

A. Walking in Time

Time limit: 1s

Imagine que você tenha uma máquina do tempo que pode ser usada no máximo três vezes, e a cada uso da máquina você pode escolher voltar para o passado ou ir para o futuro. A máquina possui três créditos fixos; cada crédito representa uma certa quantidade de anos, e pode ser usado para ir essa quantidade de anos para o passado ou para o futuro. Você pode fazer uma, duas ou três viagens, e cada um desses três créditos pode ser usado uma vez apenas. Por exemplo, se os créditos forem 5, 12 e 9, você poderia decidir fazer duas viagens: ir 5 anos para o futuro e, depois, voltar 9 anos para o passado. Dessa forma, você terminaria quatro anos no passado, em 2012. Também poderia fazer três viagens, todas indo para o futuro, usando os créditos em qualquer ordem, terminando em 2042.

Neste problema, dados os valores dos três créditos da máquina, seu programa deve dizer se é ou não possível viajar no tempo e voltar para o presente, fazendo pelo menos uma viagem e, no máximo, três viagens; sempre usando cada um dos três créditos no máximo uma vez.

Entrada


A entrada consiste de uma linha contendo os valores dos três créditos **A**, **B** e **C** ($1 \leq A, B, C \leq 1000$).

Saída

Seu programa deve imprimir uma linha contendo o caracter “S” se é possível viajar e voltar para o presente, ou “N” caso contrário.

Exemplos de Entrada	Exemplos de Saída
22 5 22	S
31 110 79	S
45 8 7	N

Maratona de Programação SBC 2016

Por Maratona de Programação SBC  Brazil

B. Assembling

Time limit: 1s

Estamos no ano de 2931. Cientistas detectaram um meteoro que, em 15 meses, irá colidir com a Terra e extinguir a vida no planeta. Não há mais tempo para preparar ofensivas contra o meteoro, então só nos resta realizar nossos últimos desejos e esperar a colisão.

Um grupo de pessoas resolve se unir e realizar o último sonho de centenas de milhares de pessoas: ver a Portuguesa campeã da Copa Libertadores da América. Para tal, será necessário a contratação de jogadores de grande habilidade, que também são muito caros.

Para conseguir fazer isso, eles estudaram a personalidade dos melhores jogadores do mundo, e chegaram à conclusão que alguns aceitariam jogar na Portuguesa mais facilmente (isto é, seria contratado por um preço menor) se percebessem que seriam as únicas "estrelas" do time. Já outros, viriam mais facilmente caso percebessem que na Portuguesa já existem outras estrelas.

Assim, através de um estudo mais detalhado das personalidades, conseguiram definir, para cada jogador, qual seria o preço para contratá-los em vários cenários.

Por exemplo, o jogador X poderia ser contratado por 3 se fosse a única estrela do time ou por 5 se já houvesse 1 estrela no time antes dele entrar. Já o jogador Y seria contratado por 4 se fosse a única estrela do time, ou 2 se já houvesse uma estrela no time.

Nesse cenário, a melhor maneira de contratar X e Y seria contratar primeiro o jogador X por 3 e depois Y por 2, gastando \$5 no total.

Você receberá os dados dos custos de contratação dos jogadores em cada cenário, e deverá dizer quanto os torcedores da Lusa deverão economizar para montar o time dos sonhos e conquistar a tão sonhada Libertadores.

Entrada

A entrada contém várias instâncias.

A primeira linha contém um número inteiro N ($2 \leq N \leq 18$), representando a quantidade de jogadores a serem contratados. Cada uma das próximas N linhas representa um jogador. Cada uma possui N inteiros $c_0, c_1, c_2, \dots, c_{N-1}$ ($1 \leq c_i \leq 1000$, para todo $0 \leq i < N$) separados por espaços, onde c_k representa o custo para se contratar o jogador c se já tiverem sido contratados k jogadores.

A entrada termina quando $N = 0$.

Saída


Para cada instância na entrada, imprima uma linha com um inteiro representando a quantidade mínima de dinheiro que deverá ser gasto para a contratação dos N jogadores.

Exemplo de Entrada

Exemplo de Saída

Exemplo de Entrada	Exemplo de Saída
3 4 2 4 2 2 3 3 1 5 2 1 2 2 2 0	7 3

XIII Maratona de Programação IME-USP, 2009

Por XIII Maratona de Programação IME-USP, 2009  Brazil

C. Which floor?

Time limit: 1s

Memory limit: 256 MB

In a building where Polycarp lives there are *equal* number of flats on each floor. Unfortunately, Polycarp don't remember how many flats are on each floor, but he remembers that the flats are numbered from 1 from lower to upper floors. That is, the first several flats are on the first floor, the next several flats are on the second and so on. Polycarp don't remember the total number of flats in the building, so you can consider the building to be infinitely high (i.e. there are infinitely many floors). Note that the floors are numbered from 1.

Polycarp remembers on which floors several flats are located. It is guaranteed that this information is not self-contradictory. It means that there exists a building with equal number of flats on each floor so that the flats from Polycarp's memory have the floors Polycarp remembers.

Given this information, is it possible to restore the exact floor for flat n ?

Input

The first line contains two integers n and m ($1 \leq n \leq 100$, $0 \leq m \leq 100$), where n is the number of the flat you need to restore floor for, and m is the number of flats in Polycarp's memory.

m lines follow, describing the Polycarp's memory: each of these lines contains a pair of integers k_i, f_i ($1 \leq k_i \leq 100$, $1 \leq f_i \leq 100$), which means that the flat k_i is on the f_i -th floor. All values k_i are distinct.

It is guaranteed that the given information is not self-contradictory.

Output

Print the number of the floor in which the n -th flat is located, if it is possible to determine it in a unique way. Print -1 if it is not possible to uniquely restore this floor.

Examples

input
10 3 6 2 2 1 7 3
output
4

input
8 4 3 1 6 2 5 2 2 1
output
-1

Note

In the first example the 6-th flat is on the 2-nd floor, while the 7-th flat is on the 3-rd, so, the 6-th flat is the last on its floor and there are 3 flats on each floor. Thus, the 10-th flat is on the 4-th floor.

In the second example there can be 3 or 4 flats on each floor, so we can't restore the floor for the 8-th flat.

D. Leader's Impeachment

Time limit: 1s

Analógimôn Go! é um jogo bastante popular. Os jogadores de *Analógimôn Go!* são divididos em três grandes times: Time Valor, Time Instinto e Time Místico, que são liderados pelos seus líderes Kandera, Esparky e Blanque, respectivamente. Naturalmente, você faz parte de um desses times!

O líder do seu time está sendo acusado de infringir as regras do jogo por gerenciar incorretamente os doces recebidos do Professor que são destinados ao time. Isto criou uma grande polêmica dentro da equipe: alguns jogadores defendem que o líder realmente agiu incorretamente e deve sofrer um *impeachment* e ser afastado de seu cargo, enquanto outros defendem que ele não infringiu as regras, que a acusação é inverídica e que ele deve continuar no cargo.

Para resolver a situação, uma votação será realizada entre todos os N jogadores do seu time. Cada jogador deverá votar se o *impeachment* deve ou não ocorrer. Se o número de votos favoráveis ao *impeachment* foi maior ou igual a $2/3$ (dois terços) do total de jogadores, o líder será afastado. Caso contrário, a acusação é arquivada e ele continuará no cargo.

Dados os votos de todos os jogadores, determine o resultado da votação.

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso contém o inteiro N ($1 \leq N \leq 10^5$), o número de jogadores em seu time. A próxima linha contém N inteiros v_1, \dots, v_N ($v_i = 0$ ou 1), indicando os votos dos jogadores. O valor 1 indica um voto favorável ao *impeachment*, enquanto o valor 0 indica um voto contrário ao mesmo.

A entrada termina com fim-de-arquivo (EOF).

Saída

Para cada caso de teste, imprima uma linha contendo a palavra *impeachment* se o líder deve ser afastado de seu cargo, ou *acusacao arquivada* caso contrário.

Exemplo de Entrada	Exemplo de Saída
6 1 0 1 1 0 1 5 0 1 1 1 0	impeachment acusacao arquivada

Programação Competitiva, UFPR

Por Ricardo Oliveira, UFPR  Brazil

E. Count on a tree II

Time limit: 1.207s

Memory limit: 1536 MB

You are given a tree with **N** nodes. The tree nodes are numbered from **1** to **N**. Each node has an integer weight.

We will ask you to perform the following operation:

- **u v** : ask for how many different integers that represent the weight of nodes there are on the path from **u** to **v**.

Input

In the first line there are two integers **N** and **M**. (**N** ≤ 40000, **M** ≤ 100000)

In the second line there are **N** integers. The *i*-th integer denotes the weight of the *i*-th node.

In the next **N-1** lines, each line contains two integers **u v**, which describes an edge (**u**, **v**).

In the next **M** lines, each line contains two integers **u v**, which means an operation asking for how many different integers that represent the weight of nodes there are on the path from **u** to **v**.

Output

For each operation, print its result.

Example**Input:**

```
8 2
105 2 9 3 8 5 7 7
1 2
1 3
1 4
3 5
3 6
3 7
4 8
2 5
7 8
```

Output:

```
4
4
```

Just for fun...

F. Corrida

Time limit: 1s

Todo ano, os habitantes da Mlogônia, apesar das crises internas, reúnem-se em torno de um esporte que é a paixão nacional: as corridas de carros. A Grande Corrida anual é um enorme evento organizado pela Associação de Corridas da Mlogônia (ACM), sendo amplamente televisionado e reportado em jornais e revistas de todo o país. Os resultados da corrida são tema principal das rodas de conversa por semanas.

Por bastante tempo, os resultados da Grande Corrida eram compilados manualmente. Observadores especializados iam à pista medir o tempo de cada um dos N carros, numerados de 1 a N , em cada uma das M voltas, anotando então os resultados em tabelas para posterior análise por parte das equipes e dos jornalistas. Muitos erros eram introduzidos nesse processo, e a organização decidiu informatizar todo o sistema.

A ACM percebeu que o esforço necessário para a construção do sistema seria grande, e optou por contar com a ajuda de uma equipe de programadores. Percival foi contratado para escrever a parte do software que determina quais foram os carros vencedores, mas está com dificuldades e pede sua ajuda. A sua tarefa, neste problema, é determinar os três carros melhor colocados, fornecidos os tempos que cada carro levou para completar cada volta da corrida.

No segundo caso de teste abaixo, existem 5 carros numa corrida de duas voltas. Os tempos de cada carro em cada volta foram como na tabela a seguir.

	Volta 1	Volta 2	Tempo total
Carro 1	3	7	10
Carro 2	2	5	7
Carro 3	1	1	2
Carro 4	15	2	17
Carro 5	2	2	4

Sendo assim, o vencedor foi o carro 3 (com um tempo total de 2), seguido pelo carro 5 (com um tempo total de 4) e pelo carro 2 (com um tempo total de 7).

Entrada

A primeira linha da entrada contém dois inteiros N ($3 \leq N \leq 100$) e M ($1 \leq M \leq 100$) representando o número de carros e o número de voltas da corrida, respectivamente.

Cada uma das N linhas seguintes representa um carro: a primeira linha representa o primeiro carro, a segunda linha representa o segundo carro, e assim por diante. Cada linha contém M inteiros representando os tempos em cada volta da corrida: o primeiro inteiro é o tempo da primeira volta, o segundo inteiro é o tempo da segunda volta, e assim por diante ($1 \leq \text{qualquer número da entrada que represente o tempo de uma volta} \leq 10^6$).


Garante-se que não houve dois carros que gastaram o mesmo tempo para completar a corrida inteira.

Saída

A saída consiste de três linhas, contendo um único inteiro cada. A primeira linha contém o número do carro que ganhou a corrida, a segunda contém o número do segundo colocado e a terceira contém o número do terceiro colocado.

Exemplos de Entrada	Exemplos de Saída
3 1 1 2 3	1 2 3
5 2 3 7 2 5 1 1 15 2 2 2	3 5 2

OBI - Olimpíada Brasileira de Informática 2011 Fase 1 Nível 1

Por OBI - Olimpíada Brasileira de Informática 2011  Brazil

G. Converting to Hexadecimal

Time limit: 1s

Os dados armazenados no computador estão em binário. Uma forma econômica de ver estes números é usar a base 16 (hexadecimal).

Sua tarefa consiste em escrever um programa que, dado um número natural na base 10, mostre sua representação em hexadecimal.

Entrada


A entrada é um número inteiro positivo **V** na base 10 ($1 \leq V \leq 2 \times 10^9$).

Saída

A saída é o mesmo número **V** na base 16 em uma única linha (não esqueça do caractere de fim-de-linha). Use letras maiúsculas, conforme os exemplos.

Exemplos de Entrada	Exemplos de Saída
10	A
15	F
16	10
31	1F
65535	FFFF

Prova 1 de Programação de Computadores da UNILA (2015/2)

Por M.C. Pinto, UNILA  Brazil

H. Mergulho

Time limit: 1s

O recente terremoto em Nlogônia não chegou a afetar muito as edificações da capital, principal epicentro do abalo. Mas os cientistas detectaram que o principal dique de contenção teve um dano significativo na sua parte subterrânea que, se não for consertado rapidamente, pode causar o seu desmoronamento, com a consequente inundação de toda a capital.

O conserto deve ser feito por mergulhadores, a uma grande profundidade, em condições extremamente difíceis e perigosas. Mas como é a sobrevivência da própria cidade que está em jogo, seus moradores acudiram em grande número como voluntários para essa perigosa missão.

Como é tradicional em missões perigosas, cada mergulhador recebeu no início do mergulho uma pequena placa com um número de identificação. Ao terminar o mergulho, os voluntários devolviam a placa de identificação, colocando-a em um repositório.

O dique voltou a ser seguro, mas aparentemente alguns voluntários não voltaram do mergulho. Você foi contratado para a penosa tarefa de, dadas as placas colocadas no repositório, determinar quais voluntários perderam a vida salvando a cidade.

Entrada

A entrada contém vários casos de teste e termina com EOF. Cada caso de teste é composto de duas linhas. A primeira linha contém dois inteiros **N** e **R** ($1 \leq R \leq N \leq 10^4$), indicando respectivamente o número de voluntários que mergulhou e o número de voluntários que retornou do mergulho. Os voluntários são identificados por números de 1 a N. A segunda linha da entrada contém R inteiros, indicando os voluntários que retornaram do mergulho (ao menos um voluntário retorna do mergulho).

Saída

Seu programa deve produzir uma única linha para cada caso de teste, contendo os identificadores dos voluntários que não retornaram do mergulho, na ordem crescente de suas identificações. Deixe um espaço em branco após cada identificador (note que isto significa que deve haver um espaço em branco também após o último identificador). Se todos os voluntários retornaram do mergulho, imprima apenas o caractere '*' (asterisco).

Exemplo de Entrada	Exemplo de Saída
5 3 3 1 5 6 6 6 1 3 2 5 4	2 4 *

Maratona de Programação da SBC 2013.

I. Digitando no Telefone Celular

Time limit: 3s

Uma equipe de pesquisadores está desenvolvendo uma nova tecnologia para economizar tempo ao digitar mensagens de texto em dispositivos móveis. Eles estão trabalhando em um novo modelo que tem um teclado completo, assim os usuários podem digitar qualquer letra pressionando a tecla correspondente. Desta forma, um usuário precisa pressionar P teclas para digitar uma palavra de comprimento P .

No entanto, isto não é suficientemente rápido. A equipe vai montar um dicionário de palavras comuns que um usuário pode digitar. O objetivo é reduzir o número médio de teclas pressionadas necessárias para digitar palavras que constam no dicionário. Durante a digitação de uma palavra, sempre que houver apenas uma possibilidade para a seguinte letra, o sistema de telefone celular irá introduzi-la automaticamente, sem a necessidade de digitação. Para ser mais preciso, o comportamento do sistema de telefone celular irá ser determinado pelas seguintes regras:

1. O sistema nunca irá adivinhar a primeira letra de uma palavra, ou seja, para a primeira letra sempre terá que ser pressionada a tecla correspondente.

2. Se uma sucessão não-vazia de letras $c_1c_2\dots c_n$ for introduzida, e houver uma letra c tal que cada palavra no dicionário que começa com $c_1c_2\dots c_n$ também começa com $c_1c_2\dots c_nc$, em seguida, o sistema coloca a entrada c automaticamente, sem a necessidade de uma combinação de teclas. Caso contrário, o sistema aguarda o usuário.

Por exemplo, se o dicionário é composto das palavras "hello", "hell", "heaven" e "goodbye", e o usuário pressiona "h", o sistema colocará a letra "e" automaticamente, porque cada palavra que começa com "h" também começa com "he". No entanto, uma vez que existem palavras que começam com "hel" e com "hea", o sistema precisa agora esperar a próxima digitação do usuário. Se o usuário pressionar então o "l", obtendo-se a palavra parcial "hel", o sistema de entrada incluirá um segundo "l" automaticamente. Quando se tem o "hell" como entrada, o sistema não pode supor nada, porque é possível que a palavra terminou, ou também é possível que o usuário pode querer pressionar "o" para obter "hello". Desta forma, para digitar a palavra "hello" o usuário precisa de três teclas, "hell" exige duas e "heaven" também requer duas, porque quando a entrada é "hea" o sistema pode colocar automaticamente o restante da palavra aplicando repetidamente a segunda regra. Da mesma forma, a palavra "goodbye" precisa de apenas uma tecla, porque depois de pressionar a inicial "g", a sistema irá preencher automaticamente a palavra inteira. Neste exemplo, o número médio de teclas necessário digitar uma palavra no dicionário é, então, $(3 + 2 + 2 + 1) / 4 = 2.00$.

Dado um determinado dicionário, sua tarefa então é calcular o número médio de teclas necessárias para escrever uma palavra no dicionário com o novo sistema para celular desenvolvido pelos pesquisadores.

Entrada

Cada caso de teste é descrito por várias linhas. A primeira linha contém um número inteiro N representando o número de palavras no dicionário ($1 \leq N \leq 10^5$). Cada uma das próximas N linhas contém uma string não-vazia de no máximo 80 letras minúsculas do alfabeto inglês, representando uma palavra no dicionário. Dentro de cada caso de teste todas as palavras são diferentes, e a soma dos comprimentos de todas as palavras é, no máximo, 10^6 .

Saída

Para cada caso de teste de entrada imprima um número com duas casas decimais (arredondado caso necessário) que representa o número médio de pressionamentos de tecla necessários para digitar uma palavra no dicionário.

Exemplo de Entrada	Exemplo de Saída
4	2.00
hello	1.67
hell	2.71
heaven	
goodbye	
3	
hi	
he	
h	
7	
structure	
structures	
ride	
riders	
stress	
solstice	
ridiculous	

ACM/ICPC South America Contest 2012.

Por Pablo Ariel Heiber, UBA  Argentina