6º Hackathon Fulltime
FATEC Garça

(26 de agosto de 2023)

Caderno de Problemas

Informações Gerais

- Este caderno contém 10 problemas – A a J, com páginas numeradas de 01 a 13. Verifique se o caderno está completo.

- A competição possui duração de 4 horas (início as 13:00h término as 17:00h);

- Durante a competição é permitida aos competidores a consulta a material próprio impresso em papel (livros, apostilas, anotações);

- NÃO é permitido acesso a conteúdo da Internet e nem conteúdos disponíveis em qualquer meio magnético;

- É vedada a comunicação entre os competidores durante a competição, bem como a troca de material de consulta entre eles;

- Cada competidor terá acesso a 1 computador dotado do ambiente de submissão de programas beecrowd, dos compiladores, link-editores e IDEs requeridos pelas linguagens de programação permitidas;

- Não é permitido o uso de notebooks ou outro tipo de computador ou assistente pessoal;

- Os problemas têm o mesmo valor na correção;

- A correção é automatizada, portanto siga atentamente as exigências do problema quanto ao formato da entrada e saída de seu programa;

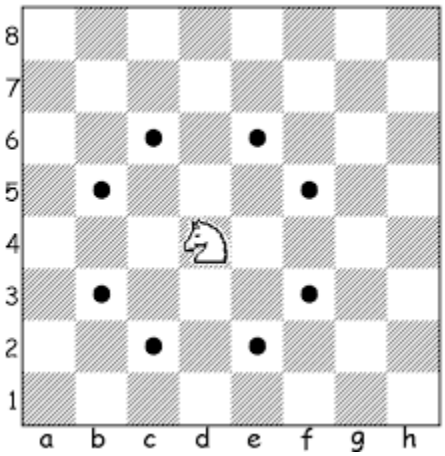
- Não implemente nenhum recurso gráfico nas suas soluções (janelas, menus, etc.), todos os dados devem ser lidos da entrada padrão e escritos na saída padrão. Nem utilize qualquer rotina para limpar a tela ou posicionar o cursor;

- Preste muita atenção na linguagem escolhida;

- As soluções serão testadas com outras entradas além das apresentadas como exemplo nos problemas;

Problema A
Mais cavalos

Dado a posição inicial de um cavalo em um tabuleiro de xadrez e a posição destino, deve se dizer se, com exatamente um único movimento, o cavalo consegue alcançar a posição destino. Se isso for possível, o movimento é classificado como válido, caso contrário, o movimento é dito inválido.



Em um tabuleiro de xadrez se utiliza números, de 1 a 8, para especificar a linha do tabuleiro e letras, de 'a' a 'h' para especificar a coluna.

Entrada

A entrada é composta por uma única linha contendo a posição inicial do cavalo e a posição destino, separadas por um espaço em branco. Uma posição no tabuleiro é especificada por um caractere, que representa a coluna, seguido de um número inteiro que representa a linha.

Saída

A saída consiste em uma linha contendo a mensagem "VALIDO" caso o movimento seja um movimento válido de um cavalo no jogo de xadrez ou "INVALIDO" caso contrário.

Exemplos

Entrada	**Saída**
d4 b5	VALIDO

Entrada	**Saída**
a1 g6	INVALIDO

Entrada	**Saída**
h8 f7	VALIDO

Problema B
Fatorial de Novo!

Mateus, um calouro de engenharia, está desenvolvendo uma nova notação posicional para representar números inteiros. Ele o apelidou de "A Curious Method" ("Um Método Curioso"), representado pela sigla ACM. A notação ACM usa os mesmos dígitos que a notação decimal, isto é, de 0 a 9.

Para converter um número *A* da notação ACM para a notação decimal, você deve adicionar *k* termos, onde *k* é o número de dígitos de *A* (na notação ACM), O valor do *i*-ésimo termo, correspondente ao *i*-ésimo dígito *ai*, contando da direita para a esquerda, é *ai* × *i*!. Por exemplo, 719_{ACM} é equivalente a 53_{10}, já que 7 × 3! + 1 × 2! + 9 × 1! = 53.

Mateus acabou de iniciar seus estudos sobre teoria dos números, e provavelmente não sabe quais propriedades um sistema numérico deve ter, mas no momento, ele só está interessado em converter um número de ACM para decimal. Você pode ajudá-lo?

Entrada

Cada caso de teste é dado por uma única linha não-nula contendo, no máximo, 5 dígitos, representando um número na notação ACM. A linha não possui zeros no início.

O último caso de teste é representado por uma linha contendo um único zero.

Saída

Para cada caso de teste, escreva uma única linha contendo a representação decimal do número ACM correspondente.

Exemplos

Entrada	**Saída**
719 1 15 110 102 0	53 1 7 8 8

Problema C
Horos e os copos

Astrologia é uma pseudociência segundo a qual as posições relativas dos corpos celestes poderiam prover informação sobre a personalidade, as relações humanas, e outros assuntos relacionados à vida do ser humano. Horo não acredita em astrologia, mas ele ganhou uma coleção de copos de aniversário, e cada copo estava estampado com um dos signos do zodíaco. Em um momento de curiosidade, resolveu pesquisar quais seriam os signos de seus amigos. Considerando a tabela abaixo com as datas de cada signo, ajude Horo a descobrir o signo de seus amigos.

♈	Áries 21/03 a 20/04	♎	Libra 23/09 a 22/10
♉	Touro 21/04 a 20/05	♏	Escorpião 23/10 a 21/11
♊	Gêmeos 21/05 a 20/06	♐	Sagitário 22/11 a 21/12
♋	Câncer 21/06 a 22/07	♑	Capricórnio 22/12 a 19/01
♌	Leão 23/07 a 22/08	♒	Aquário 20/01 a 18/02
♍	Virgem 23/08 a 22/09	♓	Peixes 19/02 a 20/03

A entrada contém a data de aniversário de um amigo de Horo (data no formato dd/mm).

Saída

Para cada data de aniversário, informe o signo do amigo de Horo, com letras minúsculas e sem acento.

Exemplos

Entrada	**Saída**
26/02	peixes

Entrada	**Saída**
16/10	libra

Entrada	**Saída**
27/05	gêmeos

Problema D
Guing Pictures

Enzo recently traveled to the city of Montevideo, where he saw a big sign with the name of the city. He decided to take pictures of the sign to make a collage and send it to his friend Demonio. Enzo wants to form the name of his friend by taking one or several pictures of sections of the sign. For example, with the string **"MONTEVIDEO"**, he might form the name of his friend by putting together **"DE-MON-I-O"**, using four pictures to form the entire name. It is easy to show that the result cannot be achieved with fewer pictures.

You will be given the name of a city and a list of friends' names. Return the minimum number of pictures needed to form the name of each friend. When forming the names, pictures cannot be rotated, reflected or modified in any way.

Input

The first line contains a string **C** indicating the name of the city. The second line contains a positive integer **N** representing the number of friends. Each of the following **N** lines contains a string indicating the name of a friend. All strings are non-empty and consist only of uppercase letters. The sum of the lengths of all strings is at most $\mathbf{2 \times 10^5}$.

Output

Output **N** lines, each line with an integer indicating the minimum number of needed pictures to form the corresponding name in the input, or the value **"-1"** if it is not possible to generate the name.

Samples

Input	**Output**
MONTEVIDEO 4 DEMONIO MONTE EDIT WON	4 1 4 -1

Input	**Output**
SANTIAGO 3 TITA SANTIAGO NAS	3 1 3

Problema E
Cartão

Mike frequentemente precisa saber se ele poderia colocar um cartão retangular de tamanho **A** x **B** dentro de um envelope de tamanho **C** x **D**. Para ser mais rápido, Mike realmente não tenta colocar um cartão em um envelope, ele apenas coloca um cartão na mesa e tenta cobri-lo com um envelope. Naturalmente, tanto o cartão quanto o envelope podem ser girados, mas não podem ser dobrados.

Agora, Mike quer ser ainda mais rápido. Ele decidiu encontrar as respostas para todos os tamanhos de cartões e envelopes com os quais ele opera. É aí que você entra. Seu programa deve calcular a resposta para um caso específico. O programa deve funcionar da mesma maneira que o Mike faz seus testes, então em casos de limite a resposta é “yes”.

Entrada

A primeira linha contém quatro inteiros **A**, **B**, **C** e **D**, delimitados por um espaço. Todos os valores são menores que $2x10^9$.

Saída

A saída contém somente uma string: “yes” or “no” (sem aspas).

Exemplos

Entrada	**Saída**
2 3 3 4	yes

Problema F
Qual é a Altura?

Nick é um cientista que viaja por diversos universos paralelos, juntamente com o seu neto, Mory. Em um desses universos, havia um programa de televisão, que premiava quem adivinhasse as alturas máximas de arremessos de frutas. Neste local, a massa da fruta não influenciava na altura máxima do arremesso. Nick calculava o ângulo do arremesso, que formava sempre uma parábola, e extraía uma função de segundo grau da trajetória. Ajude Nick e Mory a ganhar muitos prêmios neste programa.

Entrada

A entrada é composta por vários casos de teste. A primeira linha contém um número inteiro **T** (2 <= **T** <= 99) relativo ao número de casos de teste. As **T** linhas seguintes possuem três valores inteiros **A** (**A** < 0), **B** e **C** (-100 <= **B, C** <= 100), representando os coeficientes de uma função de segundo grau, na forma $ax^2 + bx + c$.

Saída

Para cada caso de teste de entrada do seu programa, você deve imprimir um número real, com aproximação de duas casas decimais, a altura máxima do arremesso de uma fruta.

Fórmulas úteis:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \qquad Xv = \frac{-b}{2a} \qquad Yv = \frac{-\Delta}{4a}$$

Exemplos

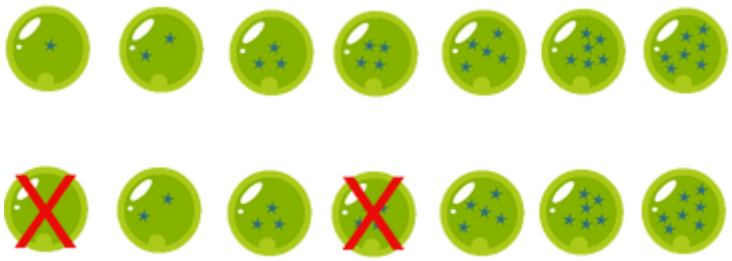
Entrada	Saída
3 -1 4 1 -1 3 0 -1 -1 3	5.00 2.25 3.25

Problema G
Xenlonghinho

Kogu está buscando as esferas do dragão para invocar Xenlonguinho e pedir para ele reviver seu amigo Kuriri, que infelizmente morreu na última batalha dos guerreiros Zê.

Porém Kogu está tendo muita dificuldade para encontrar as esferas, por isso Xenlonguinho que é seu conhecido há muito tempo, decidiu abrir uma exceção e aceitou ser invocado caso Kogu encontre todas as esferas cujo número de divisores da quantidade de estrelas da esfera sejam par.

Por exemplo se existem sete esferas, Kogu não precisaria encontrar as esferas de uma e quatro estrelas, pois elas têm uma quantidade ímpar de divisores, então ele só precisa pegar 5 esferas para invocar Xenlonguinho.



Como Kogu não é muito bom em contas, ele pediu para você escrever um programa que dado o total de esferas existentes, mostre a quantidade mínima de esferas que ele precisa procurar.

Entrada

A primeira linha consiste de um inteiro **C** que representa a quantidade de casos de teste. As linhas subsequentes contêm um inteiro **N** ($2 \leq N \leq 1000$) que representa a quantidade de esferas necessárias para invocar Xenlonguinho.

Saída

Seu programa deve exibir a quantidade mínima de esferas que Kogu tem que procurar.

Exemplos

Entrada	**Saída**
1 7	5

Problema H
Enigma

Dada uma configuração inicial, a máquina de criptografia alemã Enigma, da Segunda Guerra Mundial, substituía cada letra digitada no teclado por alguma outra letra. A substituição era bastante complexa, mas a máquina tinha uma vulnerabilidade: uma letra nunca seria substituída por ela mesma! Essa vulnerabilidade foi explorada por Alan Turing, que trabalhou na criptoanálise da Enigma durante a guerra. O objetivo era encontrar a configuração inicial da máquina usando a suposição de que a mensagem continha uma certa expressão usual da comunicação, como por exemplo a palavra **ARMADA**. Essas expressões eram chamadas de *cribs*. Se a mensagem cifrada era, por exemplo, **FDMLCRDMRALF**, o trabalho de testar as possíveis configurações da máquina era simplificado porque a palavra **ARMADA**, se estivesse nessa mensagem cifrada, só poderia estar em duas posições, ilustradas na tabela abaixo com uma seta. As demais cinco posições não poderiam corresponder ao *crib* **ARMADA** porque ao menos uma letra do *crib*, sublinhada na tabela abaixo, casa com sua correspondente na mensagem cifrada; como a Enigma nunca substituiria uma letra por ela própria, essas cinco posições poderiam ser descartadas nos testes.

F	D	M	L	C	R	D	M	R	A	L	F
A	R	M	A	D	A						
	A	R	M	A	D	A	←				
		A	R	M	A	D	A				
			A	R	M	A	D	A	←		
				A	R	M	A	D	A		
					A	R	M	A	D	A	
						A	R	M	A	D	A

Neste problema, dada uma mensagem cifrada e um *crib*, seu programa deve computar o número de posições possíveis para o *crib* na mensagem cifrada.

Entrada

A primeira linha da entrada contém a mensagem cifrada, que é uma sequência de pelo menos uma letra e no máximo 10^4 letras. A segunda linha da entrada contém o *crib*, que é uma sequência de pelo menos uma letra e no máximo o mesmo número de letras da mensagem. Apenas as 26 letras maiúsculas, sem acentuação, aparecem na mensagem e no *crib*.

Saída

Imprima uma linha contendo um inteiro, indicando o número de posições possíveis para o *crib* na mensagem cifrada.

Exemplos

Entrada	**Saída**
FDMLCRDMRALF ARMADA	2

Entrada	**Saída**
AAAAABABABABABABABA ABA	7

Problema I
Camisetas

O professor Rolien organizou junto às suas turmas de Ciência da Computação a confecção de uma camiseta polo que fosse ao mesmo tempo bonita e barata. Após algumas conversas, ficou decidido com os alunos que seriam feitas somente camisetas da cor preta, o que facilitaria a confecção. Os alunos poderiam escolher entre o logo do curso e os detalhes em branco ou vermelho. Assim sendo, Rolien precisa de sua ajuda para organizar as listas de quem quer a camiseta em cada uma das turmas, relacionando estas camisetas pela cor do logo do curso, tamanho (P, M ou G) e por último pelo nome.

Entrada

A entrada contém vários casos de teste. Cada caso de teste inicia com um valor **N**, (1 ≤ **N** ≤ 60) inteiro e positivo, que indica a quantidade de camisetas a serem feitas para aquela turma. As próximas **N***2 linhas contêm informações de cada uma das camisetas (serão duas linhas de informação para cada camiseta). A primeira linha irá conter o nome do estudante e a segunda linha irá conter a cor do logo da camiseta ("branco" ou "vermelho") seguido por um espaço e pelo tamanho da camiseta "P" "M" ou "G". A entrada termina quando o valor de **N** for igual a zero (0) e este valor não deverá ser processado.

Saída

Para cada caso de entrada deverão ser impressas as informações ordenadas pela cor dos detalhes em ordem ascendente, seguido pelos tamanhos em ordem descendente e por último por ordem ascendente de nome, conforme o exemplo abaixo.

Obs.: Deverá ser impressa uma linha em branco entre dois casos de teste.

Exemplos

Entrada	**Saída**
9 Maria Jose branco P Mangojata Mancuda vermelho P Cezar Torres Mo branco P Baka Lhau vermelho P JuJu Mentina branco M Amaro Dinha vermelho P Adabi Finho branco G	branco P Cezar Torres Mo branco P Maria Jose branco M JuJu Mentina branco G Adabi Finho branco G Severina Rigudinha vermelho P Amaro Dinha vermelho P Baka Lhau vermelho P Carlos Chade Losna vermelho P Mangojata Mancuda branco P Maria Joao branco P Maria Jose vermelho P Marcio Guess

Severina Rigudinha branco G Carlos Chade Losna vermelho P 3 Maria Joao branco P Marcio Guess vermelho P Maria Jose branco P 0	

Problema J
O jogo dos copos

Uma brincadeira muito comum e divertida entre dois jogadores usa uma moeda e três copos opacos (ou seja, não é possível ver o que está dentro do copo olhando pela lateral do copo). Os três copos são colocados com a boca para baixo, em uma linha, um ao lado do outro, em posições que vamos chamar de **A**, **B** e **C**. Uma moeda é colocada embaixo de um dos copos. Na brincadeira, um jogador chamado banca realiza um movimento para trocar a posição de dois copos, arrastando os copos de tal modo que se a moeda está embaixo de um dos copos envolvidos no movimento, ela continua embaixo do mesmo copo após a troca de posição. O jogador banca pode realizar três tipos de movimento, ilustrados na figura abaixo:

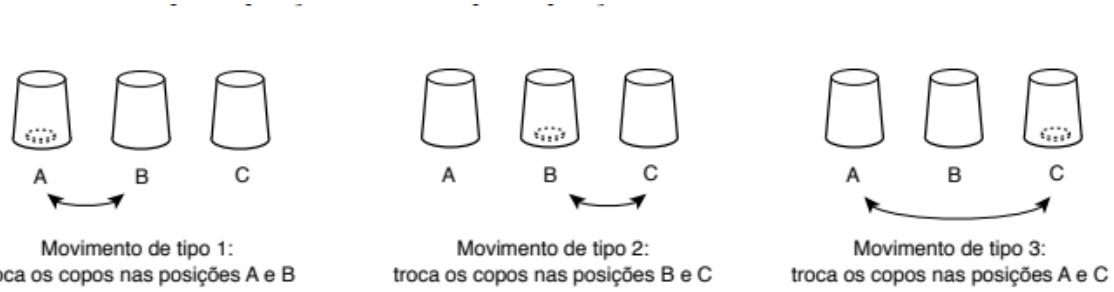
1. Trocar o copo na posição **A** com o copo na posição **B**.

2. Trocar o copo na posição **B** com o copo na posição **C**.

3. Trocar o copo na posição **A** com o copo na posição **C**.

O jogador banca realiza vários movimentos de troca tentando confundir o outro jogador, chamado espectador. Ao final o jogador espectador deve dizer em qual posição está a moeda. Por exemplo, considere que inicialmente a moeda está embaixo do copo na posição A e que o jogador banca realiza uma sequência de apenas três trocas, executando um movimento do tipo **1**, após o qual moeda termina embaixo do copo na posição **B**, seguido de um movimento do tipo **2**, após o qual a moeda termina embaixo do copo na posição **C**, seguido de um movimento do tipo **3**, após o qual a moeda termina embaixo do copo na posição **A**.

Nesta tarefa, dadas a descrição da sequência de movimentos e a posição inicial da moeda, você deve escrever um programa que determine a posição final da moeda após todos os movimentos.



Entrada

A primeira linha contém um inteiro **N**, o número de movimentos que o jogador banca realiza. **A** segunda linha contém um caractere, entre **A**, **B** e **C**, indicando a posição inicial da moeda. Cada uma das **N** linhas seguintes contém um inteiro, indicando o tipo de movimento efetuado pelo jogador banca na sequência.

Saída

Seu programa deve produzir uma única linha, com um único caractere entre **A**, **B** e **C**, a posição em que a moeda se encontra ao final da sequência de movimentos.

Restrições • 1 ≤ N ≤ 1000

Exemplos

Entrada	Saída
3 A 1 2 3	A

Entrada	Saída
6 C 1 2 3 3 1 1	B

Entrada	Saída
1 B 3	B