XII MARATONA DE PROGRAMAÇÃO InterFatecs 2023

Fase 2 - 30/09/2023

Caderno de Problemas

Organização e Patrocínio

OURO

alura

livelo

SOC

T2S

PRATA

iPORT Solutions

COLÉGIO UNIVERSO

BRONZE

FORTEC

MEDIA PARTNER

tv tribuna

APOIO INSTITUCIONAL

Fatec Praia Grande

CPS

SÃO PAULO GOVERNO DO ESTADO

interfatecs.com.br

1 Instruções

Este caderno contém 10 problemas – identificados por letras de A até K, com páginas numeradas de 3 até 29. Verifique se seu caderno está completo.

Informações gerais

1. Sobre a competição
   (a) A competição possui duração de 5 horas (início as 13:00h término as 18:00h);
   (b) NÃO é permitido acesso a conteúdo da Internet ou qualquer outro meio eletrônico digital;
   (c) NÃO é permitido o uso de ferramentas de auxílio à codificação, como GitHub Copilot, Tabnine, Amazon CodeWhisperer ou similar;
   (d) É permitido somente acesso a conteúdo impresso em papel (cadernos, apostilas, livros);
   (e) Não é permitida a comunicação com o técnico ou qualquer outra pessoa que não seja a equipe para tirar dúvidas sobre a maratona
   (f) Cada equipe terá acesso a 1 computador dotado do ambiente de submissão de programas (BOCA), dos compiladores, link-editores e IDEs requeridos pelas linguagens de programação permitidas;
   (g) NÃO é permitido o uso de notebooks, smartphones, ou outro tipo de computador ou assistente pessoal;
   (h) Todos os problemas têm o mesmo valor na correção.
2. Sobre o arquivo de solução e submissão:
   (a) O arquivo de solução (o programa fonte) DEVE TER o mesmo nome que o especificado no enunciado (logo após o título do problema);
   (b) confirme se você escolheu a linguagem correta e está com o nome de arquivo correto antes de submeter a sua solução;
   (c) NÃO insira acentos ou outros caracteres especiais no arquivo-fonte.
3. Sobre a entrada
   (a) A entrada de seu programa deve ser lida da entrada padrão (não use interface gráfica);
   (b) Seu programa será testado em vários casos de teste válidos além daqueles apresentados nos exemplos. Considere que seu programa será executado uma vez para cada caso de teste.
4. Sobre a saída
   (a) A saída do seu programa deve ser escrita na saída padrão;
   (b) Não exiba qualquer outra mensagem além do especificado no enunciado.
5. Versões das linguagens
   (a) gcc version 11.3.0 (lembre-se que não existem bibliotecas do Windows)
   (b) Python 3.10.6
   (c) Java 17.0.7+7-Ubuntu-0ubuntu122.04.2 (lembre-se que as classes devem estar fora de pacotes)

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problem A

Load balancer

Source file: loadbalancer.{ c | cpp | java | py }
Author: Prof. Dr. Leandro Luque (Fatec Mogi das Cruzes)

Load balancing is fundamental in distributed applications to promote system efficiency and availability. A load balancer can operate through various strategies. One of them involves sending a new request to the least busy server. In this scenario, when a balancer receives a request, it must decide which server to forward it to, considering the current load on each server. For instance, the load can be measured as the number of requests that the server is processing at the moment. Munarinho is developing a load balancer and needs to verify which server a request will be forwarded to at a given point in time, given the number of servers and a set of requests. To simplify the solution, assume that each request arrives at a specific time instant t (1, 2, 3, 4, ...) and that servers can process infinite requests in parallel. Also, when there is a tie in load between servers, the one with the lowest id will be selected.

Input

The first line of the input contains an integer s ($1 \leq s \leq 10^3$), representing the number of available servers. The second line contains an integer r ($1 \leq r \leq 10^4$), representing the number of requests. The third line contains an integer a ($1 \leq a \leq r$), indicating the order number of one of the informed requests (the first one starting at 1). The next r lines contain two integers t and d ($1 \leq t, d \leq 10^7$), representing, respectively, the arrival instant of the request and how long it will take for this request to be completed. Assume that if a request started at instant $t = 1$ and has a duration $d = 3$, it will finish right before $t = 4$. Therefore, at $t = 4$, the server that processed it will already be free.

Output

You must determine the id of the server to which the informed request will be destined. The servers have ids starting at 1.

Example of Input 1	Example of Output 1
3 7 2 1 10 30 1000 2 20 5 80 3 50 10 200 9 500	1

Example of Input 2	Example of Output 2
3 7 6 1 10 30 1000 2 20 5 80 3 50 10 200 9 500	3

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema B

Impressora 3D Unidimensional

Arquivo fonte: impressora3d.{ c | cpp | java | py }
Autor: Prof. Dr. Leandro Luque (Fatec Mogi das Cruzes)

Munarinho está desenvolvendo uma impressora 3D que opera de forma unidimensional, construindo camadas de material ao longo de um único eixo. Ela opera por meio de comandos de impressão, que adicionam uma quantidade de material no topo do intervalo [x, y]. A impressora possui uma altura máxima de construção, e Munarinho deve garantir que uma série de comandos de impressão não ultrapasse essa altura.

Como exemplo, assuma uma impressora com largura 20 e altura 10, além dos seguintes comandos de impressão:

```
início término (quantidade de material)
1 20 1
2 5 1
5 10 2
```

Após a impressão, teremos o seguinte resultado:

```
        X
        X X X X X X
  X X X X X X X X X
X X X X X X X X X X X X X X X X X X X X
```

A altura máxima atingida terá sido 4, no ponto 5. Portanto, os comandos são válidos para impressão.

Entrada

A primeira linha da entrada contém três inteiros separados por espaço n ($1 \leq n \leq 10^7$), m ($1 \leq m \leq 10^5$), e q ($1 \leq q \leq 10^5$), a largura da impressora, a altura da impressora e o número de comandos de impressão, respectivamente. Cada uma das próximas q linhas contém três inteiros separados por espaço, a, b ($1 \leq a \leq b \leq n$) e c ($1 \leq c \leq 10^9$), representando os intervalos de impressão [a, b] e a quantidade de material a ser depositada em cada ponto deste intervalo.

Saída

Caso a altura máxima da impressora não tenha sido atingida, a saída deve conter um inteiro i indicando a altura máxima atingida pelos comandos de impressão. Caso tenha sido atingida, a saída deve conter o texto `invalida`.

Exemplo de Entrada 1

```
12 19 8
4 6 2
12 12 1
2 9 1
5 11 2
3 11 2
3 9 1
2 8 2
11 11 2
```

Exemplo de Saída 1

```
10
```

Exemplo de Entrada 2

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

Exemplo de Saída 2

```
invalida
```

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema C

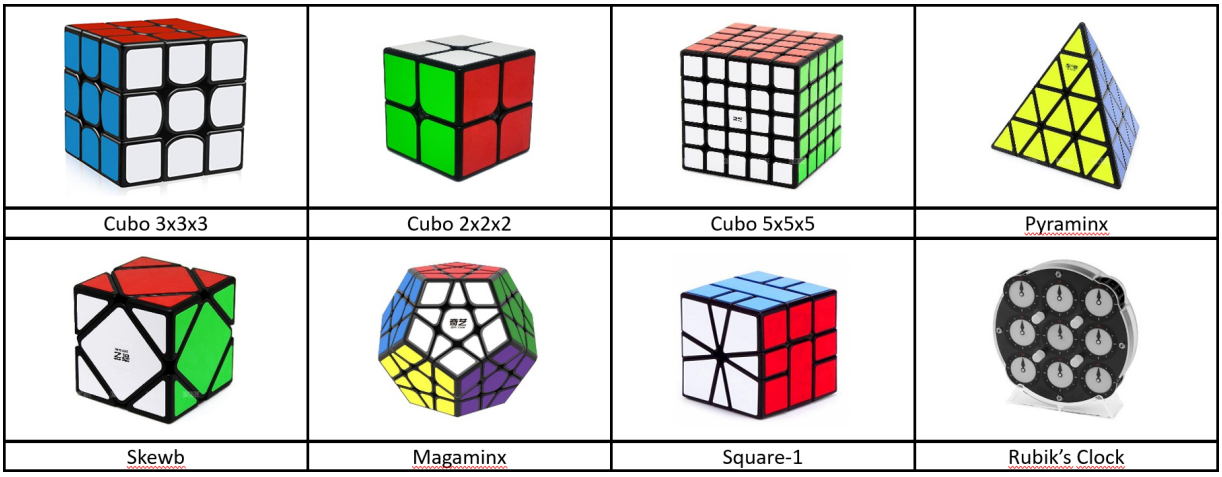
WCA

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

Autor: Prof. Me. Sérgio Luiz Banin (Fatec São Paulo e Fatec São Caetano do Sul)

Um dos brinquedos mais populares do mundo é conhecido no Brasil como "Cubo Mágico". Internacionalmente é conhecido como "Rubik's Cube", pois o seu formato mais clássico – o cubo 3x3x3 – foi inventado em 1974 pelo professor húngaro Ernö Rubik. Nos anos 1980, espalhou-se pelo mundo todo dando início à formação de uma grande comunidade de "cubistas", como são chamados no Brasil seus aficionados.

Considerando a popularidade do Cubo Mágico, ao longo do tempo foram surgindo outros brinquedos parecidos variando-se a quantidade de camadas – existem variantes com 2, 4, 5, 6 camadas e além, indo até o maior existente hoje com 21 camadas; a geometria – como o Pyraminx e o Megaminx; as faces de giro – como o Skewb (lê-se "isquiube") e o Square-1, etc.



Dada a complexidade do brinquedo e a existência de algoritmos que possibilitam sua solução em curto período de tempo, dependendo apenas da habilidade do cubista, não demorou muito para que surgissem competições nas quais o objetivo é justamente esse: resolver o Cubo no menor tempo.

A World Cube Association – WCA – é uma organização internacional voltada para a organização de competições de cubistas. A WCA instituiu regras padronizadas para as competições e assumiu a missão de nomear delegados regionais que ficam encarregados de organizar as competições e garantir que as regras da Associação sejam observadas e respeitadas por todos os competidores. Você encontrará todos os detalhes a respeito em seu site oficial: https://www.worldcubeassociation.org/

Você foi convidado por um amigo cubista e delegado da WCA para desenvolver um software que apoie a apuração dos tempos em uma competição. Fique aqui registrado que a WCA tem um sistema pronto, baseado em nuvem, e que pode ser usado pelos delegados em qualquer competição. Porém, seu amigo enfrentou um problema em uma das competições que gerenciou no passado: a internet caiu e não voltou. Então ele deseja contar com um software offline que apure os resultados a partir dos tempos registrados para os competidores e convidou você para escrevê-lo.

Uma competição tem várias provas com cubos de diferentes tipos. Os competidores fazem a inscrição e recebem um número de identificação (Id) atribuído por ordem de inscrição e cada um tem a liberdade de definir em quais provas irá participar.

No dia da competição, cada participante, em cada prova, resolve o cubo cinco vezes. Tais tempos são anotados na forma minutos:segundos:milissegundos. A partir daí resultam duas informações importantes: o Menor Tempo e a Média do participante. Para o cálculo da média, são descartados dois valores: o menor e o maior tempos, e a média é calculada com os três tempos restantes. O menor tempo sempre será o menor dos cinco tempos, não havendo descarte.

Considere o exemplo de registro de tempos para o competidor com Id = 3:

```
3 0:11:867 0:11:740 0:12:173 0:12:121 0:11:876
```

Nesse exemplo o Menor Tempo é `0:11:740` e a Média será calculada com `0:11:867, 0:11:876 e 0:12:121`, resultando no valor final `0:11:9546666666...` uma dízima periódica. Pelas regras da Associação o campo de milissegundos deve ser limitado a 3 dígitos sem arredondamentos, ou seja, qualquer valor maior que zero da 4ª casa em diante deve ser descartado. Com isso:

```
Média (0:11:867, 0:11:876, 0:12:121) = 0:11:954
```

Outro fato que pode acontecer na competição é um participante não conseguir resolver o cubo em alguma das rodadas. Neste caso será anotado o tempo `0:00:000` e isso é conhecido como DNF ou "Did Not Finish". A partir daí existem três casos possíveis:

1. Se o competidor tiver apenas um DNF entre as 5 tentativas, então o DNF será descartado como o pior tempo para cálculo da Média e o Menor Tempo será considerado normalmente;

2. Se o competidor tiver de 2 a 4 DNFs entre as 5 tentativas, então a Média será DNF e o Menor Tempo será considerado normalmente;

3. Se o competidor tiver 5 DNFs, então tanto o Menor Tempo quanto a Média serão DNF.

Após apurados os tempos de cada competidor, a classificação da prova será feita pela Média de tempo dos competidores. Havendo empate, o desempate será pelo Menor Tempo. É muito raro haver empate pelos dois critérios, uma vez que há milissegundos envolvidos. Quando houver DNF, ele deve ser considerado um tempo muito grande, ou seja, jogadores com Média e Menor Tempo DNF (caso 3 acima) ficarão no final da classificação e serão listados em ordem alfabética. Imediatamente acima deles, estarão os competidores com Média DNF e Menor Tempo apurado (caso 2 acima).

Entrada

Cada entrada contém um caso de teste. A primeira linha contém o número de competidores $NComp$ ($4 \leq NComp \leq 100$). Cada uma das próximas $NComp$ linhas contém a relação de competidores com duas informações por linha: Id e $Nome$ do Competidor, cujo tamanho máximo é de 20 caracteres. O Id e o $Nome$ são separados por um espaço em branco e o $Nome$ pode ser composto havendo, eventualmente, o caractere espaço em branco, p.ex. Ana Clara.

Terminada a relação de competidores começam os dados das provas. Existem várias provas na competição de modo que haverá vários blocos conforme descrito a seguir.

Para cada prova haverá uma linha de cabeçalho com duas informações: o número de participantes na prova $NPart$ ($3 \leq NPart \leq 100$) e o nome do cubo que será usado na prova $NomeCubo$, cujo tamanho

máximo é de 30 caracteres. Entre $NPart$ e $NomeCubo$ haverá um espaço em branco e $NomeCubo$ pode ser composto havendo, eventualmente, o caractere espaço em branco, p. ex: Cubo 3x3x3

Em seguida, haverá $NPart$ linhas contendo o Id do competidor e os 5 tempos registrados no seguinte formato: `3 0:11:867 0:11:740 0:12:173 0:12:121 0:11:876`

A última linha da entrada conterá a palavra "FIM", indicando o término do bloco de provas e também da entrada.

Saída

A saída contém uma primeira linha que serve como cabeçalho fixo.

```
.Id.Nome.....................Media......Melhor
```

Após isso, para cada prova da competição, deve ser impresso o nome do cubo usado $NomeCubo$ e, a seguir, devem ser impressos o Id, o $Nome$, a $Media$ e o $Melhor\ Tempo$ dos $NPart$ participantes, ordenados por menor $Media$ e $Melhor\ Tempo$, conforme descrito anteriormente, incluindo os casos com DNF.

Quanto ao posicionamento na linha, sigam as diretrizes:

- Para o campo Id considere 3 posições com os números alinhados à direita
- Um espaço em branco separando o campo Id do campo $Nome$
- Para o campo $Nome$ considere 20 posições com os nomes alinhados à esquerda
- Para o campo $Media$ considere 12 posições com alinhamento à direita
- Para o campo $Melhor\ Tempo$ considere 12 posições com alinhamento à direita

Quanto aos tempos ($Media$ e $Melhor\ Tempo$) considere as seguintes diretrizes:

- O formato geral desses campos é `0:00:000 (minuto:segundo:milissegundo)`
- Se o minuto for maior que zero ele deve ser exibido sem qualquer zero à esquerda
- Se minuto for igual a zero ele não deve ser exibido, nem o caractere “:” que o sucede
- Os segundos sempre devem ser exibidos com dois dígitos, ou seja, para valores menores que 10 deve ser exibido um zero à esquerda, p.ex: `09:312` (nove segundos, trezentos e doze milésimos)
- Quando o tempo for DNF ele deve estar alinhado à direita

Em todas as linhas deve haver o caractere de fim de linha.

Exemplo de Entrada 1

```
8
1 Theo Dubois
2 Lola Mallet
3 Lorium Nbani
4 Keiko Sato
5 Emily Valdorf
6 Yohanes Tetrakis
7 Yousef Rosemberg
8 Liam Chub
8 Cubo 3x3x3
1 0:31:000 0:26:000 0:00:000 0:30:000 0:00:000
2 0:18:000 0:18:000 0:19:000 0:20:000 0:22:000
3 0:09:000 0:10:000 0:11:000 0:12:000 0:13:000
4 0:17:000 0:18:000 0:19:000 0:20:000 0:21:000
5 0:10:000 0:11:000 0:12:000 0:13:000 0:14:000
6 0:00:000 0:00:000 0:00:000 0:00:000 0:00:000
7 0:16:000 0:18:000 0:20:000 0:22:000 0:24:000
8 0:00:000 0:00:000 0:00:000 0:00:000 0:00:000
FIM
```

Exemplo de Saída 1

```
.Id.Nome.....................Media......Melhor
Cubo 3x3x3
  3 Lorium Nbani               11:000      09:000
  5 Emily Valdorf              12:000      10:000
  4 Keiko Sato                 19:000      17:000
  2 Lola Mallet                19:000      18:000
  7 Yousef Rosemberg           20:000      16:000
  1 Theo Dubois                   DNF      26:000
  8 Liam Chub                     DNF         DNF
  6 Yohanes Tetrakis              DNF         DNF
```

Exemplo de Entrada 2

```
10
1 Pato Donald
2 Margarida
3 Tio Patinhas
4 Vovo Donalda
5 Zezinho
6 Huguinho
7 Luizinho
8 Peninha
9 Gansolino
10 Irmaos Metralha
10 Cubo 3x3x3
1 0:31:876 0:26:441 0:00:000 0:30:746 0:28:317
2 0:20:252 0:20:128 0:19:878 0:20:298 0:19:979
3 0:11:867 0:11:740 0:12:173 0:12:121 0:11:876
4 0:20:791 0:20:739 0:20:955 0:21:148 0:21:063
5 0:09:115 0:09:072 0:09:227 0:09:252 0:09:094
6 0:08:979 0:08:751 0:08:705 0:08:946 0:09:038
7 0:10:702 0:09:934 0:10:168 0:13:996 0:10:054
8 0:15:141 0:14:991 0:15:062 0:14:903 0:15:106
9 0:17:050 0:17:071 0:17:001 0:17:094 0:16:978
10 0:40:907 0:00:000 0:42:741 0:00:000 0:41:109
6 Megaminx 3x3
3 2:44:126 2:44:026 2:43:812 2:43:778 2:44:125
5 0:58:257 0:57:909 0:57:241 1:03:998 0:57:289
6 1:03:758 1:01:941 1:02:362 1:06:149 0:59:884
7 0:58:256 0:59:082 0:57:833 0:58:093 0:58:183
8 12:39:757 11:39:965 12:03:043 11:51:240 12:25:121
10 0:00:000 0:00:000 0:00:000 0:00:000 0:00:000
7 Skewb
2 0:08:728 0:09:201 0:08:733 0:08:905 0:08:808
3 0:04:246 0:03:973 0:04:186 0:03:813 0:03:863
5 0:02:977 0:03:100 0:02:753 0:03:236 0:02:911
4 0:10:132 0:10:229 0:09:790 0:10:120 0:09:961
6 0:03:182 0:03:269 0:03:136 0:02:918 0:02:703
7 0:02:912 0:02:939 0:03:137 0:03:261 0:02:100
8 0:07:001 0:06:858 0:07:004 0:07:123 0:06:767
FIM
```

Exemplo de Saída 2

```
.Id.Nome.....................Media......Melhor
Cubo 3x3x3
  6 Huguinho                  08:892      08:705
  5 Zezinho                   09:145      09:072
  7 Luizinho                  10:308      09:934
  3 Tio Patinhas              11:954      11:740
  8 Peninha                   15:053      14:903
  9 Gansolino                 17:040      16:978
  2 Margarida                 20:119      19:878
  4 Vovo Donalda              20:936      20:739
  1 Pato Donald               30:313      26:441
 10 Irmaos Metralha              DNF      40:907
Megaminx 3x3
  5 Zezinho                   57:818      57:241
  7 Luizinho                  58:177      57:833
  6 Huguinho                1:02:687      59:884
  3 Tio Patinhas            2:43:987    2:43:778
  8 Peninha                12:06:468   11:39:965
 10 Irmaos Metralha              DNF         DNF
Skewb
  7 Luizinho                  02:996      02:100
  5 Zezinho                   02:996      02:753
  6 Huguinho                  03:078      02:703
  3 Tio Patinhas              04:007      03:813
  8 Peninha                   06:954      06:767
  2 Margarida                 08:815      08:728
  4 Vovo Donalda              10:071      09:790
```

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema D

Injeção de dependências

Arquivo fonte: injecaodependencia.{ c | cpp | java | py }
Autor: Prof. Dr. Leandro Luque (Fatec Mogi das Cruzes)

A gestão eficiente de dependências é crucial para reduzir o acoplamento em sistemas de software, promovendo a modularidade e a reusabilidade do código. Dentre as técnicas para gerenciar dependências, a Injeção de Dependência e o padrão *Service Locator* são amplamente reconhecidos. Na Injeção de Dependência, uma classe não instancia diretamente os objetos das classes das quais depende; ao invés disso, ela os recebe através do construtor, de atributos ou de métodos de acesso.

Desafios surgem quando existem dependências cíclicas, como no caso de A -> B, B -> C e C -> A. Para instanciar A, é necessário instanciar e injetar B, que por sua vez depende de C, que depende de A, formando um ciclo. Uma solução é empregar a "injeção tardia", onde A pode ser criado sem a imediata injeção de B, que ocorrerá após a criação de todas as dependências.

Munarinho está elaborando um *framework* de injeção de dependências para um novo projeto de software e precisa que este identifique automaticamente dependências cíclicas entre os módulos do projeto.

Entrada

A primeira linha da entrada contém um inteiro n ($1 \leq n \leq 500$), representando o número de dependências no projeto. As próximas n linhas contêm dois caracteres maiúsculos a e b indicando que o módulo a depende do módulo b.

Saída

A saída deve apresentar a string "usar injecao tardia"se for identificada uma dependência cíclica no projeto; caso contrário, deve apresentar "ok".

Exemplo de Entrada 1	Exemplo de Saída 1
3 A B B C C A	usar injecao tardia

Exemplo de Entrada 2	Exemplo de Saída 2
2 A B C D	ok

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema E

Torres de Hanoi. Again.

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

Autor: Prof. Antonio Cesar de Barros Munari (Fatec Sorocaba)

O jogo Torres de Hanoi é bastante conhecido do pessoal que estuda computação. Em geral, é um dos exemplos clássicos utilizados nas aulas sobre recursividade em programação, já que um elegante algoritmo recursivo é capaz de resolvê-lo sem grande dificuldade. Para quem não está associando o nome à pessoa, Torres de Hanoi possui 3 postes fixos, geralmente chamados de A, B e C, e uma certa quantidade finita de discos. Cada disco possui um diâmetro diferente dos demais, e apresenta um furo em seu centro, de maneira que é possível encaixá-lo em qualquer um dos postes que se desejar. Na configuração inicial todos os discos encontram-se encaixados em um dos postes, em ordem de tamanho, com o disco menor em cima e o disco maior embaixo, conforme ilustra a figura 1, onde a configuração dispõe de 5 discos.

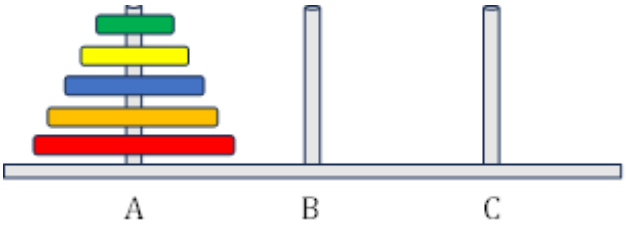


Figura E.1: Estado inicial de uma Torre de Hanoi com 5 discos.

O objetivo do jogo é transferir todos os discos para um poste indicado como destino, sendo permitido mover apenas um disco por vez, utilizando o poste restante como auxiliar para armazenamento e não podendo colocar, em qualquer momento, um disco maior sobre um disco menor no mesmo poste. Existe sempre uma sequência ótima de movimentos para atingir esse objetivo, como mostra a figura 2 onde, para um arranjo de três discos, são necessários sete movimentos. Essa quantidade ótima de movimentos é, como você já deve desconfiar, proporcional à quantidade de discos disponíveis no jogo: para um disco, um movimento; para dois discos, três movimentos; para quatro discos, quinze movimentos e assim por diante. Após cada movimento válido, o jogo assume um novo estado válido, e no processo transitamos de um estado inicial válido para um estado final também válido. É possível descrever cada estado válido do jogo por meio de uma sequência única e específica de zeros e uns, onde o comprimento dessa sequência é igual à quantidade de discos. O primeiro dígito (aquele mais à esquerda) corresponde ao maior disco, o último dígito (aquele mais à direita) representa o menor disco, e os outros dígitos seguem a mesma lógica. Por exemplo, o estado inicial válido para o jogo com 3 discos será 000; se forem 4 discos, será 0000, etc. Considerando o jogo expresso na figura 2, teremos por definição o estado inicial 000; depois do primeiro movimento, teremos 001, que significa que os dois discos maiores estão no mesmo poste e o disco menor está em um outro poste. O segundo movimento levará o jogo para uma configuração 010 (maior disco em um poste, disco intermediário em um poste diferente daquele onde está o maior disco e o disco menor estando em um poste diferente daquele onde está o disco intermediário). O terceiro movimento produzirá o estado 011 (disco maior em um poste, os outros dois discos empilhados em um mesmo outro poste, distinto daquele onde está o maior). O quarto movimento levará a 100 (e aqui perceba um padrão: o disco recém movimentado terá sempre o dígito ‘1’); após o quinto movimento teremos 101; o sexto movimento nos deixará no estado 110 e, finalmente, com o sétimo movimento, encontraremos o estado 111 (todos os discos em um mesmo poste).

Sua tarefa neste problema é bem simples: dada uma sequência de zeros e uns que expressa um estado válido

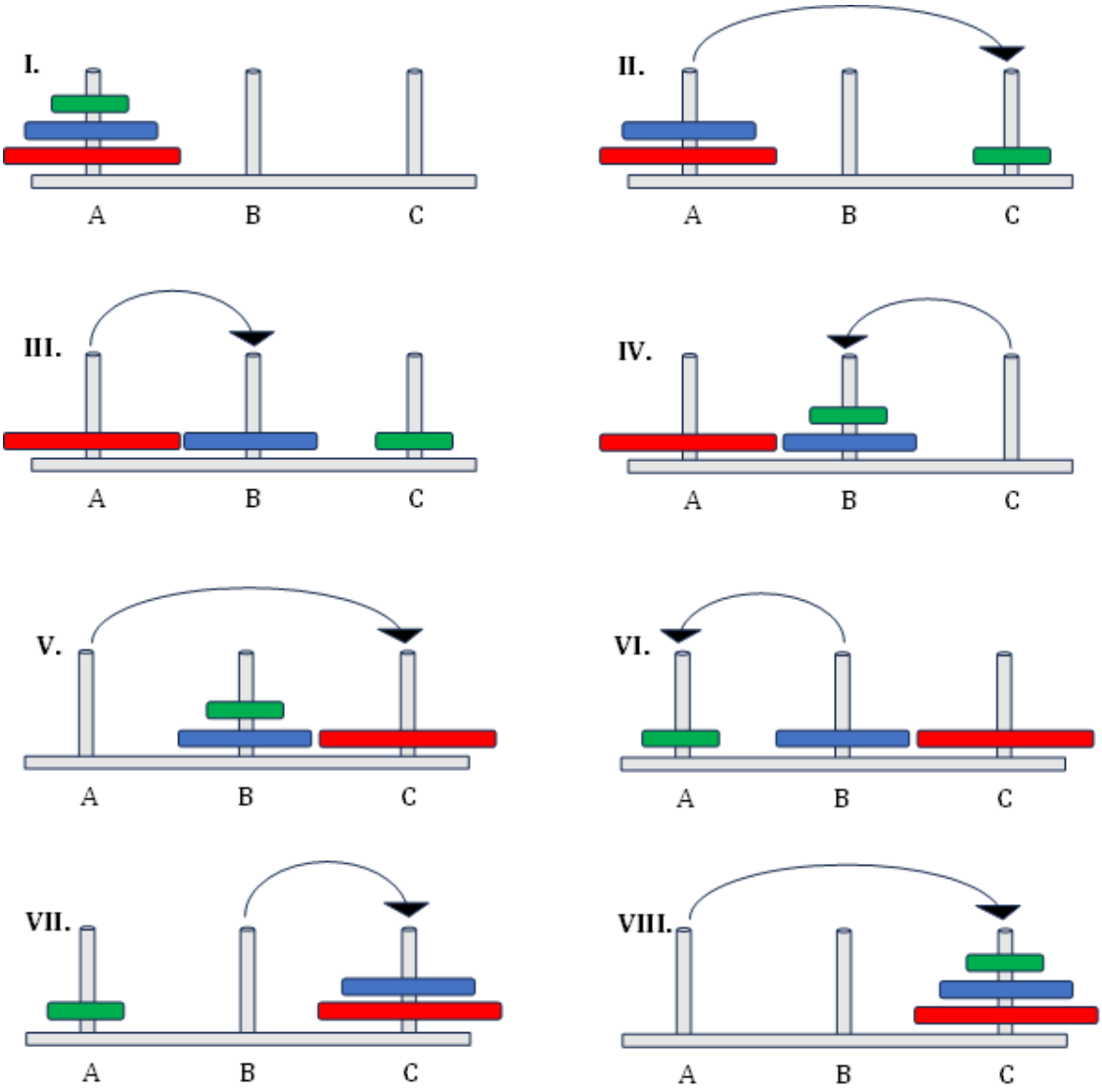


Figura E.2: Solução de uma Torre de Hanoi com 3 discos.

do jogo, indicar quantos movimentos válidos ainda faltam para finalizá-lo, de maneira ótima.

Entrada

Cada arquivo de entrada representa um único caso de teste. Na primeira linha teremos um inteiro N ($0 \leq N \leq 30$) indicando a quantidade de discos. Na segunda linha teremos uma sequência de N dígitos d ($d \in \{0, 1\}$) separados entre si por espaços em branco.

Saída

Imprimir um inteiro X que indica a menor quantidade de movimentos válidos que faltam para o jogo ser finalizado. Não esqueça de deixar uma quebra de linha após o valor impresso.

Exemplo de Entrada 1	Exemplo de Saída 1
3 0 1 0	5

Exemplo de Entrada 2	Exemplo de Saída 2
5 1 0 0 0 1	14

Exemplo de Entrada 3	Exemplo de Saída 3
6 1 0 0 1 1 1	24

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema F

Fácil demais!

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

Autor: Prof. Me. Lucio Nunes de Lira (Fatec Diadema e Fatec Ferraz de Vasconcelos)

Este problema é fácil demais! Dados números naturais positivos, que representam as posições de números primos na sequência infinita de primos, exiba uma mensagem indicando corretamente se a soma desses primos é par ou ímpar. Não entendeu? É tão simples! Por exemplo, se as posições dadas forem 1, 2, 3 e 5, simbolizando o 1º, 2º, 3º e 5º números naturais primos, respectivamente 2, 3, 5 e 11, a soma é 21, ímpar!

Entrada

Um número natural N $(1 \leqslant N \leqslant 10000)$ que representa a quantidade de posições de primos que serão dadas. Em cada uma das próximas N linhas, haverá um número natural P $(0 < P < 2^{64})$ que será a posição do número primo na sequência infinita de primos, note que há possibilidade de posições repetidas.

Saída

Apenas a palavra 'par' (em minúsculo e sem apóstrofos), indicando que a soma dos N números primos é par, ou 'impar' (em minúsculo, sem acentuação e sem apóstrofos), caso contrário.

Exemplo de Entrada 1	Exemplo de Saída 1
4 1 2 3 5	impar

Exemplo de Entrada 2	Exemplo de Saída 2
1 1	par

Exemplo de Entrada 3	Exemplo de Saída 3
3 1 4 2	par

Exemplo de Entrada 4	Exemplo de Saída 4
3 18446744073709551615 5684 245672783187735	impar

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema G

Processo Seletivo Simplificado

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

Autor: Prof. Antonio Cesar de Barros Munari (Fatec Sorocaba)

Uma das formas de contratação de professores para as unidades do Centro Paula Souza é por meio dos chamados PSSs (Processos Seletivos Simplificados). Nessa modalidade um professor é contratado por tempo determinado e poderá ministrar aulas diversas dentro de sua área de formação nas diversas unidades da nossa autarquia. Quando um PSS é lançado para uma disciplina, os interessados se inscrevem e apresentam a documentação exigida para comprovar sua qualificação para o posto e uma comissão analisa todas as inscrições e emite o resultado final daquele processo. Um dos itens que precisa ser verificado trata da titulação e tempo de experiência profissional do candidato. Esses requisitos são expostos no edital do PSS e estão descritos a seguir, no caso para disciplinas do tipo "profissionalizante".

"ANEXO II – REQUISITOS DA FUNÇÃO E DE TITULAÇÃO

Possuir, na data da inscrição: PARA DISCIPLINAS PROFISSIONALIZANTES

1. Ser graduado e titulado em programa de mestrado ou doutorado reconhecido ou recomendado na forma da lei, sendo a graduação ou a titulação em uma das áreas da disciplina, conforme edital de abertura do certame, bem como possuir experiência profissional relevante de pelo menos 03 (três) anos na área da disciplina, após a obtenção de grau acadêmico (graduação) ou da titulação (mestrado ou doutorado) na área objeto do certame; ou

2. Ser graduado em uma das áreas da disciplina, conforme edital de abertura do certame, e possuir especialização em nível de pós-graduação na mesma área da graduação, bem como experiência profissional relevante de pelo menos 05 (cinco) anos na área da disciplina, após a obtenção de grau acadêmico na área objeto do certame."

O coordenador do seu curso, que anda às voltas com alguns PSSs de disciplinas profissionalizantes, está precisando de um programa que, para um candidato, indique qual o tempo de experiência mínima será preciso comprovar caso este seja considerado apto a concorrer àquela vaga objeto do processo. A ideia é que esse programa ajude a evitar erros na interpretação desse critério de avaliação dos candidatos. O programa receberá um resumo dos dados básicos dos candidatos, elaborado pela comissão avaliadora e informará, para cada candidato, se ele foi desclassificado ou qual o tempo mínimo de atuação profissional que precisa ser comprovado.

Entrada

O arquivo de entradas se inicia com um inteiro N ($1 \leq N \leq 100$) indicando a quantidade de candidatos a serem analisados pelo programa. Seguem-se então N linhas, cada uma consistindo dos inteiros binários G, E, A e D, que indicam, respectivamente, se o candidato informa ter graduação na área, especialização na área, mestrado/doutorado na área e, finalmente, mestrado ou doutorado fora da área da disciplina objeto do oferecimento. Um valor 0 para um desses indicadores significa que o candidato não atende àquele requisito, enquanto que um valor 1 significa que o requisito é atendido. Cada linha termina com um inteiro T ($T \in \{2, 3, 5\}$), que indica o tempo mínimo de experiência profissional informado pelo candidato na sua ficha de inscrição: 2 significa "menos de 3 anos de experiência"; 3 significa "pelo menos 3 anos de experiência" e 5 significa "pelo menos 5 anos de experiência".

Saída

Para cada caso de teste o programa deverá imprimir uma mensagem informando o resultado preliminar da avaliação. Assim, caso o candidato atenda os requisitos, o texto deverá ser "Cand. 999: deferido (comprovar 99 anos)", onde 999 indica o número do candidato (primeiro candidato é 1, segundo candidato é 2, etc) e 99 indica se precisará comprovar 3 ou 5 anos de experiência, conforme os termos apresentados anteriormente. Se o candidato não comprovar os requisitos de titulação mínima, imprimir a mensagem "Cand. 999: INDEFERIDO (acad)"; se a titulação mínima for atendida mas o problema for tempo de experiência profissional insuficiente para a formação acadêmica informada, a mensagem deverá ser, então, "Cand. 999: INDEFERIDO (exp)". Observe o uso de maiúsculas e minúsculas nas mensagens, conforme os exemplos e não se esqueça de imprimir a resposta com uma quebra de linha no final.

Exemplo de Entrada 1	**Exemplo de Saída 1**
4 1 0 1 1 5 0 0 0 0 5 1 1 0 0 5 1 1 1 1 2	Cand. 1: deferido (comprovar 3 anos) Cand. 2: INDEFERIDO (acad) Cand. 3: deferido (comprovar 5 anos) Cand. 4: INDEFERIDO (exp)

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema H

Terminais de Terminus

Arquivo fonte: terminus.{ c | cpp | java | py }
Autor: Prof. Dr. Lucas Baggio Figueira (Fatec Ribeirão Preto)

Terminus se tornou a principal cidade da galáxia após a queda de Trantor, planeta central do Império Galático, com a consequente derrocada da dinastia Cleon. Terminus foi criada para ser a Fundação de um Segundo Império Galático, sendo que seu desenvolvimento tecnológico foi significativo nos últimos 500 anos. Entretanto, existem problemas práticos para serem resolvidos em Terminus e um deles está relacionado ao terminais do espaço-porto, onde diferentes espaçonaves vindas do hiperespaço atracam trazendo mercadorias e partem levando suprimentos e tecnologia para os lugares mais longínquos da galáxia.

O espaço-porto de Terminus possui 7 terminais, numerados de 1 a 7. Ao chegar, uma espaçonave é direcionada para algum desses terminais, onde a identificação da nave e o tempo-estelar é registrado. A partir daí, a nave poderá se manter no terminal que atracou ou, por motivos diversos, ser manobrada para outro terminal. De qualquer maneira, quando a nave deixar definitivamente o espaço-porto, sua identificação e tempo estelar devem ser registrados a fim de que sua estadia possa ser tarifada de acordo com o tempo-galáxia, uma unidade de tempo usada amplamente pelos sistemas habitados da Via Láctea.

Atualmente o valor do tempo-galaxia é de CT$ $4,59$ e um tempo-galáxia é igual à razão entre o tempo-estelar e $123,457$. Sendo assim, você deve, a partir do extrato de movimento de naves, determinar o valor de faturamento de espaço-porto.

Como o espaço-porto é muito movimentado, às vezes, uma espaçonave deverá aguardar na fila, ou seja, se uma espaçonave solicita um dado terminal e ele está ocupado, ela fica na fila aguardando a sua vez. Caso isso aconteça, a espaçonave pagará um adicional de 5% do valor total devido, Não obstante, se algum terminal estiver livre (sem fila), a espaçonave que está a mais tempo esperando em qualquer fila é deslocada para aquele terminal. Se uma espaçonave, a qualquer momento, trocar de terminal, deverá pagar um adicional de 7%. É importante destacar que estes valores são cumulativos entre si, mas não existe sobretaxa do adicional de fila e de deslocamento. Todas as manobras são realizadas pelos espaço-práticos, que sempre ficam atentos e, quando um terminal é detectado vago, ele é imediatamente preenchido com a espaçonave há mais tempo aguardando no espaço-porto. Os espaço-práticos, além de muito competentes, também são bem levados e manobram algumas espaçonaves à esmo, alterando sua posição na fila ou então no atracadouro. O capitão indicou que uma espaçonave só pode ser cobrada no momento de sua saída e, assim, deve necessariamente pagar o que está devendo. Portanto, deve-se estabelecer um movimento de caixa para o espaço-porto.

Entrada

As linhas contêm o extrato de movimento do espaço-porto. Cada linha possui a identificação da espaçonave I (no formato $AAA-9999$, sem espaços), o terminal T ($0 \leq T \leq 7$) e o tempo-estelar TE ($1 \leq TE \leq 2^{31}-1$) do evento, separados por um espaço em branco. Caso seja um movimento de saída, o terminal é especificado como 0. Neste caso, o valor de TE determina o tempo-estelar de saída.

Saída

A saída possui várias linhas, sendo que cada uma possui a identificação I da espaçonave e o valor total devido P (com duas casas decimais) seguido das letras: E, indicando tarifa extra por esperar na fila, e/ou X,

mudança de terminal. Quando as duas letras forem necessárias, devem ser exibidas nesta ordem, com um espaço em branco entre elas. Caso a nave não tenha ficado na fila e nem tenha trocado de terminal, não há necessidade de exibir nenhuma letra. As naves devem ser exibidas em ordem crescente de saída - as que saíram antes aparecem primeiro.

Exemplo de Entrada 1	**Exemplo de Saída 1**
XOK-1602 1 7538919 FGM-7741 2 5346870 BMA-7032 4 10289876 XKI-7025 5 8380184 FGM-7741 0 5877573 XOK-1602 0 8465897 XKI-7025 0 9151738 RQX-7568 5 7555075 BMA-7032 0 11034976 RQX-7568 0 7608977	FGM-7741 19730.97 RQX-7568 2004.02 XOK-1602 34464.06 XKI-7025 28685.56 BMA-7032 27702.03

Exemplo de Entrada 2	**Exemplo de Saída 2**
XOK-1602 1 919 XOK-1602 5 1370 XOK-1602 0 1557	XOK-1602 25.38 X

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema I

Criptografia hash

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

Autor: Prof. Me. Lucio Nunes de Lira (Fatec Diadema e Fatec Ferraz de Vasconcelos)

Daiane, profissional formada em arquitetura, gosta muito de matemática, tecnologia e segurança da informação, tanto que está cogitando seriamente ingressar em um curso superior na área de ciências da computação. Nesta área, o que mais chamou sua atenção foi a possibilidade em trabalhar com programação de computadores.

Ao conversar com alguns amigos programadores, percebeu que a habilidade de programar computadores está cada vez mais requisitada e que quanto maior a familiaridade com matemática, maior a facilidade para programar e vice-versa, inclusive escutou diversas frases do tipo "[...] eu nem gostava muito de matemática, mas quando comecei a programar entendi a aplicação daquilo que era dado na escola, como a 'Fórmula de Bhaskara' para encontrar as raízes de uma equação de segundo grau, o 'Teorema de Pitágoras' sobre os triângulos retângulos e a importância dos números primos".

A respeito dos números naturais primos, é fácil lembrar do que se tratam: são números naturais maiores do que 1 e que possuem apenas dois divisores naturais, o 1 e ele próprio. Daiane também sabe que os números primos são usados em algoritmos para criptografar dados, isto é, embaralhar uma mensagem e, consequentemente, reduzir a probabilidade de alguém não autorizado ter acesso ao seu conteúdo original, o que é essencial para manter a privacidade e a segurança desses dados.

Existem diversos algoritmos para criptografia, alguns permitem que os dados sejam embaralhados (cifragem) e, posteriormente desembaralhados (decifragem), algo útil na transmissão de mensagens entre pessoas. Outros algoritmos permitem apenas a cifragem, tornando quase impossível que o valor gerado a partir desse embaralhamento seja usado para retornar ao conteúdo original. Esses algoritmos são bastante usados para guardar senhas em bancos de dados. Como exemplo de aplicação dos algoritmos que permitem apenas cifragem, vamos supor que um cliente quer criar um usuário com senha para ter acesso a um sistema. Podemos resumir o procedimento assim:

- O cliente insere em uma tela de cadastro um nome de usuário e senha;
- O programa gera um valor com base em um embaralhamento desta senha, geralmente chamado de "valor hash", e guarda no banco de dados apenas o nome de usuário e esse valor hash;
- Para acessar o sistema, o cliente irá inserir seu nome de usuário e a senha;
- O sistema embaralhará a senha inserida e irá comparar esse valor de hash com aquele guardado no banco de dados e que está associado ao mesmo nome de usuário;
- Se os valores forem iguais, o acesso será concedido, caso contrário, o acesso será negado. Afinal, o algoritmo garante que o valor de hash será sempre o mesmo para os mesmos dados.

Essa estratégia é segura, pois caso o banco de dados seja acessado indevidamente, as senhas originais dos usuários não estarão expostas, apenas o valor hash. Que bom seria se todas as empresas tivessem esse cuidado com os dados de seus clientes...

Mal Daiane anunciou que pretende mudar de área e já recebeu um convite de sua amiga Michelly, dona de uma pequena empresa. A empresária disse que precisa melhorar a segurança de seu sistema e que pretende

coletar todas as senhas de seus clientes e guardá-las usando a estratégia que vimos anteriormente. Daiane então elaborou um algoritmo para criptografar as senhas:

1. Coleta-se o nome de usuário e a senha de um cliente;
2. Converte-se cada caractere da senha para o seu respectivo código ASCII (vide Figura I.1);
3. Multiplica-se cada código pelo número de sua posição, sendo que o primeiro está na posição 1, o segundo está na posição 2 e assim por diante;
4. Somam-se todos os códigos já multiplicados por suas posições, gerando um número natural S;
5. Calcula-se a decomposição de S em fatores primos, isto é: (a) cria-se uma lista vazia L; (b) encontra-se o menor número primo P que é divisor de S; (c) guarda-se P no final de L; (d) encontra-se o quociente Q da divisão de S por P; (e) atribui-se a S o valor de Q; (e) repete-se os passos (b), (c), (d) e (e) até que S seja 1;
6. Em L estão todos os fatores primos que, se multiplicados, resultam no S original, ou seja, o valor de S antes de ser alterado por consequência do passo 5. Note que, por razão do procedimento, podem existir fatores repetidos em L, o que aconteceria, por exemplo, se S fosse 100 (2 * 2 * 5 * 5);
7. Transforma-se todo fator primo de L em um texto, concatenando o fator que estiver na posição i+1 à direita do fator que estiver na posição i, gerando um único texto F;
8. Concatena-se o texto correspondente ao conteúdo do S original à esquerda de F, gerando um texto H, esse é o valor de hash do algoritmo de Daiane. Veja a ilustração de exemplo na Figura I.2.

Caracteres e respectivos códigos ASCII (tabela parcial)

Caractere	Código	Caractere	Código	Caractere	Código	Caractere	Código	Caractere	Código
A	65	N	78	a	97	n	110	0	48
B	66	O	79	b	98	o	111	1	49
C	67	P	80	c	99	p	112	2	50
D	68	Q	81	d	100	q	113	3	51
E	69	R	82	e	101	r	114	4	52
F	70	S	83	f	102	s	115	5	53
G	71	T	84	g	103	t	116	6	54
H	72	U	85	h	104	u	117	7	55
I	73	V	86	i	105	v	118	8	56
J	74	W	87	j	106	w	119	9	57
K	75	X	88	k	107	x	120		
L	76	Y	89	l	108	y	121		
M	77	Z	90	m	109	z	122		

Figura I.1: Tabela ASCII parcial, apenas com os caracteres e códigos usados no problema.

Você, que tem bastante experiência com programação, se ofereceu para ajudar sua amiga Daiane a implementar esse algoritmo em uma linguagem de programação. Para isso seu programa irá receber diversas entradas com o nome de usuário e senha de clientes e, para cada cliente, irá exibir o correspondente nome de usuário e o valor de hash da senha, conforme procedimento criado por ela. Atenção as saídas devem estar ordenadas alfabeticamente pelo nome de usuário.

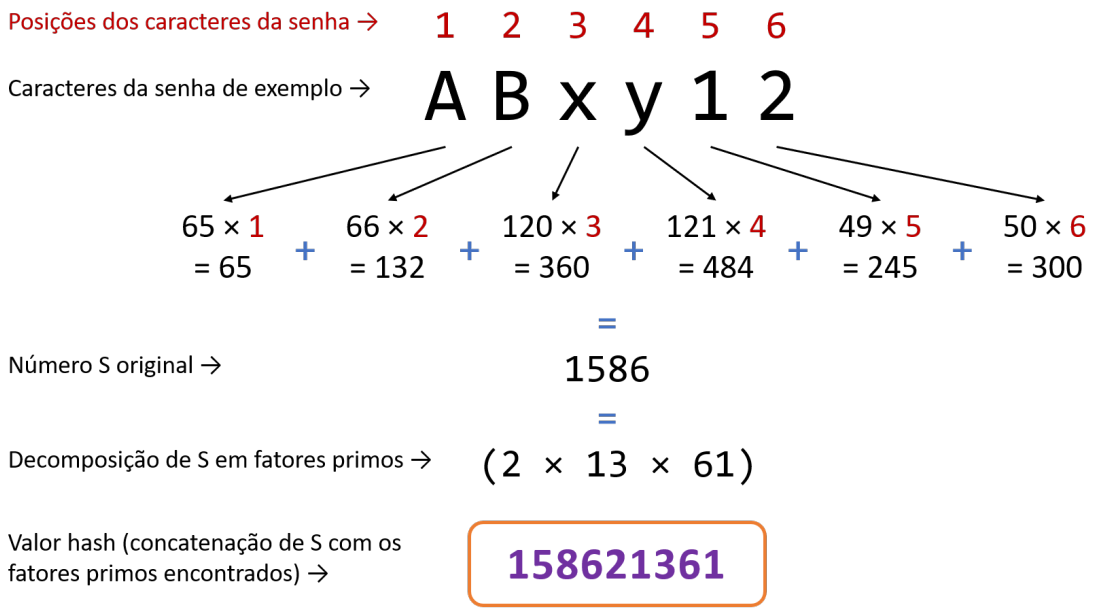


Figura I.2: Ilustração do procedimento para geração do valor hash do algoritmo de Daiane.

Entrada

A entrada é composta por uma quantidade variável de linhas L $(1 \leqslant L \leqslant 2000)$, em que cada linha contém um nome de usuário U e uma senha S. No sistema da empresa de Michelly, U é o próprio nome do cliente, porém, sem espaços e compostos apenas por letras minúsculas e não acentuadas do alfabeto latino (a..z), limitado a 50 caracteres. O sistema de Michelly autoriza senhas com no máximo 100 caracteres, por isso S é composto apenas de letras maiúsculas não acentuadas do alfabeto latino (A..Z), letras minúsculas e não acentuadas do alfabeto latino (a..z) e dígitos decimais (0..9). A entrada é encerrada com a palavra *ACABOU*, com todos os caracteres maiúsculos.

Saída

A saída deverá ser o nome de usuário em uma linha e, na linha seguinte, o respectivo valor hash calculado com base na senha associada ao mesmo usuário. Após cada par de linhas devem existir trinta caracteres '-' (hífen), sem os apóstrofos, seguido de uma quebra de linha, vide exemplos. Note que os usuários deverão ser exibidos ordenados alfabeticamente pelo nome de usuário.

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

`megandenise ABxy12`
`ACABOU`
	`usuario...: megandenise`
	`valor hash: 158621361`
	`------------------------------`

Exemplo de Entrada 2	Exemplo de Saída 2
clara ABC123 ana abc123 bia a1b2c3 duda 1a2b3c andrecavalcante ZZZzzz zequinha 1234567890 helena x9x9 ACABOU	usuario...: ana valor hash: 134221161 ------------------------------ usuario...: andrecavalcante valor hash: 237023579 ------------------------------ usuario...: bia valor hash: 149025149 ------------------------------ usuario...: clara valor hash: 115025523 ------------------------------ usuario...: duda valor hash: 163421943 ------------------------------ usuario...: helena valor hash: 82223137 ------------------------------ usuario...: zequinha valor hash: 2925335513 ------------------------------

Exemplo de Entrada 3	Exemplo de Saída 3
jaentendeualogica QUEROSOHVER ACABOU	usuario...: jaentendeualogica valor hash: 5174213199 ------------------------------

Exemplo de Entrada 4	Exemplo de Saída 4
maria 2AB ACABOU	usuario...: maria valor hash: 37823337 ------------------------------

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema J

Pim

Arquivo fonte: pim.{ c | cpp | java | py }
Autor: Prof. Dr. Leandro Luque (Fatec Mogi das Cruzes)

Num universo paralelo, Silvio Pranktos adora fazer jogos com pegadinhas com seus convidados. Entre eles estão o jogo "É, não e porquê"e o Jogo do Pim. Nesse jogo, os participantes devem contar os números em ordem crescente, substituindo os múltiplos de 4 pela palavra "pim". Por exemplo, os primeiros dez números do jogo seriam:

```
1 2 3 pim 5 6 7 pim 9 10
```

Sivio desafiou os participantes a escreverem uma sequência numérica do "Jogo do Pim"e quer que você crie um programa para verificar se eles acertaram a sequência!

Entrada

A entrada consiste em um único inteiro n ($1 \leqslant N \leqslant 10^4$), indicando o número máximo da sequência.

Saída

Seu programa deve imprimir uma única linha contendo a sequência até o número máximo informado.

Exemplo de Entrada 1	Exemplo de Saída 1
1	1

Exemplo de Entrada 2	Exemplo de Saída 2
10	1 2 3 pim 5 6 7 pim 9 10