MARATONA DE PROGRAMAÇÃO

InterFatecs

Fatec Guaratinguetá

CPS Centro Paula Souza

SÃO PAULO GOVERNO DO ESTADO

Secretaria de Desenvolvimento Econômico

9ª Maratona de Programação InterFatecs

Etapa única: 07/11/2020

Caderno de Questões

Apoio:

FUNDAÇÃO FAT

IBM

FATECOINS

Quanto vale a sua ideia?

treinaweb CURSOS

CENTRO REGIONAL DE TRATAMENTO EM HEMATOLOGIA E ONCOLOGIA

www.interfatecs.com.br

1 Instruções

Este caderno contém 11 problemas – identificados por letras de A até K, com páginas numeradas de 3 até 28. 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) É permitido a consulta a materiais já publicados anteriormente ao dia da competição
   (c) Não é permitido a comunicação com o técnico ou qualquer outra pessoa que não seja a equipe para tirar dúvidas sobre a maratona
   (d) É vedada a comunicação entre as equipes durante a competição, bem como a troca de material de consulta entre elas;
   (e) Cada integrante da equipe poderá utilizar o seu computador/notebook para resolver os problemas, porém, apenas um competidor da equipe ficará responsável pela submissão
   (f) 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 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.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problem A

Abbreviating Names

Source file: abbreviating.{ c | cpp | java | py }
Author: Anderson V. Araujo, Caio Tokunaga e Pedro Flores (UFMS)

Yves is a receptionist at an art gallery and is organizing the opening of a new collection by a famous painter in the city. To save space on the pages and speed up the entry of each guest, he wants to put only an abbreviation of the full names of the guests in the entry list. For each guest, the first and last name must be inserted in full and the others in short form. Help Yves to create a program to accomplish this task.

Input

The entry contains several test cases, each case is a guest's name, containing from 2 to 100 characters, without special characters. The entry ends with EOF.

Output

The program must print the guest list so that it is ordered by the names with the middle names of each guest abbreviated.

Example of Input 1

```
Alda Gusmao Antonia Naves
Ismael Novais Silveira Da Silva De Melo Santos
Prince Uria Noite
Ismael Castelo Gorjao
Jeferson Vargas Capistrano
Layra Bogado
Kyara Ramalho Cavaco
Telmo Lagos Ourique
Ariele Lousa
Nicolae Chaves Fitas
Tania Nascimento Montenegro
Aayush Areosa Sintra
Gastao Saraiva Guilheiro
Ariane
Lina Madureira Saloio
```

Example of Output 1

```
Aayush A. Sintra
Alda G. A. Naves
Ariane
Ariele Lousa
Gastao S. Guilheiro
Ismael C. Gorjao
Ismael N. S. D. S. D. M. Santos
Jeferson V. Capistrano
Kyara R. Cavaco
Layra Bogado
Lina M. Saloio
Nicolae C. Fitas
Prince U. Noite
Tania N. Montenegro
Telmo L. Ourique
```

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema B

Agrupamentos

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

Você está trabalhando num projeto que envolve a exibição em um mapa da localização de prestadores de serviço de uma grande empresa. Entre os desafios do projeto está uma lentidão cada vez maior no carregamento do mapa, dado o grande volume de prestadores exibidos - que hoje passa dos 50 mil. Ainda, a visualização do mapa está muito poluída, pois muitos pontos são exibidos em uma área pequena.

Você fez uma PoC (prova de conceito) com uma biblioteca de agrupamento (*clusterização* - que agrupa pontos próximos) do componente de mapas *frontend* que usa e percebeu que o problema de visualização é resolvido. No entanto, agrupar no *frontend* não resolve o problema de ter que carregar milhares de pontos a partir do *backend*. Portanto, você decidiu implementar um algoritmo de clusterização no *backend*. O algoritmo é uma variante do implementado em diversas bibliotecas de mapas disponíveis no mercado.

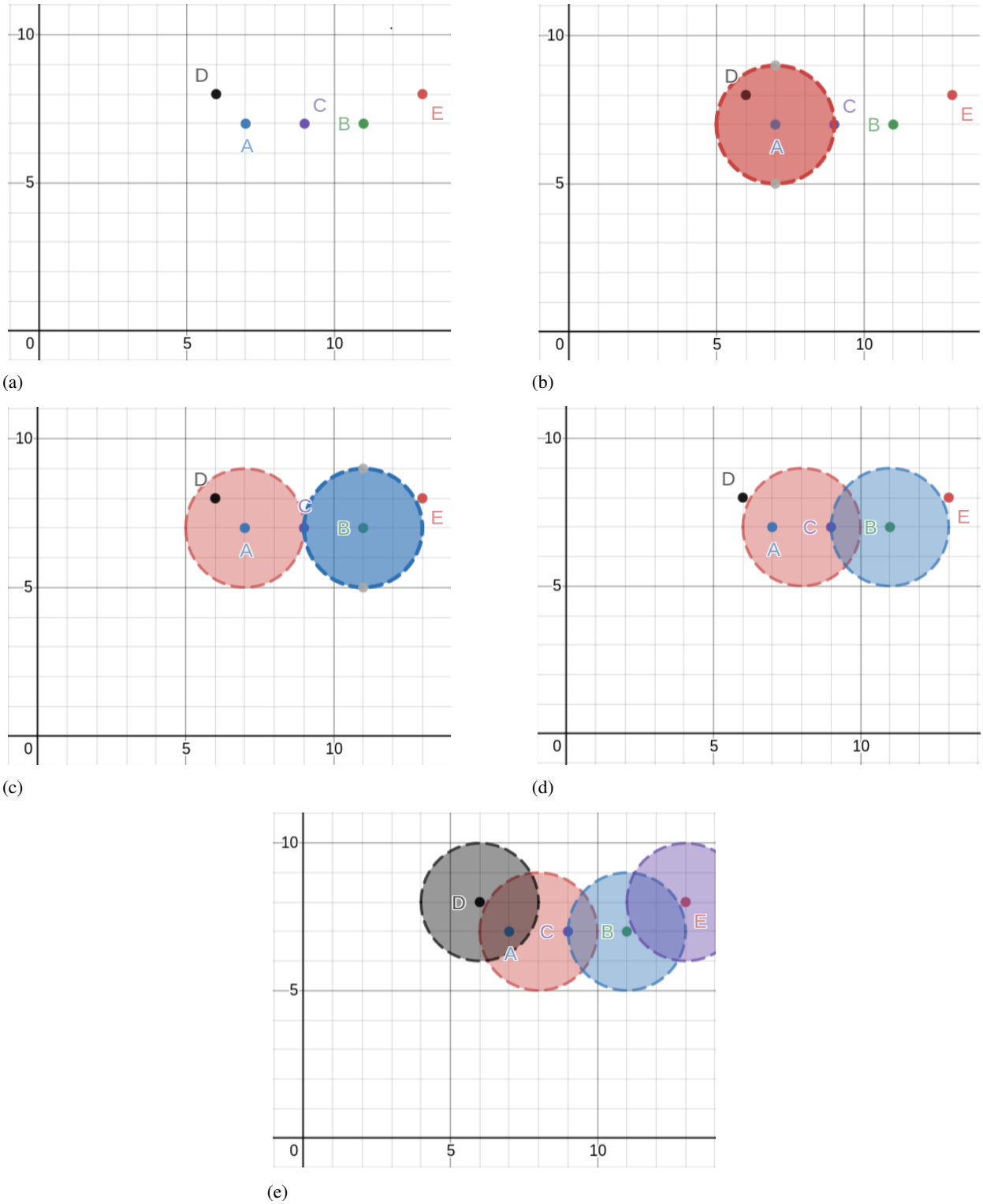
Em resumo, o algoritmo consiste nos seguintes passos:

1. Selecione um ponto (no nosso caso, selecionaremos o primeiro ponto na entrada);
2. Verifique qual é o *cluster* mais próximo:
    - Se não existir *cluster* mais próximo ou se o *cluster* mais próximo estiver a mais de x unidades de distância (considerando a distância euclidiana entre o ponto e o centro do *cluster*), crie um novo *cluster* com o ponto como seu centro;
    - Caso contrário, adicione o ponto ao *cluster* mais próximo e recalcule o centro do *cluster* para ser igual a média aritmética das posições de seus pontos constituintes. Caso um ponto esteja à mesma distância de dois ou mais *clusters*, ele deverá ser adicionado ao *cluster* criado primeiro.
3. Selecione o próximo ponto e repita (2) até passar por todos os pontos.

Considere um mapa com os seguintes pontos: A (7, 7), B (11, 7), C (9, 7), D (6, 8), E (13, 8) - Figura 2a; e raio do *cluster* igual a 2.

1. Começaremos pelo primeiro ponto informado: A (7,7);
2. Como ainda não existem *clusters*, será criado um novo com centro em (7, 7) - Figura 2b;
3. O próximo ponto (B) está à quatro unidades de distância do primeiro *cluster* e, portanto, será criado um novo *cluster* com (11, 7) como centro - Figura 2c;
4. O terceiro ponto (C) está a duas unidades de distância do primeiro e do segundo *clusters*. Como as distâncias são iguais, deve-se inclui-lo no primeiro cluster criado (A). Após adicioná-lo, o novo centro do *cluster* deverá ser atualizado para ((xa + xc) / 2, (ya + yc) / 2) = ((7 + 9) / 2, (7 + 7) / 2) = (8, 7) - Figura 2d;
5. O mesmo processo segue até o fim, resultando em 4 clusters - Figura 2e.

Seu trabalho é implementar o algoritmo explicado.



(a) (b)

(c) (d)

(e)

Figura B.1: Passo a passo do algoritmo

Entrada

A entrada começa com uma linha com dois inteiros: N ($1 \leqslant N \leqslant 100$) indicando o número de prestadores de serviço e R ($1 \leqslant R \leqslant 100$) representando o raio que será usado para os *clusters*. As próximas N linhas contêm coordenadas cartesianas X Y inteiras ($-1000 \leqslant X, Y \leqslant 1000$) representando a localização de um prestador de serviço. A última linha de entrada termina com uma quebra de linha.

Saída

Como saída, imprima uma linha com um número C indicando o número de *clusters* criados pelo algoritmo. Em seguida, imprima C linhas, cada uma com coordenadas cartesianas inteiras (trucando o resultado se necessário) X Y indicando o centro de cada *cluster*, na ordem em que foram criados. A última linha de saída termina com uma quebra de linha.

Exemplo de Entrada 1

```
5 2
7 7
11 7
9 7
6 8
13 8
```

Exemplo de Saída 1

```
4
8 7
11 7
6 8
13 8
```

Exemplo de Entrada 2

```
7 38
-966 -351
-10 899
517 524
-376 -594
-595 -696
362 -943
978 945
```

Exemplo de Saída 2

```
7
-966 -351
-10 899
517 524
-376 -594
-595 -696
362 -943
978 945
```

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema C

Sertanejo

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

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

Sorotim é uma cidade em que todos gostam de música sertaneja e existe uma tradição que é mantida há tanto tempo que ninguém se lembra como começou. Cada nova música é composta por uma dupla. Não há composições feitas por trios, quartetos, muito menos solo (quase considerado crime!). Os Sorotinenses se orgulham muito dessa tradição e se orgulham mais ainda do seu maior compositor: o brilhante Saracura, que compôs mais de 1000 obras em parceria com mais de 20 compositores.

Em sua homenagem, alguns alunos da Fatec da cidade decidiram fazer um TCC sobre o conjunto da obra. Após as pesquisas e levantamentos iniciais, ficou claro que sua vida era cheia de "causos"deliciosos e experiências intensas. Com isso decidiram aprofundar na vida do artista através de entrevistas a serem feitas com seus parceiros diretos e indiretos. Porém, surgiu uma complicação. É tanta gente para entrevistar que de alguma forma a lista precisa ser organizada de um modo lógico. Detectado o problema, o grupo recorreu ao orientador, prof. Miruna, grande apreciador de música sertaneja e grande especialista em algoritmos, nessa ordem. Miruna não perdeu tempo, elaborou uma sistemática para resolver o problema e indicou um novo integrante para o grupo, você.

A sistemática consiste em atribuir um número ao compositor conforme a proximidade dele com o Saracura. Chamaremos este número de "número de proximidade". A numeração será da seguinte forma: quem compôs diretamente com o Saracura receberá o número 1; todos que não foram parceiros diretos, mas compuseram com alguém que possui número 1 recebe o número 2. Quem compôs apenas com quem tem o número 2 recebe o número 3, e assim por diante. Dessa forma, para cada compositor, deve-se identificar qual o parceiro de menor numeração e a ele atribui-se esse número mais 1. Se algum compositor nunca compôs com o Saracura (eita tristeza!), então seu número será infinito.

Graças a essa ideia genial do prof. Miruna, você já tem seu TCC encaminhado. Agora é só escrever o programa.

Entrada

A entrada contém um único caso de teste. A primeira linha contém um número inteiro N ($1 \leqslant N \leqslant 100$), contendo o número de parcerias (duplas sertanejas). As próximas N linhas contém o nome dos dois compositores separados por um espaço em branco, a letra e minúscula e um segundo espaço em branco (" e "). Cada nome de compositor tem no máximo 10 caracteres, o primeiro caractere é maiúsculo e os demais minúsculos. Os nomes na entrada não tem qualquer ordem, ou seja, uma linha pode conter "Saracura e Bianor", enquanto que outra linha pode conter "Bianor e Saracura". Além disso, sabe-se que ao todo estarão presentes nas duplas no máximo 30 compositores, incluindo o Saracura.

Saída

O programa deve produzir na saída L linhas, onde L é o número de compositores distintos presentes no conjunto de entrada, menos o Saracura. Cada linha deve conter o nome do compositor seguido pelo caractere ‘:’ (dois pontos), um espaço em branco e o seu "número de proximidade"com Saracura. Se algum compositor não tiver composto direta ou indiretamente com Saracura, deve-se escrever a palavra ’infinito’ no lugar do

número.

A saída deve estar ordenada por dois critérios: o primeiro é o "número de proximidade"crescente e o segundo é a ordem alfabética do nome do compositor, também crescente.

Não se esqueça de imprimir o final de linha na última linha da saída.

Exemplo de Entrada 1	**Exemplo de Saída 1**
7 Castrinho e Firmino Saracura e Bianor Edmar e Saracura Kauan e Tiberio Bianor e Castrinho Castrinho e Domingos Saracura e Edmar	Bianor: 1 Edmar: 1 Castrinho: 2 Domingos: 3 Firmino: 3 Kauan: infinito Tiberio: infinito

Exemplo de Entrada 2	**Exemplo de Saída 2**
2 Nivaldo e Saracura Pitico e Rancharia	Nivaldo: 1 Pitico: infinito Rancharia: infinito

Exemplo de Entrada 3	**Exemplo de Saída 3**
8 Saracura e Domingos Bianor e Saracura Isnard e Marquito Domingos e Rancharia Saracura e Uirapuru Marquito e Kauan Rancharia e Bianor Uirapuru e Rancharia	Bianor: 1 Domingos: 1 Uirapuru: 1 Rancharia: 2 Isnard: infinito Kauan: infinito Marquito: infinito

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema D

Chinelândia

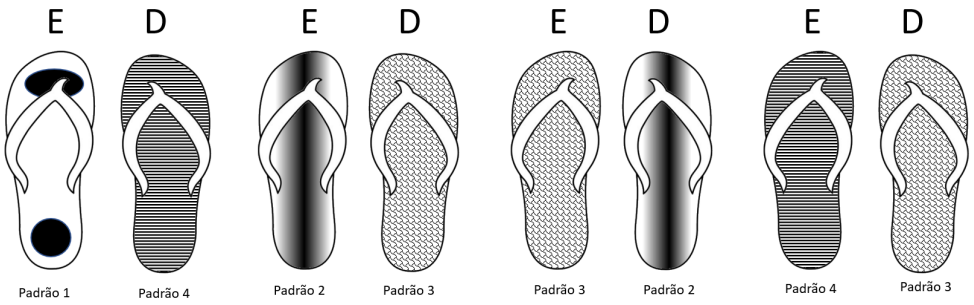
Arquivo fonte: chinelandia.{ c | cpp | java | py }
Autor: Sérgio Luiz Banin (Fatec São Paulo e Fatec São Caetano do Sul)

Os cidadãos de Chinelândia são muito orgulhosos do seu principal produto de exportação, chinelos. A última moda local é usar pares despareados, pé esquerdo uma estampa, pé direito outra estampa. Os cidadãos costumam comprar muitos pares e, para acompanhar essa tendência, as lojas passaram a vender os pares já despareados atados por um grampo que não permite a separação, a menos que sejam quebrados. E é proibido por lei quebrar um grampo antes de adquirir o par. Ocorre que, às vezes, alguém compra um novo par despareado e acaba adquirindo um pé repetido, ou seja, com uma estampa que já possui. Disso surgiu a ideia de se organizar um evento, a feira de trocas de pés repetidos.

Para participar da feira, cada cidadão precisa elaborar uma lista dos pés repetidos que possui, porém, eles não são bons com listas. Então, alguns cidadãos eminentes se reuniram e decidiram pedir sua ajuda.

Isso mesmo! Você, programador, pode ajudá-los escrevendo um programa que será distribuído à população. Neste programa o cidadão informará os pares que comprou e o programa avisará quais são os pés que tem unidades repetidas e, portanto, disponíveis para troca. O que vai facilitar sua vida é o fato de que cada desenho de estampa recebe um número inteiro único que é chamado de "padrão".

A título de exemplo, veja que na situação a seguir o pé direito/padrão 3 tem dois exemplares o que significa que há um repetido e disponível para troca.



Ahh, programador, você não precisa se preocupar com o tamanho do chinelo. Os Chinelandeses são descendentes dos Hobbits e tem uma característica genética curiosa, todos calçam o número 59 do padrão humano.

Entrada

A entrada contém um único caso de teste. Na primeira linha há um número inteiro que é quantidade de pares de chinelos ($1 \leqslant NPC \leqslant 2000$) que o cidadão comprou. Em seguida há NPC linhas contendo dois números inteiros - E D - separados por um espaço em branco. Tais números representam pares de chinelos: o primeiro número da linha, E, é o padrão do pé esquerdo e o segundo, D, é o padrão do pé direito. Garante-se que em nenhuma linha haverá dois números de padrão iguais, ou seja, garante-se $E \neq D$.

Saída

A saída deve conter uma linha para cada caso de ocorrência de repetição para a qual será exibido o número do padrão, uma letra para o pé (E ou D - maiúscula) e quantas unidades estão disponíveis para troca. As três informações devem estar separadas por um espaço em branco e todas as linhas devem conter o final de linha, inclusive a última. As linhas devem estar ordenadas por número do padrão como primeiro critério e pela letra do pé como segundo critério de ordenação.

Considere que algum cidadão pode não ter nada para trocar. Neste caso, o programa deve gravar na saída o texto: SEM TROCAS DESTA VEZ, em letras maiúsculas e com o final de linha.

Exemplo de Entrada 1

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

Exemplo de Saída 1

```
3 D 1
```

Exemplo de Entrada 2

```
9
11 6
5 11
6 3
2 8
12 6
11 4
2 6
7 3
3 1
```

Exemplo de Saída 2

```
2 E 1
3 D 1
6 D 2
11 E 1
```

Exemplo de Entrada 3

```
9
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
9 10
```

Exemplo de Saída 3

```
SEM TROCAS DESTA VEZ
```

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema E

Escalonamento FIFO

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

Autor: Allbert Velleniche de Aquino Almeida (Fatec Guaratinguetá)

Durante as aulas de Sistemas Operacionais, o Professor Geraldinho explicou aos alunos os diversos algoritmos de escalonamento utilizados em Sistemas Operacionais. Dentre os algoritmos estudados, um dos mais simples é o algoritmo FIFO (*First In First Out*), de acordo com o qual cada processo é executado de acordo com sua ordem de chegada na fila. O Professor Geraldinho desafiou os alunos a desenvolverem um programa para calcular o tempo médio de espera de cada processo (tempo que o processo esperou até ser executado) e o tempo médio de *turnaround* de cada processo (tempo que leva desde de criação até a finalização do processo).

O seu programa deverá ler a entrada padrão de uma lista de processos com seus respectivos tempos de chegada e de duração do processo e deverá imprimir na saída os valores para as seguintes métricas: Tempo Médio de Espera (TME) e Tempo Médio de *Turnaround* (TMT).

Processo 1 (Tempos):
Chegada: 0 e Duração: 20
Processo 2 (Tempos):
Chegada: 0 e Duração: 10
Processo 3 (Tempos):
Chegada: 4 e Duração: 6
Processo 4 (Tempos):
Chegada: 4 e Duração: 8

O TME é calculado pelo instante em que o processo foi iniciado, subtraído o tempo de chegada: P1: 0, P2: 20, P3: 26, P4: 32. TME = (0+20+26+32)/4 = 19,5.

O TMT é calculado pelo instante em que o processo foi finalizado, subtraído o tempo de chegada: P1: 20, P2: 30, P3: 32, P4: 40. TMT = (20+30+32+40)/4 = 30,5.

Entrada

A entrada é composta por um número inteiro que indica a quantidade de processos N ($1 \leqslant N \leqslant 100$). As próximas N linhas contém uma série de pares de números inteiros separados por um espaço em branco indicando o tempo de chegada TC e a duração D de cada processo. ($0 \leqslant TC, D \leqslant 60$).

Saída

A saída é composta por uma linha contento a sigla TME, seguida de ":"(dois pontos) e o valor do Tempo Médio de Espera, conforme o exemplo "TME:TE"(sem aspas), onde TE é o valor da métrica. Outra linha contendo a sigla TMT, seguida de ":"(dois pontos), o valor do Tempo Médio de Turnaround "TMT:TT"(sem aspas), onde TT compreende o valor da métrica, ambos valores (TE e TT) com uma casa decimal. Finalize com uma quebra de linha.

Exemplo de Entrada 1

```
4
0 20
0 10
4 6
4 8
```

Exemplo de Saída 1

```
TME:19.5
TMT:30.5
```

Exemplo de Entrada 2

```
3
0 8
2 4
4 10
```

Exemplo de Saída 2

```
TME:4.7
TMT:12.0
```

Exemplo de Entrada 3

```
6
0 5
4 8
6 10
8 5
10 20
12 20
```

Exemplo de Saída 3

```
TME:12.8
TMT:24.2
```

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema F

Escola de Música

Arquivo fonte: escola.{ c | cpp | java | py }
Autor: Allbert Velleniche de Aquino Almeida (Fatec Guaratinguetá)

A escola de música Sustenido passou por sérios problemas financeiros diante do cenário pandêmico causado pela Covid-19. Muitos alunos cancelaram matrículas e o impacto financeiro para a escola e seus funcionários foram devastadores. Porém, a Sustenido está superando esse momento de crise e tentando se reinventar. Um dos cenários traçados pela escola é direcionar suas ações de marketing com objetivo de conseguir mais alunos na mesma faixa etária que os alunos matriculados atualmente, formando assim uma rede de colaboração.

A escola enviou um formulário para todos os seus alunos solicitando o preenchimento da idade, porém, ela se viu perdida em meio as informações enviadas e não sabe como calcular. Por esse motivo, a escola Sustenido precisa de sua ajuda para determinar o valor central das idades dos alunos.

Moda, média e mediana são medidas obtidas de conjuntos de dados, que podem ser usadas para representar todo o conjunto. A tendência dessas medidas é resultar em um valor central. Por essa razão, elas são chamadas de medidas de centralidade.

Moda: É chamado de moda o dado mais frequente de um conjunto. Veja um exemplo: em uma escola, os oito alunos da turma "A" possuem as seguintes idades: $\{12, 13, 13, 13, 11, 12, 14, 11\}$ anos. Perceba que a idade 13 repete-se 3 vezes e nenhuma idade aparece mais que essas, por isso a moda é 13. A moda pode conter mais que um conjunto, quando elas se repetem o mesmo número de vezes, por exemplo: $\{12, 14, 13, 11, 12, 13\}$. Nesse caso, a idade 12 e 13 se repete o mesmo número de vezes. Por isso, o conjunto possui duas modas (12 e 13) e é chamado de bimodal.

Média, mais precisamente chamada de média aritmética simples, é o resultado da soma de todas as informações de um conjunto de dados dividida pelo número de informações que foram somadas. Exemplo: o conjunto $\{14, 15, 13\}$ tem soma 42 e média 14.

Mediana é o valor que separa a metade maior e a metade menor de uma amostra, uma população ou uma distribuição de probabilidade. Por isso, é importante ordenar o conjunto de dados em ordem crescente. Em termos mais simples, mediana pode ser o valor do meio de um conjunto de dados. No conjunto de dados $\{11, 12, 12, 13, 14, 14, 15\}$, por exemplo, a mediana é 13. Se houver um número par de observações, não há um único valor do meio, então, a mediana é definida como a média dos dois valores do meio.

Será que você consegue ajudar a escola a encontrar o valor central das idades???

Entrada

A entrada é composta por vários casos de teste. Cada caso de teste se inicia com uma linha contendo um inteiro N representando a quantidade de alunos ($1 \leqslant N \leqslant 600$). Em seguida, para cada valor de N, existe uma linha com a idade ID de um determinado aluno ($2 \leqslant ID \leqslant 18$). O programa continuará sendo executado até encontrar o fim de arquivo (EOF - End Of File).

Saída

Para cada caso de teste, o programa deverá imprimir o resultado da moda no seguinte formato "MODA=MO"(sem aspas), onde MO são os valores da moda do conjunto. Se houver mais que um valor, deverá ser separado por vírgulas, em ordem crescente. Na próxima linha, o resultado da média com o valor com até duas casas decimais, no formato "MEDIA=M"(sem aspas e acentuação), onde M é o resultado da média. E, por último, a mediana contendo o valor também com duas casas decimais, no formato "MEDIANA=ME"(sem aspas), onde ME representa o valor do cálculo da mediana.

Exemplo de Entrada 1	Exemplo de Saída 1
4 12 14 12 13 5 13 12 12 11 11 6 11 11 9 9 4 4	MODA=12 MEDIA=12.75 MEDIANA=12.50 MODA=11,12 MEDIA=11.80 MEDIANA=12.00 MODA=4,9,11 MEDIA=8.00 MEDIANA=9.00

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema G

Festa

Arquivo fonte: festa.{ c | cpp | java | py }
Autor: Anderson V. Araujo, Gabriel Rocha e Pedro Novais (UFMS)

Teobaldo é a pessoa mais popular da faculdade. Lá, ele é chamado de Teo por todos. Durante a pandemia, ele anda meio chateado, pois não está podendo ver seus amigos e com eles organizar suas famosas festas. Então, durante o tempo que está em casa, decidiu que assim que essa situação se resolver ele vai fazer a maior festa que a faculdade já viu! Uma incrível festa para que possa rever seus amigos e todos possam se divertir.

O problema é que, sendo tão popular, Teobaldo conhece muitas pessoas e algumas dessas não gostam umas das outras e jamais permaneceriam no mesmo ambiente. Assim, ele decidiu que faria uma festa com dois ambientes separados dividindo as pessoas que não se gostam, para que todos os alunos pudessem comparecer.

Agora, você, amigo do Teo, exímio programador, vai ajudá-lo a definir se, sabendo todos que participariam da festa e todas as inimizades, seria realmente possível realizar a festa dividindo os convidados desta forma.

Entrada

A entrada consiste em um inteiro N ($1 \leq \mathrm{N} \leq 1000$) que representa o número de pessoas que virão para a festa. Em cada linha a seguir haverá dois inteiros U e V ($1 \leq \mathrm{U}, \mathrm{V} \leq \mathrm{N}$) que representam as inimizades entre as pessoas da festa (as pessoas U e V jamais ficariam no mesmo ambiente), o fim da entrada se dá por EOF.

Saída

Seu programa deve imprimir a resposta que você dará a Teobaldo sobre ser ou não possível realizar a festa da forma como ele pensou. Então, se for possível. imprima "FESTA!". Caso contrário imprima "Lascou...".

Exemplo de Entrada 1	Exemplo de Saída 1
8 1 6 1 5 2 5 2 7 3 6 3 8 3 4	FESTA!

Exemplo de Entrada 2	Exemplo de Saída 2
6 1 2 1 3 1 4 2 3	Lascou...

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema H

Quociente Eleitoral

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

Autor: Allbert Velleniche de Aquino Almeida (Fatec Guaratinguetá)

Em município não muito distante, chamado Nlogônia, nessa última eleição, a população ficou indignada com um candidato a vereador que foi menos votado para o cargo que venceu outro vereador mais votado. Essa população não entendeu o tal quociente eleitoral e pediu ajuda a um desenvolvedor para calcular o número de candidatos cada partido terá direito.

Para o cálculo do quociente eleitoral são utilizados os votos válidos. Para isso, obtém-se o número de eleitores que foram votar, excluindo os votos brancos e nulos. Em seguida, calcula-se o quociente eleitoral dividindo-se os votos válidos pela quantidade de lugares a preencher, desprezando-se a fração se menor que 0,5 e se igual ou superior a 0,5 arredonda-se para 1.

Para exemplificar, vamos tomar como base que tenhamos: votos válidos (46.322) ÷ número de vagas (17) = 2.724,8 = quociente eleitoral (2.725). A seguir, calcula-se o quociente partidário, dividindo-se a quantidade de votos de cada partido pelo quociente eleitoral. Despreza-se a fração, qualquer que seja. Os partidos que não alcançaram quociente partidário (menor que 1), não concorrem à distribuição de lugares.

Partidos	Votação	Quociente eleitoral	Quociente partidário
1	15.992	÷ 2.725 = 5,8	= 5
2	12.811	÷ 2.725 = 4,7	= 4
3	7.025	÷ 2.725 = 2,5	= 2
4	6.144	÷ 2.725 = 2,2	= 2
5	2.237	÷ 2.725 = 0,8	= 0*
6	2.113	÷ 2.725 = 0,7	= 0*
			Total = 13 (sobram 4 vagas a distribuir)

Caso a soma do quociente partidário seja menor que o número de vagas, essa diferença precisa ser distribuída (VAGAS = quantidade de vagas - soma do quociente partidário). Os partidos que não conseguiram ao menos uma vaga (quociente partidário=0) não entrarão na distribuição das vagas. Para distribuição das vagas, calcula-se a média de cada partido que obteve quociente, dividindo a quantidade de votos pelo quociente partidário obtido acrescido de 1.

Exemplo: Partido 1 -> 15.992/(5+1) = 2665,3

O partido que obter a maior média, tem seu quociente partidário aumentado em 1. Repete-se a operação até que as vagas sejam todas distribuídas (vagas =0).

Como estamos no ano de eleição, seria uma ótima oportunidade para ajudar os eleitores entenderem essa relação do quociente eleitoral.

Entrada

Cada caso de teste contém dois números inteiros N ($1 \leqslant N \leqslant 20$) e NP ($1 \leqslant NP \leqslant 20$), que correspondem, respectivamente, ao número de vagas e ao número de partidos. As próximas NP linhas terão um inteiro NVP, correspondente ao número de votos válidos obtidos por cada partido ($1 \leqslant NVP \leqslant 100.000$).

Saída

A saída deverá conter uma linha com dois valores inteiros. O primeiro indicando o total de votos válidos TVV e o segundo correspondente ao quociente eleitoral QE. Nas próximas N linhas, deverá ser exibida a quantidade do quociente partidário no seguinte formato: "Partido N: V"(sem aspas), onde N representa o número do partido (sequencialmente e iniciado em 1) e V representa o número de vagas do quociente partidário. Finalize com uma quebra de linha.

Exemplo de Entrada 1

```
17 6
15992
12811
7025
6144
2237
2113
```

Exemplo de Saída 1

```
46322 2725
Partido 1: 7
Partido 2: 5
Partido 3: 3
Partido 4: 2
Partido 5: 0
Partido 6: 0
```

Exemplo de Entrada 2

```
13 5
23987
19281
16544
9573
4936
```

Exemplo de Saída 2

```
74321 5717
Partido 1: 5
Partido 2: 4
Partido 3: 3
Partido 4: 1
Partido 5: 0
```

Exemplo de Entrada 3

```
19 4
57020
90220
25130
5200
```

Exemplo de Saída 3

```
177570 9346
Partido 1: 6
Partido 2: 10
Partido 3: 3
Partido 4: 0
```

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema I

Reforma

Arquivo fonte: reforma.{ c | cpp | java | py }
Autor: Érico de Souza Veriscimo (IFSP - São Miguel Paulista)

Marcus está aproveitando que tem mais tempo em casa para terminar uma reforma. Para terminar a obra, basta colocar os rodapés utilizando os pedaços de cerâmica que sobraram de outra obra. Estes pedaços são todos da mesma altura, variando apenas no comprimento. Entretanto, Marcus não tem ferramentas apropriadas para realizar cortes, ou seja, deve utilizar as peças do tamanho que estão.

Como ele é seu amigo, você se prontificou a fazer um algoritmo que, dado o tamanho em centímetros do local que se deve colocar os rodapés, bem como a quantidade e tamanho dos pedaços de cerâmica, define se é possível ou não finalizar a obra, lembrando que ele não tem ferramenta para cortar a cerâmica em peças menores.

Entrada

A entrada começa com uma linha com dois inteiros, N ($1 \leqslant \mathrm{N} \leqslant 10^5$) indicando o tamanho em centímetros que se deve colocar de rodapés, e Q ($1 \leqslant \mathrm{Q} \leqslant 10^3$) representando a quantidade de peças. A próxima linha contém Q valores separados por espaço representando o comprimento de cada peça.

Saída

Como saída, imprima uma linha contendo a palavra SIM, se for possível finalizar a obra, ou NAO se não for possível.

Exemplo de Entrada 1	**Exemplo de Saída 1**
`125 5` `10 24 100 4 125`	`SIM`

Exemplo de Entrada 2	**Exemplo de Saída 2**
`170 4` `250 110 50 10`	`SIM`

Exemplo de Entrada 3	**Exemplo de Saída 3**
`210 4` `250 100 50 10`	`NAO`

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema J

Acessibilidade

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

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

Acessibilidade é um assunto de extrema importância. Todas as edificações, sejam públicas ou privadas, que se destinam ao uso coletivo, devem garantir às pessoas com deficiência condições de acessibilidade a todos os serviços e em todas as suas dependências.

No Brasil, o Estatuto da Pessoa com Deficiência (Lei Brasileira de Inclusão da Pessoa com Deficiência) define as normas gerais, e a Norma Técnica NBR 9050, editada pela ABNT, dispõe especificamente sobre os meios de acessibilidade em edificações, mobiliários, espaços e equipamentos urbanos. Essa norma traz todas as recomendações e cuidados necessários na hora de projetar uma construção para que seja acessível, tais como: medidas, distâncias necessárias, proteção contra quedas, altura correta para o alcance (lateral e frontal) do usuário de cadeira de rodas, entre outros.

Um dos equipamentos fundamentais para acessibilidade do cidadão cadeirante é a rampa, que consiste de um plano inclinado cuja máxima declividade é definida pela NBR 9050. Para ser acessível ao cadeirante autônomo (que depende apenas de suas próprias forças para realizar a subida), uma rampa precisa contemplar diversos detalhes e seu projeto deve ser executado por profissional de engenharia habilitado e devidamente treinado nos termos da Norma.

O primeiro passo é calcular sua inclinação, cujo valor máximo deve ser de 8,33% (relação 1:12 entre altura e comprimento). Tal cálculo é feito com a fórmula:

$i = H \times 100 \div C$

onde:

i é a inclinação da rampa expressa em porcentagem;

H é a altura do desnível;

C é o comprimento da projeção horizontal da rampa.

O segundo parâmetro fundamental é a largura da rampa para circulação em linha reta, cujo valor mínimo admissível é de 0,80 metros.

Rampas onde haja espaço para atender simultaneamente os dois requisitos $i \leqslant 8,334\%$ e $L \geqslant 0,80m$ são consideradas como Projeto Simples. Quando pelo menos um desses dois requisitos não for atendido trata-se de um Projeto Especial, que demanda um profissional mais experiente.

Nesta tarefa, você deve elaborar um programa de computador que irá ler as dimensões principais de altura, comprimento e largura e, em seguida, calcular a inclinação, bem como definir se o projeto é simples ou especial.

Entrada

A entrada contém diversos casos de teste. Em cada linha, há três números reais separados por um caractere em branco. O primeiro é a altura H ($H > 0$) da rampa. O segundo é o comprimento da projeção horizontal C ($C > 0$). O terceiro é a Largura L ($L > 0$) da rampa. Para indicar o fim da entrada de dados, a última

linha três valores iguais a 0.0. Utilize números reais de precisão dupla.

Saída

A saída deve conter uma linha para cada linha da entrada informando com letras maiúsculas se é um PROJETO SIMPLES ou um PROJETO ESPECIAL. Ao final de cada linha deve ser impresso o final de linha, inclusive na última.

Exemplo de Entrada 1	Exemplo de Saída 1
0.8 16.0 1.2 1.0 25.5 0.7 1.2 30.0 1.2 1.5 32.5 1.2 1.5 20.0 0.9 1.8 20.0 0.9 1.8 30.0 0.9 3.0 40.0 1.0 3.0 36.0 1.0 3.0 30.0 1.0 0.0 0.0 0.0	PROJETO SIMPLES PROJETO ESPECIAL PROJETO SIMPLES PROJETO SIMPLES PROJETO SIMPLES PROJETO ESPECIAL PROJETO SIMPLES PROJETO SIMPLES PROJETO SIMPLES PROJETO ESPECIAL

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema K

LuBank

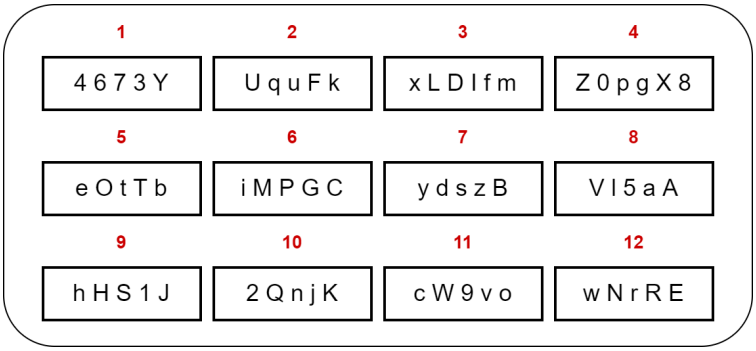
Arquivo fonte: lubank.{ c | cpp | java | py }
Autor: Lucio Nunes de Lira (Fatec São Paulo)

LuBank é um banco digital que pretende melhorar a segurança para seus clientes e um ponto crítico encontrado por seus profissionais de TI está relacionado com a forma como é realizado o $login$ no aplicativo bancário para $smartphones$. Atualmente o login é feito com a inserção do nome de usuário e da senha em campos específicos do aplicativo por meio de um teclado virtual, em que cada tecla corresponde a um único caractere. O problema desta abordagem é que alguém mal-intencionado pode furtivamente acompanhar a digitação da senha, anotá-la e acessar a conta indevidamente.

Uma equipe formada por alunos da Fatec propôs uma solução baseada na implantação de um novo teclado que evita a digitação direta dos caracteres da senha e, consequentemente, dificulta a visualização e interpretação por terceiros. O novo teclado terá apenas 12 teclas, onde 10 terão 5 caracteres associados a cada uma e 2 teclas terão 6 caracteres associados a cada uma. Essa configuração permitirá senhas compostas pelos dez dígitos da base decimal e pelas 52 letras do alfabeto latino não acentuado, sendo 26 maiúsculas e 26 minúsculas.

Nesta proposta, o cliente continua utilizando o teclado comum para inserir seu nome de usuário e o novo teclado para inserir a senha. Com o novo teclado, o cliente precisa pressionar as teclas na sequência correspondente à sua senha. Portanto, a primeira tecla deverá ser aquela que contém o primeiro caractere de sua senha, a segunda tecla deverá ser aquela que contém o segundo caractere de sua senha e assim por diante. Note que as teclas podem ser pressionadas mais de uma vez, inclusive para caracteres diferentes da senha. É importante lembrar que todos os caracteres estão distribuídos entre as teclas e que não há associação de um mesmo caractere a mais de uma tecla, logo há apenas uma sequência correta para representar cada senha.

A equipe também se preocupou em destacar que a configuração do novo teclado será dinâmica, ou seja, irá variar os caracteres associados a cada tecla de acordo com alguns critérios ainda indefinidos. A Figura 1 ilustra um exemplo de configuração do novo teclado. Note que as teclas sempre serão numeradas de 1 a 12.



O seu objetivo, como integrante desta equipe prodígio de alunos, é desenvolver um programa que receba como entrada: (I) uma possível configuração do novo teclado; (II) os logins dos clientes do banco, compostos unicamente de usuário e senha e; (III) a sequência de tentativas de login feitas durante o período em que vigorou a configuração dada do teclado, compostas apenas do nome de usuário e da sequência de teclas pressionadas para representar a senha do respectivo usuário.

O seu programa deve exibir, a cada tentativa de login, o nome do usuário inserido e uma mensagem indicativa

de acordo com a situação: (I) acesso concedido: caso o usuário exista e a sequência de teclas corresponda à sequência válida para a senha do login daquele usuário; (II) acesso negado: caso o usuário exista, mas a sequência de teclas não corresponda à sequência válida para a senha de seu login; (III) usuário bloqueado: caso o usuário exista e tenham sido feitas três tentativas inválidas para o login deste usuário; (IV) usuário inexistente: caso o usuário não exista entre os logins de clientes do banco.

O Lubank exigiu mais algumas características para seu programa: (a) cada login correto de um usuário zera seu contador de tentativas, permitindo mais três tentativas futuras de login; (b) uma vez que um usuário é bloqueado não será mais desbloqueado, mesmo com futuras tentativas com login correto.

Entrada

A entrada inicia com doze linhas (em nenhuma ordem pré-determinada) que representam a configuração das teclas do novo teclado, sendo que cada linha é iniciada com o número da tecla e seguida dos caracteres associados àquela tecla, os valores são separados por ponto e vírgula. As próximas L $(0 \leqslant L \leqslant 50)$ linhas são os logins dos clientes do LuBank, compostos unicamente de uma string U $(3 \leqslant tamanho(U) \leqslant 50)$ representando o nome de usuário (podendo ser formado pelos mesmos caracteres permitidos nas senhas), um ponto e vírgula e outra string S $(3 \leqslant tamanho(S) \leqslant 60)$ representando a senha do usuário; os logins são encerrados quando for inserido um login em que U e S sejam iguais à string 'fim'. As próximas linhas representam as tentativas de login realizadas durante a vigência da configuração dada para o novo teclado e na mesma ordem em que foram tentadas; cada uma dessas linhas é composta primeiramente pelo nome do usuário e, em seguida, pela sequência de teclas pressionadas para representar a respectiva senha, sendo que os valores são separados por ponto e vírgula; as tentativas de login são finalizadas com o fim do arquivo de entrada.

Saída

A saída é uma mensagem indicando o nome de usuário, pós-fixado de dois pontos, um espaço e o resultado de cada tentativa de login, sendo os possíveis resultados correspondentes àqueles descritos no texto e com formatação idêntica ao ilustrado nos casos de exemplo. Toda linha de saída deve ser finalizada com uma quebra de linha.

Exemplo de Entrada 1	Exemplo de Saída 1
7;y;d;s;z;B 9;h;H;S;1;J 10;2;Q;n;j;K 11;c;W;9;v;o 12;w;N;r;R;E 1;4;6;7;3;Y 2;U;q;u;F;k 4;Z;0;p;g;X;8 5;e;O;t;T;b 3;x;L;D;I;f;m 8;V;l;5;a;A 6;i;M;P;G;C BiaCavalcante;abcXYZ AnaMaria;12345 duda2;AbCdEfG fim;fim BiaCavalcante;8;5;11;4;1;4 BiaCavalcante;8;7;6;3;7;7 AnaMaria;9;10;1;1;8 BiaCavalcante;8;7;6;3;7;7 AnaMaria;9;10;1;1;8 BiaCavalcante;8;7;6;3;7;7 duda3;8;5;6;7;12;3;6;9 duda2;8;5;6;7;12;3;6;9 duda2;8;5;6;7;12;3;6 BiaCavalcante;8;5;11;4;1;4 duda2;8;5;6;7;12;3;6;12 duda2;8;5;6;7;12;3;6;10 duda2;8;5;6;7;12;3;6	BiaCavalcante: acesso concedido BiaCavalcante: acesso negado AnaMaria: acesso concedido BiaCavalcante: acesso negado AnaMaria: acesso concedido BiaCavalcante: usuario bloqueado duda3: usuario inexistente duda2: acesso negado duda2: acesso concedido BiaCavalcante: usuario bloqueado duda2: acesso negado duda2: acesso negado duda2: acesso concedido

Exemplo de Entrada 2	Exemplo de Saída 2
9;N;5;D;L;G 12;U;X;a;n;R 2;Y;c;S;K;u 3;4;T;B;b;O 5;H;E;2;W;9 7;F;z;r;0;g 6;i;m;A;1;v 1;e;7;I;d;h 11;3;P;j;M;o;Z 8;x;V;q;w;k;C 4;f;s;8;l;6 10;t;y;Q;J;p goku;DragonBall FreEza;imortals2 bulma;7EsFeRaS fim;fim goku;9;7;12;7;11;12;3;12;4;4 bulma;5;4;7;1;12;12 bulma;1;5;4;7;1;12;12;2 FreEza;6;6;11;7;10;12;4;4;5 FreEza;6;6;11;7;10;12;4;4 FreEza;6;6;11;7;10;12;4;3 FreEza;6;6;11;7;10;12;4;1 FreEza;6;6;11;7;10;12;4;4;5	goku: acesso concedido bulma: acesso negado bulma: acesso concedido FreEza: acesso concedido FreEza: acesso negado FreEza: acesso negado FreEza: usuario bloqueado FreEza: usuario bloqueado