XXIII SEMANA
DE INFORMÁTICA

Maratona de Programação Seletiva Interna 2018 da UFV

Caderno de Problemas

UFV, 24 de agosto de 2018

Instruções:

- Este caderno contém 12 problemas: as páginas estão numeradas de 1 a 14, não contando esta página de rosto. Verifique se o caderno está completo.
- Em todos os problemas, a entrada de seu programa deve ser lida da *entrada padrão*. A saída deve ser escrita na *saída padrão*.

Problema 1. Yes

Arquivo-fonte: `yess.c`, `yess.cpp` ou `yess.java`

Age of Empires (AoE) 2 é um divertido jogo de estratégia criado pela Microsoft há quase 20 anos (mas ainda muito popular). Tal jogo é bastante popular entre ex-alunos do DPI e Paulo Lucio, um desses ex-alunos, é um dos maiores jogadores de AoE do universo.

Uma de suas estratégias consiste em criar um cavalo (um tipo de unidade do jogo) e enviá-lo para passear pela base do inimigo. À medida em que a base é revelada, ele conta a quantidade de cada categoria de unidade vista.

Na versão de AoE usada nesta questão há as seguintes categorias de unidades, cada uma com dois tipos de unidades (listas após seu nome):

- infantaria: soldado, piqueiro
- cavalaria: cavalo, camelo
- capivaria: capivara, carrapato
- cerco: catapulta, trebuchet
- artilharia: arqueiro, escaramucador

Por motivos de eficiência, sempre que o Paulo Lucio vê uma unidade, ele anota o nome dela em um documento de texto. Dadas as anotações do Paulo Lucio, imprima a quantidade total de unidades vistas de cada categoria (use a mesma ordem da listagem acima!).

Entrada

A entrada começa com um inteiro N ($0 \leq N \leq 1.000.000$), que indica a quantidade de unidades vistas. A seguir há N linhas, cada uma contendo um string com o nome de uma unidade (apenas as unidades listadas acima aparecem na entrada).

Saída

Seu programa deve escrever a lista de categorias de unidades, na mesma ordem descrita acima, e logo após o nome de cada categoria, um espaço em branco e a quantidade total de unidades da categoria, relatada pelo Paulo.

Exemplos

Entrada	Saída
7 soldado cavalo catapulta camelo soldado soldado arqueiro	infantaria 3 cavalaria 2 capivaria 0 cerco 1 artilharia 1

Problema 2. No

Arquivo-fonte: `noo.c`, `noo.cpp` ou `noo.java`

No jogo Age of Empires, cada time possuiu um exército, há diversos tipos de integrantes no exército e cada um deles possui uma excelente vantagem em relação a alguns outros integrantes. Por exemplo, os escaramuçadores (guerreiros com lanças e escudos) são excelentes contra arqueiros, e os arqueiros são excelentes contra atiradores a cavalos.

Dadas as vantagens de cada tipo de integrante de exército disponível no jogo e informações sobre os integrantes do exército inimigo, deseja-se saber o número mínimo de unidades diferentes que um exército deve ter de modo que se tenha pelo menos um guerreiro bom contra qualquer guerreiro inimigo.

Por exemplo, suponha estejam disponíveis as seguintes 6 unidades:

- escaramuçador (bons contra: arqueiro e cavalo)
- cavalo (bons contra: arqueiro)
- capivara (bons contra: gramador)
- arqueiro (bons contra: atirador e arqueiro)
- piqueiro (bons contra: cavalo)
- camelo (bons contra: atirador)

Se o inimigo possui apenas unidades do tipo "arqueiro", será necessário no mínimo um tipo de unidade para combatê-lo (basta ter "escaramuçador" ou "arqueiro"). Por outro lado, se o inimigo possuir "arqueiro", "cavalo" e "atirador", o número mínimo de unidades diferentes para combatê-lo será 2 (basta ter "escaramuçador" e "camelo", ou "escaramuçador" e "arqueiro").

Entrada

A entrada começa com uma linha contendo dois inteiros N e M, respectivamente o número de unidades distintas disponíveis no jogo e o número de unidades distintas do inimigo ($1 \leq N \leq 15$, $M \leq N$).

A seguir, haverá N linhas descrevendo cada unidade disponível no jogo. Cada uma dessas linhas começa com um string x (representando o nome da unidade). A seguir, haverá um inteiro K ($0 \leq K \leq N$) e então K strings descrevendo as unidades (sem repetição) contra as quais x é forte. As unidades descritas serão únicas e todas unidades mencionadas na entrada serão descritas em alguma linha. As unidades do exemplo acima são meramente ilustrativas, as unidades podem ter qualquer nome, sempre um string com letras minúsculas, de 1 a 15 caracteres.

Finalmente, haverá uma linha contendo M strings, informando as unidades que o inimigo possui.

Saída

Escreva a quantidade mínima de unidades distintas que um exército deve ter de modo que sempre se tenha pelo menos um guerreiro bom contra qualquer guerreiro inimigo. Se isso não for possível, escreva -1.

Exemplos

Entrada	Saída
8 3 escaramucador 2 arqueiro cavalo cavalo 1 arqueiro capivara 1 grameiro arqueiro 2 atirador arqueiro pi1ueiro 1 cavalo camelo 1 atirador grameiro 0 atirador 0 arqueiro cavalo atirador	2

Problema 3. Food, please

Arquivo-fonte: `food.c`, `food.cpp` ou `food.java`

Um dos principais desafios do jogo Age Of Empires é conseguir comida para criar mais aldeões e soldados. Uma das melhores civilizações do jogo, a Lucius, é tão avançada que possui uma técnica única de conseguir alimentos.

No centro da cidade (uma construção que funciona como "prefeitura" e "maternidade" no jogo) há uma espécie de arma que é utilizada para lançar flechas em capivaras. Após uma capivara ser abatida, os aldeões retiram a carne dela, gerando comida para o jogo.

Porém, tal arma tem uma restrição muito inconveniente: as capivaras mudam constantemente de lugar e mover a arma para apontar para uma determinada direção é um processo lento.

Felizmente, se o jogador fizer uma pesquisa na Universidade do jogo (uma construção onde se pode pesquisar tecnologias para melhorar determinadas tarefas), é possível prever o local onde cada capivara estará e o melhor momento para abatê-la.

Sua tarefa consiste em determinar a quantidade máxima de capivaras que a civilização Lucius consegue abater com esta arma dadas as posições de cada capivara e o momento em que elas poderão ser abatidas.

Por simplicidade, assuma:

- Há N capivaras no jogo e cada uma só poderá ser abatida em um momento específico (apenas no momento em que ela parar descansar e, assim, virar um alvo fácil).
- A i-ésima ($1 \leq i \leq N$) capivara do jogo só poderá ser abatida no minuto i.
- O sistema de mira da arma foi mapeado para coordenadas 1D. A i-ésima capivara estará (no tempo i) na coordenada x_i (onde x_i é um inteiro). Inicialmente (no minuto 0) a arma aponta para a coordenada 0.
- A última capivara sempre é muito gorda e, dessa forma, você sempre deverá abatê-la (se não for possível abatê-la a resposta do seu programa deverá ser 0).
- Infelizmente, a arma consegue se mover em no máximo 1 unidade da coordenada por minuto e, com isso, nem sempre é possível atirar em todas capivaras.

A entrada consiste nas posições das capivaras no momento que podem ser abatidas.

Entrada

A entrada começa com um número N ($0 \leq N \leq 4,000$), que indica a quantidade de capivaras no jogo. A seguir há N inteiros x_i ($-1,000,000 \leq x_i \leq 1,000,000$) indicando a posição de cada capivara.

Saída

Escreva uma linha contendo um inteiro, indicando a quantidade máxima de capivaras que podem ser abatidas.

Exemplos

Entrada	Saída
9 1 -4 -1 4 5 -4 6 7 -2	4

Entrada	Saída
5 1 2 3 4 5	5

Explicação:

No primeiro exemplo, temos a seguinte configuração:

Capivara	1	2	3	4	5	6	7	8	9
Posição	1	-4	-1	4	5	-4	6	7	-2

Na melhor solução, a arma:

- começa da posição 0 no tempo 0 (como sempre)
- move para a posição 1 (chegando no tempo 1) e mata a capivara 1.
- move para a posição -1 (chegando no tempo 3) e mata a capivara 3.
- move para a posição -4 (chegando no tempo 6) e mata a capivara 6.
- move para a posição -2 (chegando no tempo 8), espera 1 minuto sem mover a arma e mata a capivara 9 (a última) no tempo 9.

No segundo exemplo todas as capivaras podem ser abatidas: a arma começa na posição 0 no tempo 0 e se move para a posição i, chegando no tempo i, justamente no momento ideal de atirar na i-ésima capivara.

Problema 4. Wood, please

Arquivo-fonte: `wood.c`, `wood.cpp` ou `wood.java`

Conforme todos agora já sabem, Paulo Lucio é um dos melhores jogadores de Age of Empires do universo. Uma das técnicas que ele utiliza para sempre vencer é utilizar um programa de computador para calcular o melhor exército que ele pode comprar.

Os inimigos do Paulo contrataram a Empresa Júnior "Sem Insetos" para desenvolver um programa similar ao dele (obviamente não conseguirão vencer o Paulo, mas os inimigos pelo menos conseguirão sobreviver por mais tempo). Sua tarefa, como membro da empresa, é implementar tal programa.

Você deverá ler uma lista descrevendo o custo de cada unidade (medido em quantidade de comida, madeira e ouro). Além disso, cada tipo de uma unidade possui um poder de ataque.

A seguir, você receberá uma lista descrevendo a quantidade de comida, madeira e ouro que o jogador possui. Seu programa devera escrever o maior poder de ataque total do exército (soma do poder de ataque de cada unidade) que poderá ser comprado com essa quantidade de recursos.

Entrada

A entrada começa com uma linha contendo um número N. A seguir, ha N ($0 \leq N \leq 50$) linhas descrevendo os tipos de unidades (tipos distintos) que podem ser criadas. Cada linha contem 4 inteiros $a\ c\ m\ o$, onde a é o poder de ataque da unidade ($0 \leq a \leq 1.000.000$), c é a quantidade de comida necessária para comprar 1 integrante dela, m é a quantidade de madeira e o é a quantidade de ouro ($0 \leq c, m, o \leq 60$).

Por fim, há uma linha contendo 3 inteiros CMO ($0 \leq C, M, O \leq 60$), indicando a quantidade de cada recurso (comida, madeira e ouro) que o jogador possui para gastar com o exército.

Saída

Escreva uma linha contendo um inteiro, indicando poder de ataque máximo do exército que pode ser obtido comprando-se unidades dos tipos informados com os recursos disponíveis).

Exemplos

Entrada	Saída
3 5 10 0 0 40 20 5 0 30 0 0 10 40 5 20	110

O melhor exército pode ser obtido comprando-se 2 integrantes do primeiro tipo (custo $10, 0, 0$ por integrante e poder 5), 1 do segundo tipo (custo $20, 5, 0$ e poder 40), e 2 do segundo tipo (custo $0, 0, 10$ e poder 30).

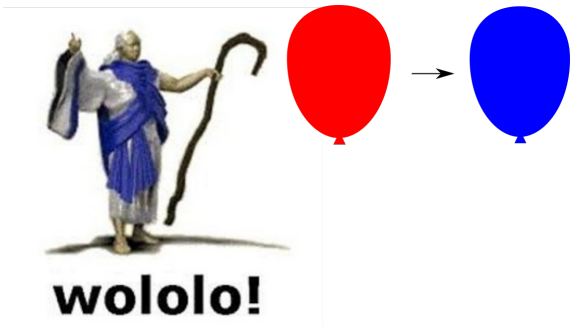
Problema 5. Gold, please

Arquivo-fonte: `gold.c`, `gold.cpp` ou `gold.java`

Um desafio que a organização da Semana de Informática teve este ano foi comprar pacotes de balões para a maratona. Como vocês devem saber, é necessário um balão com uma cor distinta para cada questão da competição.

Apos decidirem comprar pacotes com balões sortidos, os organizadores separaram os balões por cores e atribuíram uma cor para cada problema. Porem, uma das questões da prova ficou sem balão e, além disso, sobraram alguns balões com 3 cores muito raras.

Um dos organizadores teve a ideia de utilizar alguns monges do jogo Age of Empires para converter tais balões para uma única cor. Como os monges precisam recuperar a energia após cada conversão, decidiram converter o numero mínimo de balões para uma determinada cor de modo a terminarem essa tarefa rapidamente.



Entrada

A entrada contém 3 números inteiros indicando a quantidade disponível de balões de cada cor rara. Todos os números são inteiros positivos menores que 1.000.000.

Saída

Escreva o numero mínimo de balões que precisarão ser convertidos de modo que todos tenham a mesma cor.

Exemplos

Entrada	Saída
1 4 10	5

Entrada	Saída
3 1 1	2

Primeira entrada: a melhor estratégia consiste em converter todos os balões para a cor do terceiro balão (assim, precisaremos converter $1 + 4 = 5$ balões). Segunda entrada: o melhor é converter os balões para a primeira cor.

Problema 6. Capivarić

Arquivo-fonte: `ic.c`, `ic.cpp` ou `ic.java`

A Croácia chegou à final da Copa do Mundo FIFA 2018! Conseguiu a vaga ao derrotar a Inglaterra na prorrogação de uma emocionante partida das semi-finais. Antes havia vencido todos os jogos da fase de grupos (inclusive contra a Argentina!) e passado pela Dinamarca nos pênaltis (gols de Kramarić, Modrić e Rakitić) e pela Rússia também nos pênaltis (gols de Brozović, Modrić, Vida e Rakitić). Note que, com exceção do Vida, os que marcaram gols de pênalti têm o nome terminado com "ić" (pronuncia-se "itch"). Aliás, não é propriamente o nome ou sobrenome, mas um patronímico, e significa "filho de", ocorrendo em 2/3 dos jogadores da Croácia.

Outro caso interessante é o da Islândia, que disputou a copa pela primeira vez, tornando-se o menor país a disputar uma Copa da Mundo. Entre os jogadores estavam Sigurdsson (gol contra a Croácia), Finnbogason (gol de empate contra Argentina), Halldórsson, Gislason, Magnússon, etc. O sufixo "sson" (ou "son") também significa "filho de".

Capivarić ficou animado com essa descoberta e já se considera croata. Seu irmão Capivarsson se diz islandês, mas na verdade os dois, jogadores de futebol da Atlética das Ciências Exatas da UFV, são filhos de capivaras brasileiras mesmo.

Nesta questão você deve ler o nome de vários jogadores e identificar se eles são, provavelmente, da Croácia (se terminar com "ic"), da Islândia (se terminar com "sson" ou "son") ou de nenhuma das duas.

Observações

seria "ić", mas nesta questão use "ic" mesmo.

Entrada

A entrada começa com uma linha contendo um número N, indicando o número de jogadores a serem avaliados. Em seguida existem N linhas, cada uma com um nome de um jogador. Restrições: $1 \leq N \leq 30$; nomes dos jogadores com 5 a 80 caracteres, contendo apenas letras maiúsculas e minúsculas, espaço em branco ou hífen; apenas a primeira letra de cada palavra é maiúscula, então os patronímicos, se existirem, serão sempre letras minúsculas.

Saída

Seu programa deve produzir uma linha para cada jogador da entrada, com uma das palavras: "Croacia", "Islandia", ou "Nenhuma", dependendo do nome do jogador e de acordo com o enunciado.

Exemplos

Entrada	Saída
9 Luka Modric Alfred Finnbogason Lionel Messi Ivan Rakitic Gylfi Sigurdsson Cristiano Ronaldo dos Santos Aveiro Neymar da Silva Santos Junior Kylian Mbappe Lottin Alisson	Croacia Islandia Nenhuma Croacia Islandia Nenhuma Nenhuma Nenhuma Islandia

Problema 7. Capiflix

Arquivo-fonte: `capiflix.c`, `capiflix.cpp` ou `capiflix.java`

Capivarić costumava passar o dia (e algumas noites!) assistindo séries na Capiflix, a provedora de filmes e séries via *streaming* mais popular entre as capivaras. Mas, muita coisa mudou depois que ingressou em Ciência da Computação na UFV e começou a jogar futebol na Atlética: não tem mais tempo livre para assistir tudo o que quer! Uma coisa, entretanto, não mudou: quando começa a assistir uma série, não consegue parar, assiste episódio após episódio até terminar, usando todas as horas livres disponíveis na semana. Como agora não tem tanto tempo livre, precisa escolher com cuidado as séries que dá conta de assistir.

Capivarić só começa uma série se puder assistir todos os episódios de todas as temporadas no tempo livre que tem em uma semana. E não assiste séries de poucos episódios, pois não são interessantes. Com tantas séries disponíveis na Capiflix, e sem tempo a perder fazendo contas, Capivarić quer um programa que verifica se determinada série se encaixa ou não em suas restrições.

Entrada

A entrada contém apenas uma linha, com 5 números inteiros: T, o número de temporadas da série; E, o número de episódios de cada temporada (toda temporada tem o mesmo número de episódios); D, a duração em minutos de cada episódio (todo episódio tem a mesma duração); H, o número de horas livres na semana; e M, o mínimo de episódios de uma série interessante.

Restrições: $1 \leq T \leq 10$, $1 \leq E \leq 15$, $15 \leq D \leq 100$, $1 \leq H \leq 80$, $5 \leq M \leq 20$.

Saída

Seu programa deve produzir uma única linha na saída, contendo a palavra "SIM" ou a palavra "NAO", informando respectivamente se a série atende ou não as restrições de Capivarić.

Exemplos

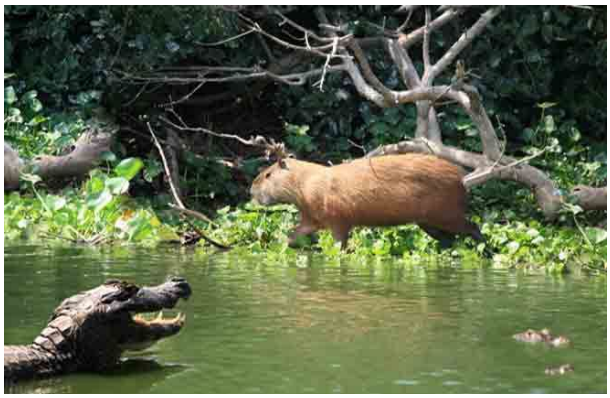
Entrada	Saída
`2 9 40 15 5`	`SIM`

Entrada	Saída
`2 9 40 15 20`	`NAO`

No primeiro exemplo, a série tem duração total de 720 minutos (18 episódios de 40 min), então pode ser assistida dentro das 15h livres da semana. No segundo, também pode ser assistida dentro do tempo livre, mas não é uma série interessante, pois não atende o requisito de ter no mínimo 20 episódios.

Problema 8. Capivara viúva na UFV-Florestal

Arquivo-fonte: `viuvaf.c`, `viuvaf.cpp` ou `viuvaf.java`



Conforme muitos sabem, há inúmeras capivaras nas lagoas da UFV. Recentemente, um grupo de pesquisadores decidiu fazer um cadastro de todos os casais de capivaras. Nesse cadastro, as capivaras que formam um casal receberam um identificador (um string com exatamente 10 caracteres), sendo que casais que são parentes sempre recebem o mesmo identificador.

Esta semana descobriram que uma das capivaras do campus de Florestal foi comida por um jacaré!

Dada a lista de identificadores das capivaras, determine o identificador da capivara que está faltando na UFV.

Como exemplo, considere que temos como entrada os seguinte 7 identificadores:

AAAAAAAAAA
AAAAAAAAAA
AAAAAAAAAB
AAAAAAAAAB
AAAAAAAAAA
AAAAAAAAAA
AAAAAAAAAB

Neste caso há 3 casais completos (2 com ids AAAAAAAAAA e 1 com id AAAAAAAAAB) e está faltando uma capivara do tipo AAAAAAAAAB (que formaria um quarto casal).

Observações

Sempre haverá exatamente uma capivara faltando.

Entrada

A entrada começa com uma linha contendo um inteiro N, indicando o número de identificadores das capivaras restantes na UFV. Em seguida existem N linhas, cada uma com um identificador de uma capivara (com exatamente 10 letras maiúsculas). Restrições: $1 \leq N \leq 100$.

Saída

Seu programa deve produzir uma linha com o identificador da capivara que está faltando.

Exemplos

Entrada	Saída
7 AAAAAAAAAA AAAAAAAAAA AAAAAAAAAB AAAAAAAAAB AAAAAAAAAA AAAAAAAAAA AAAAAAAAAB	AAAAAAAAAB

Problema 9. Capivara viúva na UFV-Viçosa

Arquivo-fonte: `viuvav.c`, `viuvav.cpp` ou `viuvav.java`

Ocorreu o mesmo problema relatado no problema anterior no campus de Viçosa (que tem muito mais capivaras do que em Florestal)!

Resolva o problema anterior de modo que ele consiga tratar uma quantidade maior de capivaras de forma eficiente.

Observações

Sempre haverá exatamente uma capivara faltando.

Entrada

A entrada começa com uma linha contendo um inteiro N, indicando o número de identificadores das capivaras restantes na UFV. Em seguida existem N linhas, cada uma com um identificador de uma capivara (com exatamente 10 letras maiúsculas). Restrições: $1 \leq N \leq 50.000.000$.

Saída

Seu programa deve produzir uma linha com o identificador da capivara que esta faltando.

Exemplos

Entrada	Saída
7 AAAAAAAAAA AAAAAAAAAA AAAAAAAAAB AAAAAAAAAB AAAAAAAAAA AAAAAAAAAA AAAAAAAAAB	AAAAAAAAAB

Problema 10. Busca Capivárica

Arquivo-fonte: `busca.c`, `busca.cpp` ou `busca.java`

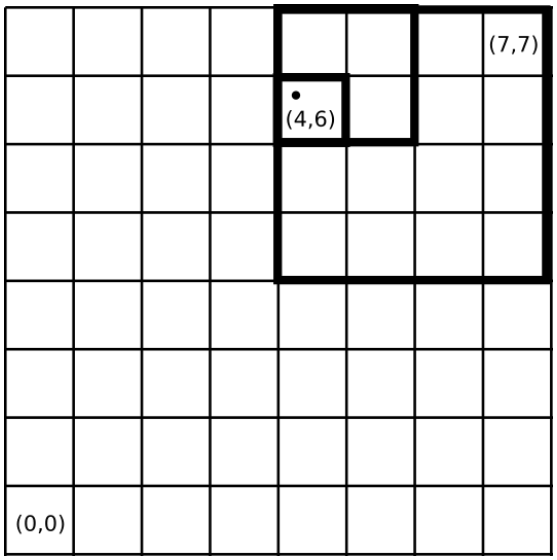
URGENTE: um bebê capivara acaba de se perder na UFV! Se algum de vocês o tiver visto, por favor, avise a vigilância da universidade imediatamente.

Infelizmente, até o momento, as buscas não tiveram exito. A boa notícia é que há uma testemunha que estava brincando com o bebê e sabe onde ele se encontra. No entanto, como esta testemunha é muito nova, ela não sabe como chegar até o bebê e nem falar onde ele se encontra.

Felizmente descobriram que as capivaras (mesmo as jovens) têm uma excelente capacidade de ler mapas e, assim, olhando para o mapa da UFV, a testemunha sabe exatamente onde o bebê se encontra. Apesar de não saber falar (e nem apontar), ela consegue dizer "quente" e "frio" (pois ela gosta muito de brincar de "tá quente"/"tá frio"). Após estudarem sobre busca binária, o pessoal da No Bugs teve a brilhante ideia de dividir o mapa da UFV (representado por uma grade quadrada) em quatro quadrantes e perguntar para a testemunha se o bebe estava em um deles. Esse processo é repetido (sempre dividindo o quadrante onde o bebê se encontra) até se ter certeza da célula do mapa contendo o local onde está o bebê.

A cada pergunta, a testemunha responde e palavras: $X\ Y\ Z\ W$ onde X, que pode ser "quente" ou "frio", indica se a capivara está no quadrante 1 (superior direito), Y é relativo ao quadrante 2 (superior esquerdo), Z é relativo ao quadrante 3 (inferior esquerdo) e W é relativo ao quadrante 4 (inferior esquerdo).

Considere o exemplo da figura abaixo. Neste caso, o mapa da UFV tem 8×8 células e a capivara está no ponto destacado na célula $(4, 6)$:



Na primeira pergunta, a capivara testemunha responderia "quente frio frio frio" (o bebê está no primeiro quadrante do mapa, abrangendo da coordenada $(4, 4)$ a $(7, 7)$). A seguir, ela responderia "frio quente frio frio" (o bebê está no segundo quadrante da nova região sendo considerada, abrangendo da coordenada $(4, 6)$ até $(5, 7)$). Finalmente, ela responderia "frio frio quente frio" (o bebê se encontra no terceiro quadrante, contendo apenas a célula $(4, 6)$). Com essa terceira resposta, já saberíamos que o bebê se encontra em algum lugar da célula $(4, 6)$.

Como um membro da No Bugs, sua tarefa é desenvolver um programa que, dada as respostas da testemunha, indica em qual célula o bebê capivara se encontra.

Observações

As coordenadas do mapa sempre crescem da esquerda para direita e de baixo para cima, sendo $(0, 0)$ as coordenadas da célula do canto inferior esquerdo, conforme exemplo acima.

Entrada

A entrada começa com uma linha contendo um inteiro N indicando as dimensões do mapa (sempre uma grade quadrada, $1 \leq N \leq 20.000.000$, com N potência de 2). Em seguida, haverá algumas linhas com as respostas da testemunha, sendo que cada linha contém 4 strings (cada uma pode ser "quente" ou "frio", sendo que há exatamente 3 strings "frio" por linha).

Saída

Seu programa deve escrever uma linha na saída contendo as coordenadas x e y (separadas por um espaço) da célula onde o bebê se encontra. Porém, se as respostas da testemunha não forem suficientes para determinar com exatidão qual é essa célula, escreva a frase "interrogue mais!".

Exemplos

Entrada	Saída
8 quente frio frio frio frio quente frio frio frio frio quente frio	4 6
8 quente frio frio frio	interrogue mais!

O primeiro exemplo é o exemplo dado no enunciado.

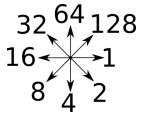
Problema 11. Falta d'agua

Arquivo-fonte: `agua.c`, `agua.cpp` ou `agua.java`

Como muitos sabem, frequentemente a falta de chuva reduz o nível das lagoas da UFV. Preocupadas com a situação da lagoa no segundo semestre de 2018, um grupo de capivaras decidiu pedir ao DPI - Departamento de Pesquisas Ídricas (sim, elas não são muito boas com português) para calcular a quantidade de água que as lagoas receberão caso chova. Como membro do DPI, ajude-as nessa tarefa.

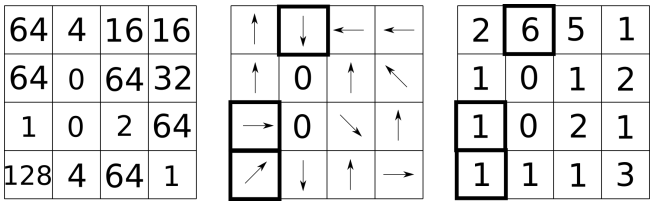
Inicialmente as capivaras criaram um mapa de direção de fluxo. Tal mapa representa a UFV e consiste em uma matriz quadrada (a área da universidade foi aproximada) onde cada célula contém um código representando a direção de escoamento da água (sso foi calculado utilizando um algoritmo desenvolvido pelas capivaras). A ideia é que, ao chover, a água que cai em cada célula escoará completamente para uma das 8 células vizinhas.

A figura abaixo indica o código de cada uma das 8 possíveis direções de fluxo (excluindo o código 0, que indica presença de lagoa). Por exemplo, se a direção de fluxo da célula do centro tiver código 64, então a água dela será direcionada para a célula de cima; e se o código for 1, a água será direcionada para a célula da direita.



Nessa simulacao supoem que cada celula recebera 1 litro de agua por minuto. Seu objetivo e' calcular quantos litros de agua as lagoas receberao por minuto (supondo que ja esteja chuvendo ha um tempo e, portanto, a quantidade de agua passando em cada celula ja esteja estabilizada).

Considere, por exemplo, as duas figuras abaixo, que supõem a UFV representada por uma matriz 4×4:



A figura da esquerda apresenta o código das direções de fluxo na matriz, a do centro ilustra as direcoes e a da direita contem o fluxo acumulado em cada celula (exceto na lagoa).

Note que a primeira célula do canto superior esquerdo receberá 2 litros de água por minuto (1 litro da chuva e 1 litro vindo da célula que está abaixo dela) e toda essa água escoará para fora do mapa. A lagoa recebera 10 litros de água, sendo 8 litros vindo das células destacadas e 2 litros vindos diretamente da chuva que cai na lagoa.

Observações

É garantido que, se uma determinada quantidade de água cair em uma célula, essa água eventualmente chegará a um destino final, isto é, ou escoará para fora do terreno ou para uma lagoa.

Entrada

A entrada começa com um inteiro N ($0 \leq N \leq 2,000$) indicando as dimensões da matriz quadrada que representa o mapa. A seguir N linhas, cada uma contendo N inteiros, representando o código das direções de fluxo no mapa.

Saída

Escreva uma linha contendo um inteiro, indicando a quantidade de água fluindo para as lagoas a cada minuto.

Exemplos

Entrada	Saída
4 64 4 16 16 64 0 64 32 1 0 2 64 128 4 64 1	10

Problema 12. PCdaU

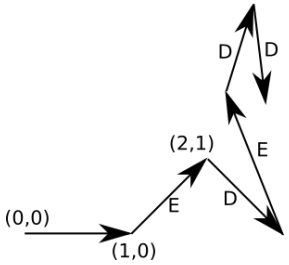
Arquivo-fonte: `pcdau.c`, `pcdau.cpp` ou `pcdau.java`

Em 2018, as capivaras da UFV decidiram criar um partido político para as eleições. Nasceu então o Partido das Capivaras da UFV (PCdaU).

Assim como nas redes sociais, há uma imensa briga entre as capivaras com visão mais à direita e as com visão mais à esquerda e, dessa forma, está muito difícil definir a orientação política do PCdaU. Como uma das principais visões do partido é a democracia, decidiu-se que isso será definido com base na visão política da maioria das capivaras da universidade. Dessa forma, será feita uma votação para escolher a orientação do partido.

Infelizmente as capivaras não sabem usar urnas eletrônicas. Felizmente, há uma forma bem simples de avaliar a visão política de uma capivara: basta analisar seu movimento! Se, ao andar, a capivara virar mais vezes à direita que à esquerda, sua orientação política é "direita". Se ela virar mais vezes à esquerda que à direita, ela é de "esquerda". Se ela vira a mesma quantidade de vezes em cada direção, ela é de "centro".

Os políticos do PCdaU pediram a você para criar um programa que, dada a rota de uma capivara, determina se ela é de esquerda, de direita ou de centro.



Considere, por exemplo, a rota ao lado (exibimos apenas as coordenadas dos 3 primeiros pontos). Note que a capivara começa no ponto $(0, 0)$, se move até o ponto $(1, 0)$, faz uma rotação à esquerda e, após chegar em $(2, 1)$, faz uma rotação à direita. No total, são feitas 3 rotações à direita e 2 à esquerda. Então, a capivara que fez esta rota é de "direita".

Entrada

A entrada começa com uma linha contendo um inteiro N ($2 \leq N \leq 10.000.000$), indicando o número de coordenadas da rota percorrida pela capivara sendo analisada. A seguir há N pares de inteiros X Y, indicando a coordenada (X, Y) de cada ponto da rota ($-10.000 \leq X \leq 10.000$, $-10.000 \leq Y \leq 10.000$).

Saída

Seu programa deve produzir uma linha contendo a palavra "`direita`" se a capivara for de direita, "`esquerda`" se a capivara for de esquerda, ou "`centro`" se a capivara for de centro.

Exemplos

Entrada	Saída
7 0 0 1 0 2 1 3 0 2 2 3 3 3 2	direita

Entrada	Saída
6 0 0 1 0 2 1 3 0 2 2 3 3	centro