evitar acontecimentos similares no futuro. Infelizmente, deslizamentos de rochas são muito comuns nesse estado, pois nele há muitas montanhas e serras. Assim, para minimizar o número de tragédias ocasionadas pelos deslizamentos de rochas e outros imprevistos, o governador decidiu criar rotas alternativas entre cada cidade do estado e a capital. Para isso, é necessário inicialmente identificar quais são os segmentos de estradas atualmente críticos, isto é, se bloqueados fazem com que não haja caminho possível entre alguma cidade e a capital. Um segmento de estrada é um trecho (caminho) que liga duas cidades distintas.

Sua tarefa é escrever um programa para identificar esses segmentos críticos de estradas.

Entrada

A entrada é composta de vários casos de testes. A primeira linha de um caso de teste contém dois inteiros N e M que indicam, respectivamente, o número de cidades ($2 \leq N \leq 100$) e o número de segmentos de estrada ($1 \leq M \leq 10000$). Cada uma das N linhas seguintes contém o nome de uma cidade (apenas letras minúsculas e maiúsculas, comprimento máximo de 20 caracteres). A primeira dessas cidades é a capital do estado. Cada uma das M linhas seguintes descreve um segmento de estrada, contendo um par de nomes de cidades separados por um espaço em branco. Note que, como as montanhas causam dificuldade na construção de estradas, muitos segmentos de estrada são de mão única. Um segmento com duas mãos é representado por dois trechos de mão única. Você deve supor que existe ao menos um caminho de cada cidade para a capital. O final da entrada é indicado por $N = M = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste, seu programa deve listar os segmentos críticos, com um segmento crítico por linha. Cada segmento crítico deve ser representado por dois nomes de cidades separados por um espaço em branco. Os segmentos críticos de estrada devem ser listados na mesma ordem em que aparecem na entrada; para cada segmento, as cidades devem ser listadas na mesma ordem em que aparecem na entrada. Se não existir nenhum segmento crítico seu programa deve imprimir uma linha contendo apenas a palavra “Nenhuma”.

Imprima uma linha em branco após cada caso de teste.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	Saída para o exemplo de entrada
6 10 PortoAlegre Gramado Canela NovoHamburgo Pelotas RioGrande Canela Gramado Canela NovoHamburgo Gramado NovoHamburgo NovoHamburgo PortoAlegre PortoAlegre NovoHamburgo RioGrande Pelotas Pelotas PortoAlegre PortoAlegre Pelotas Pelotas RioGrande NovoHamburgo Canela 3 5 Sacramento SanFrancisco SantaClara SanFrancisco Sacramento Sacramento SantaClara SantaClara SanFrancisco SanFrancisco Sacramento Sacramento SanFrancisco 3 4 Recife Olinda Paulista Olinda Recife Paulista Recife Olinda Paulista Paulista Olinda 0 0	Gramado NovoHamburgo NovoHamburgo PortoAlegre RioGrande Pelotas Pelotas PortoAlegre SantaClara SanFrancisco Nenhuma