Maratona Baiana de Programação 2019

Universidade Católica de Salvador

Instruções

- Sobre a prova:
  1. A prova tem duração de 5 horas;
- Sobre a entrada:
  1. A entrada do seu programa deve ser lida da entrada padrão (standard input);
  2. Quando uma linha da entrada contém vários valores, estes são separados por um único espaço em branco, a menos que explicitado o contrário no enunciado do problema;
  3. Toda linha da entrada terminará com um caractere final-de-linha.
- Sobre a saída:
  1. A saída de seu programa deve ser escrita na saída padrão;
  2. Quando uma linha da saída contém vários valores, estes devem ser separados por um único espaço em branco, a menos que explicitado o contrário no enunciado do problema;
  3. Toda linha da saída deve terminar com um caractere final-de-linha.
- Sobre as submissões:
  1. Você deve submeter o código fonte das suas soluções;
  2. Os comandos utilizados para compilação serão:
     (a) C11: gcc -std=c11 -lm 03 arquivo.c
     (b) C++11: g++ -std=c++11 -lm -O3 arquivo.cpp
     (c) Java: javac arquivo.java
     (d) Python2: python2 arquivo.py
     (e) Python3: python3 arquivo.py
  3. Para linguagem Java, o nome da classe principal do arquivo deve ser **Main**. A classe deve ser pública.

Este caderno contém 12 problemas e 15 páginas

08 de Junho de 2019

Problema A

Multiplicação de Matrizes

Tempo limite: 1 segundo
Limite de memória: 256 megabytes

João, aluno exímio do Departamento de Computação da UCSal, tem estudado multiplicação de matrizes para a prova da álgebra linear que vem se aproximando. Durante o processo, João decidiu unir a necessidade de estudar à sua paixão por programar. O jovem, então, pensou num problema simples: escrever um programa que imprime a soma de todos os elementos da matriz $A \times B$, onde A é uma matriz de dimensões $n \times m$, B uma matriz de dimensões $m \times h$.

João conseguiu resolver o problema que propôs muito rápido. Para deixar o problema ainda mais desafiador, João definiu que a matriz A seria composta por elementos $a_{ij} = iX + jY + Z$ e a matriz B seria composta por elementos $b_{ij} = iP + jQ + R$, onde X, Y, Z, P, Q, R são números pré-definidos.

João percebeu que agora o problema ficou muito mais difícil do que esperava. Você consegue ajudá-lo a resolver tal problema?

Note que a notação a_{ij} indica o elemento na linha i e na coluna j da matriz A, e que as linhas e colunas são numeradas a partir de zero.

Entrada

A primeira linha da entrada contém três inteiros n, m, h ($1 \leq n, m, h \leq 10^9$) – os números que definem as dimensões das matrizes A, B.

A segunda linha contém três inteiros X, Y, Z ($0 \leq X, Y, Z \leq 10^9$) – as constantes que definem os elementos da matriz A.

A terceira linha contém três inteiros P, Q, R ($0 \leq P, Q, R \leq 10^9$) – as constantes que definem os elementos da matriz B.

Saída

Imprima uma única linha contendo um inteiro que representa a soma de todos os elementos da matriz $A \times B$. Como a resposta pode ser muito grande, imprima o resto de sua divisão por $10^9 + 7$.

Exemplos

standard input	standard output
1 3 1 0 0 1 0 0 1	3
3 1 3 0 0 1 0 0 1	9
4 2 5 2 1 2 1 3 0	1440

Problema B

Escola de Verão de Programação

Tempo limite: 2 segundos
Limite de memória: 256 megabytes

No estado de Aihab, localizado na região sudoeste do país Lisarb, não falta investimento para maratonas de programação. Por conta disso, hoje há centenas de competidores participando ativamente dos vários eventos organizados mensalmente por todo o estado. Essa modalidade cresceu tanto que foi criado um cadastro estadual com todo o histórico desses competidores, bem como uma pontuação para o seu desempenho.

Uma das novidades para o ano de 9102 será a criação de uma escola de verão de programação no estado sem custos para os participantes. Infelizmente, não há recursos para custear a participação de todos os competidores cadastrados. Portanto, cada competidor preencherá em um formulário o seu nome, cidade de residência, e valor em Imaginários necessário para sua participação (Imaginário é a moeda de Lisarb). Depois disso, a organização selecionará os participantes de acordo com os seguintes critérios:

1. Não pode haver mais de um participante por cidade;
2. O valor total gasto em Imaginários não poderá ultrapassar o limite de verba do evento;
3. A soma das pontuações dos competidores escolhidos deve ser a maior possível, respeitando-se as duas restrições anteriores

Dado que as informações fornecidas pelos competidores combinadas com o cadastro estadual possuem todas as informações necessárias para realizar tal seleção, o seu time de desenvolvedores foi selecionado pelos organizadores para implementar o sistema de seleção especificado.

Entrada

A entrada é iniciada por dois inteiros, N e T ($1 \leq N \leq 1000$, $1 \leq T \leq 10000$), representando a quantidade de competidores presentes no cadastro estadual e o limite de verba em Imaginários disponível para a realização da escola de verão. Seguem N linhas, cada uma contendo as informações de um dos competidores. Essas informações são, em ordem, um inteiro V ($0 \leq V \leq 1000$) representando o valor em Imaginários requisitado pelo competidor, um inteiro P ($0 \leq P \leq 2500$) representando a pontuação desse competidor no cadastro, e uma cadeia de caracteres C ($1 \leq |C| \leq 50$), formada apenas por letras minúsculas e não possui espaços ou acentos, indicando a cidade de residência do competidor.

Saída

A saída do programa deve ser a soma das pontuações dos competidores escolhidos para a escola de verão.

Exemplos

standard input	standard output
3 1000 1000 1500 salvador 1000 2000 jequie 0 1000 cachoeira	3000
2 1000 500 1500 laurodefreitas 500 2000 vitoriadaconquista	3500
2 1000 500 1500 ilheus 500 2000 ilheus	2000

Problema C

Tree Parcel Service

Tempo limite: 2 segundos
Limite de memória: 256 megabytes

John é um antigo funcionário da TPS (Tree Parcel Service), a maior companhia de transporte de encomendas de Treeland. Treeland é uma pacata cidade dividida em n distritos, numerados de 1 a n.

John trabalha somente às segundas-feiras e, por isso, prefere ficar o máximo possível no trabalho. Num dia de trabalho, ele escolhe uma permutação p dos distritos de Treeland e começa a entregar todas as encomendas da seguinte forma: primeiro, entrega todos os pacotes destinados a moradores do distrito p_1, então vai até p_2 e entrega todos os pacotes destinados a moradores do distrito p_2, e assim por diante, até que todos os distritos sejam visitados e que todos os pacotes sejam corretamente entregues.

Treeland é uma cidade enorme e, portanto, é impraticável para John entregar todos os pacotes a pé. A TPS oferece a seus mais exímios funcionários, como John, um veículo de entrega extremamente peculiar chamado Treeship. Um Treeship não é como um carro – ele não é feito para percorrer estradas comuns. Na verdade, esses veículos se movem através de $n - 1$ túneis bidirecionais construídos pela própria TPS. Tais túneis foram planejados de forma que existe **exatamente uma** sequência de túneis conectando qualquer par de distritos.

Cada um desses túneis tem um custo associado. Mais especificamente, o i-ésimo dos túneis tem um custo c_i. Esses veículos se movem através de túneis dedicados e, por isso, podem atingir velocidades estrondosas. Na verdade, o tempo que o veículo demora para se mover de um distrito u para outro distrito v é dado pelo maior custo de um túnel no caminho único entre u e v.

Sua tarefa é encontrar uma permutação p que maximize o tempo que John irá trabalhar caso tenha pacotes para entregar em todos os distritos de Treeland.

Entrada

A primeira linha contém um inteiro n ($2 \leq n \leq 10^5$) – o número de distritos em Treeland.

As próximas $n - 1$ linhas descrevem os túneis da TPS. A i-ésima delas contém três inteiros a_i, b_i, c_i ($1 \leq a_i, b_i \leq n$, $1 \leq c_i \leq 10^9$) – os distritos a_i, b_i que são diretamente conectados por esse túnel e seu custo c_i.

Nenhum par (a_i, b_i) aparecerá duas vezes na entrada.

Saída

A saída deve conter uma única linha com n inteiros separados por espaços – uma permutação p dos inteiros $1, 2, \ldots, n$ que maximize o tempo que John irá levar para concluir seu dia de trabalho.

Exemplos

standard input	standard output
5 1 2 1 2 3 1 3 4 1 4 5 1	2 5 1 4 3
6 1 3 1 2 3 1 3 4 1 4 5 1 4 6 1	4 3 6 1 5 2
6 1 3 5 2 3 1 3 4 1 4 5 1 4 6 1	5 3 1 2 6 4
6 1 3 1 2 3 1 3 4 1 4 5 5 4 6 10	4 3 6 1 5 2
7 6 2 10 7 3 1 5 1 1 7 6 1 2 5 1 4 1 1	1 6 5 7 2 3 4
5 5 4 4 4 2 3 5 3 2 5 1 1	1 4 5 2 3

Problema D

Formatando Datas

Tempo limite: 1 segundo
Limite de memória: 256 megabytes

Rogério gosta muito de viajar, mas o que mais lhe confunde nos diferentes países e idiomas de Bytelândia é o formato de representação de datas. Por exemplo, os seguintes formatos são possíveis: dia-mês-ano, mês-dia-ano, ano-mês-dia, mês-ano-dia, e assim por diante. Em outras palavras, qualquer permutação de dia-mês-ano pode ser válida. Curiosamente, em Bytelândia, todo mês tem 30 dias e todo ano tem 12 meses.

Rogério percebeu que analisando algumas data é possível deduzir um único formato para qual a data seja válida, ou pelo menos eliminar algumas opções. Por exemplo, a data 20-08-95 só é válida no formato dia-mês-ano. Já a data 13-16-19 é inválida em qualquer formato.

Rogério tem trabalhado muito, o que ocupa bastante o seu corpo e mente, e por isso não tem mais paciência para analisar as datas e descobrir o seu formato. Ele pede sua ajuda para fazer um programa que, dado três números, responda se existe um formato em que a data seja válida.

Entrada

A primeira e única linha de entrada possui 3 números inteiros, separados por espaço, que representam a data a ser analisada. Os números da entrada estão entre 1 e 99.

Saída

A saída esperada possui apenas uma linha. Caso existam várias formas de rearranjar os três números em formatos de data válidos, seu programa deve imprimir "ambigua". Caso a data não seja válida em nenhum formato, a saída deve conter apenas a palavra "invalida". Se for possível deduzir apenas um formato válido, o seu programa deve imprimir esse formato, de acordo com a ordem, como por exemplo "d-m-a" para a entrada "23 1 95", em que "d" é dia, "m" é mês e "a" é ano.

Exemplos

standard input	standard output
30 1 95	d-m-a
4 4 4	ambigua
22 14 35	invalida

Problema E

Amortizando Explosões

Tempo limite: 2 segundos
Limite de memória: 256 megabytes

Gigabyte é o nome da principal rede de usinas termoelétricas de Bytelândia. As usinas foram construídas sobre uma rede ancestral de túneis subterrâneos, que há séculos atrás facilitava o transporte de metais quando a cidade ainda era um pólo de mineração.

Entretanto, após alguns anos de operação da Gigabyte, a *Nanobyte*, empresa responsável pela construção e manutenção das usinas, notou que a existência dos túneis poderia acarretar numa terrível reação em cadeia caso os reatores das usinas explodissem.

A rede de túneis pode ser vista como um *grid*. As usinas são identificadas por números de 1 a n e se encontram em pontos específicos da rede de túneis chamados de interseções, representados por coordenadas na forma (x, y), onde x, y são números inteiros. Toda interseção é ligada por túneis somente às oito interseções vizinhas (pontos cardeais e colaterais).

Depois de muita pressão do governo local, o chefe de operações da Nanobyte, *Robert Tarjan*, decidiu instalar **exatamente um** amortizador de explosões em cada um dos túneis.

Um amortizador de explosões é um dispositivo capaz de amenizar o impacto de uma explosão. Um amortizador geralmente não é capaz de anular todo o efeito causado pela explosão de um reator. Tarjan avaliou que é necessária a ação de **no mínimo** k amortizadores para que exista garantia de que o efeito de uma explosão foi devidamente contido. A partir disso, Tarjan chegou a seguinte (e possivelmente catastrófica) conclusão: se uma usina i explodir e existir uma outra usina j a menos de k túneis de distância da usina i, a usina j também explodirá.

Você foi contratado pelo governo para escrever um programa que, para cada usina i presente na Gigabyte, informe se, caso o reator da usina i exploda, existe a garantia que **nenhuma outra usina da rede será danificada**.

Entrada

A primeira linha contém dois inteiros n, k ($1 \leq n, k \leq 10^5$) – o número de usinas presentes na rede e a quantidade mínima de amortizadores necessários para conter uma explosão, respectivamente.

A entrada é composta por mais n linhas. A i-ésima dessas contém dois inteiros espaçados x_i, y_i ($1 \leq x_i, y_i \leq 10^5$) – as coordenadas da i-ésima usina.

Saída

O seu programa deve imprimir n linhas. A i-ésima destas deve conter `NO`, caso a explosão do reator da usina i não possa causar uma reação em cadeia, e `YES`, caso contrário.

Exemplo

standard input	standard output
4 2 1 1 2 2 1 2 4 4	YES YES YES NO

Problema F

Jogo dos 7 errors

Tempo limite: 1 segundo
Limite de memória: 256 megabytes

Você é bom em perceber detalhes? O jogo dos 7 erros consiste em encontrar 7 diferenças entre duas imagens, e sempre foi uma brincadeira muito muito famosa nos jornais impressos ou nos livros infantis. Para facilitar a criação de desafios para seus leitores, um jornal decidiu criar um gerador automático de jogos de 7 erros criando quadros aleatórios de letras e gerando uma cópia com 7 letras trocadas. Um exemplo pode ser visto abaixo:

Cabia aos leitores então apontar onde estavam as 7 letras alteradas. Embora pareça simples, à medida que o tamanho do quadro era aumentado, maior era a dificuldade. Tanto que criou uma rivalidade mortal entre Robério e Cordeiro. Robério apostou que conseguiria encontrar as 7 mudanças antes de Cordeiro. Para isso, ele desenvolveu um aplicativo que faz a análise de uma foto desses quadros e reconhece automaticamente os caracteres contidos neles. Agora ele só precisa identificar a posição das mudanças. Você pode ajudá-lo?

Entrada

A entrada é iniciada por dois inteiros, L e C ($1 \leq L \leq 100$, $1 \leq C \leq 100$, $7 \leq L * C \leq 10000$), representando a quantidade de linhas e colunas em um quadro, respectivamente. Seguem então L linhas, cada uma contendo C letras minúsculas, representando o primeiro quadro. Depois, outras L linhas, cada uma contendo C letras minúsculas, representam o segundo quadro.

Saída

A saída do programa deve ser a posição das 7 letras diferentes nos dois quadros. Essas posições devem ser ordenadas de forma não decrescente pela linha, e, em caso de empate, de forma não decrescente pela coluna.

Exemplos

standard input	standard output
2 4 abcd efgh wxyz tugv	1 1 1 2 1 3 1 4 2 1 2 2 2 4
1 7 abcdefg bcdefgh	1 1 1 2 1 3 1 4 1 5 1 6 1 7

Problema G

Mega-Sena

Tempo limite: 1 segundo
Limite de memória: 256 megabytes

A mega-sena é o jogo de apostas que movimenta a maior quantidade de dinheiro no país. Basicamente, 6 números de 1 a 60 sem repetição são sorteados semanalmente, e qualquer pessoa pode fazer apostas com 6, 7, 8, 9 ou 10 números nessa faixa. Obviamente, o preço sobe vertiginosamente a cada número extra apostado.

São premiadas apostas que acertam 4, 5 ou 6 números sorteados, respectivamente chamadas de quadras, quinas e apostas vencedoras.

No entanto, esse tipo de sorteio é visto por poucas pessoas como um imposto para pessoas que não entendem muito de estatística. Mais precisamente, existe uma chance em 50.063.860 de uma aposta de 6 números sair vencedora.

Um grupo de estudantes resolveu estudar esse problema experimentalmente, e saiu solicitando apostas contendo de 6 a 10 números a diferentes pessoas. Agora, eles realizaram um sorteio e desejam saber quantas apostas foram vencedoras em cada categoria de premiação. Claramente a quantidades de apostas recolhida é grande demais para fazer a análise manualmente, e eles precisam da sua ajuda para automatizar esse experimento.

Entrada

A entrada é iniciada por seis inteiros representando os valores sorteados de 1 a 60 em ordem crescente. Na próxima linha, um inteiro N indica quantas apostas foram recolhidas pelos estudantes. As próximas N linhas seguem o padrão: um inteiro M indica quantos números estão contidos na aposta, seguido por M valores de 1 a 60 em ordem crescente.

1. $1 \leq N \leq 1000$;
2. $6 \leq M \leq 10$.

Saída

A saída do programa deve conter uma linha com três inteiros separados por espaço indicando, respectivamente, a quantidade de apostas vencedoras, quinas e quadras entre as apostas na entrada.

Exemplos

standard input	standard output
10 20 30 40 50 60 6 6 1 2 3 4 5 6 7 4 5 6 7 8 9 10 8 10 20 30 40 41 42 43 44 6 11 22 33 44 55 59 9 23 24 25 26 27 31 32 33 34 6 7 20 21 30 50 60	0 0 2
1 2 3 4 5 6 3 6 1 2 3 4 5 6 6 2 3 4 5 6 7 6 3 4 5 6 7 8	1 1 1

Problema H

Par-par-ou-ímpar-ímpar

Tempo limite: 1 segundo
Limite de memória: 256 megabytes

O Campeonato Mundial de par-ou-ímpar ocorre todo dia em escolas por todo mundo. Criado desde que a primeira escola do mundo foi feita, é um esporte conhecido e praticado em quase todo mundo. Esse campe- onato pode ocorrer quando dois ou mais jogadores estão dispostos a jogar. Os prêmios desses campeonatos variam de uma coxinha no recreio até uma BMW.

Para dar mais emoção ao jogo, Jacinto Pena criou uma variação do famoso jogo: o não tão famoso par-par-ou-ímpar-ímpar. Nessa variação, um número natural é escolhido por cada participante, e a soma deles é par-par se todos os seus divisores **maiores que 1** são pares. Se a soma não for par-par, ela é ímpar-ímpar. Em particular, note que o número 1 é par-par.

Jacinto, infelizmente, criou um jogo que favorece o jogador que escolhe ímpar-ímpar. Dado um limite máximo para a soma dos números escolhidos, você pode dizer a quantidade de valores que darão a vitória ao jogador que escolheu par-par?

Entrada

A entrada possui um único inteiro N ($0 \leq N \leq 10^{18}$) representando o limite máximo para a soma dos números escolhidos pelos dois participantes.

Saída

A saída do programa deve ser a quantidade de valores de 0 até N – a soma dos valores escolhidos pelos dois jogadores – que darão a vitória ao jogador que escolheu par-par.

Exemplos

standard input	standard output
4	3
1	1

Problema I

Árvore de Palavras

Tempo limite: 2 segundos
Limite de memória: 256 megabytes

Joanna é uma garota que adora jogos de tabuleiro. Após ter recebido de presente o jogo *Ricochet Robots* (um jogo de robôs deslizantes extremamente desafiador), Joanna passou a se interessar por outros jogos que envolvam desafios de raciocínio. Durante sua pesquisa, Joanna descobriu o jogo Árvore de Palavras.

Neste jogo, uma árvore enraizada aleatória é montada, com n vértices numerados de 1 a n e $n-1$ arestas. A raíz da árvore é o vértice de número 1. O vértice-pai do i-ésimo vértice é dado por p_i ($i > 1$, $p_i < i$). Após isto, uma letra aleatória do alfabeto latino (inclui k, w, y) é colocada em cada uma das $n-1$ arestas.

O objetivo do jogo é formar palavras. Uma palavra s pode ser formada se existe uma sequência de vértices $v_1, v_2, \ldots, v_n$ t.q. $p_{v_i} = v_{i+1}$ e as letras presentes nas arestas de v_1 até v_n, quando concatenadas, resultem na palavra s.

Dada uma Árvore de Palavras, Joanna se perguntou, dentre as palavras que podem ser formadas, qual aquela lexicograficamente maior. Você pode ajudar Joanna a resolver esse desafio?

Entrada

A primeira linha da entrada contém um único inteiro n ($2 \leq n \leq 10^5$) – a quantidade de vértices da Árvore de Palavras.

A segunda linha contém $n-1$ inteiros separados por espaços. O i-ésimo deles é p_{i+1} ($1 \leq p_i < i+1$) – o vértice-pai do vértice $i+1$.

A terceira linha contém uma sequência de $n-1$ caracteres. O i-ésimo deles é o caractere colocado na aresta que conecta os vértices p_{i+1} e $i+1$.

Os caracteres são caracteres latinos e minúsculos.

Saída

Imprima numa única linha a palavra que pode ser formada que é lexicograficamente maior.

Exemplos

standard input	standard output
3 1 2 ab	ba
3 1 1 ab	b
4 1 1 2 akk	ka

Problema J

Wet in the Wild

Tempo limite: 2 segundos
Limite de memória: 256 megabytes

Um novo parque da franquia *Wet in the Wild* será construído em Salvador. O antigo parque da franquia, agora somente usado durante os eternos ensaios de Carnaval, será demolido e dará lugar a um moderno parque.

Um dos novos brinquedos do parque será o Toboágua Gigante. Alice, a engenheira responsável pelo projeto, preparou um desenho preliminar do brinquedo que agradou bastante os investidores. O brinquedo consiste de m tubos de água. Cada um destes tubos conecta pares de interseções dentre n pontos de interseção (numerados de 1 a n) escolhidos pelo engenheiro.

Como ditam os princípios da Física, em um tubo que conecta as interseções (u, v), a água irá fluir de u para v se a altura da interseção u for **maior** que a altura da interseção v. Caso contrário, a água fluirá de v para u. É possível assumir que duas interseções distintas nunca estarão a mesma altura do chão.

Alice deseja escolher alturas distintas para cada uma das interseções do Toboágua Gigante. Entretanto, o brinquedo deve seguir uma importante restrição: deve existir uma **única** interseção a de onde **toda água do brinquedo** flui (a fonte), e uma **única** interseção b para onde **toda água do brinquedo** flui (o poço). Note que por todas as interseções do brinquedo deve passar água.

Dadas as informações sobre os m tubos do Toboágua Gigante, ajude Alice a descobrir quantos pares de interseções a, b existem tal que seja possível escolher alturas para cada uma das interseções de forma que a seja a fonte e b seja o poço.

Entrada

A primeira linha da entrada contém dois inteiros n, m ($2 \leq n \leq 10^5$, $1 \leq m \leq 10^5$) – a quantidade de interseções e de tubos do Toboágua Gigante, respectivamente.

As próximas m linhas contém as descrições de cada um dos tubos. A i-ésima delas contém dois inteiros u_i, v_i ($1 \leq u_i, v_i \leq n$) – as interseções ligadas pelo i-ésimo tubo.

É garantido que todo par de interseções está conectado indiretamente por tubos.

Saída

Imprima um inteiro numa única linha – a quantidade de pares `(fonte, poço)` possíveis.

Exemplos

standard input	standard output
2 1 1 2	2
3 2 1 2 1 3	2
3 3 1 2 1 3 3 2	6
4 5 1 2 1 3 3 4 2 4 4 2	12
4 3 1 2 2 3 2 4	0

Problema K

Copa do Mundo

Tempo limite: 1.5 segundos
Limite de memória: 256 megabytes

A Copa do Mundo de Futebol Feminino está começando! Vai ter Brasil, vai ter EUA, vai ter Alemanha, vai ter Marta, e vai ter muito gol! Júlia, fã fervorosa do futebol apresentado pela seleção brasileira, já separou sua pipoca pra assistir o jogo do Brasil esse domingo.

Enquanto o jogo não chega, Júlia decidiu juntar sua empolgação com a Copa do Mundo a sua paixão por programação. A programadora construiu um programa que, dados os times de um grupo e os resultados dos jogos que foram jogados até o momento, mostra aqueles classificados para as oitavas-de-finais do torneio.

No futebol, cada vitória confere três pontos ao time vencedor. Um empate confere um ponto a ambos os times. A quantidade de pontos é o critério usado para ranquear os times. Isto é, aquele que tem mais pontos é o primeiro do grupo, e assim por diante. Como critério de desempate, o número de vitórias é utilizado. Caso o empate persista, o saldo de gols (diferença entre gols marcados e gols sofridos) é utilizado. Caso o empate ainda persista, o time que vem primeiro no dicionário é colocado à frente.

Na Copa, os grupos são formados por 4 times. Os 2 primeiros de cada grupo avançam para próxima fase. Júlia te desafiou a escrever um programa equivalente ao dela. Dados um grupo e o resultado dos jogos, determine os 2 primeiros times do grupo.

Entrada

A primeira linha contém 4 palavras separadas por espaço – os nomes dos times do grupo. Os nomes serão compostos somente por letras minúsculas.

A segunda linha contém um inteiro n – a quantidade de jogos jogados até o momento.

As próximas n linhas contém a descrição de cada um dos jogos. A i-ésima delas contém uma palavra s_1, um inteiro g_1, uma palavra s_2 e um inteiro g_2, indicando que o time s_1 marcou g_1 gols neste jogo e que o time s_2 marcou g_2 gols no mesmo.

- $1 \leq n \leq 10$;
- $0 \leq g_i \leq 10$.

Saída

Na primeira linha, imprima o nome do primeiro lugar do grupo.

Na segunda linha, imprima o nome do segundo lugar do grupo.

Exemplo

standard input	standard output
alemanha brasil china coreia 6 brasil 7 alemanha 1 coreia 2 china 1 coreia 2 alemanha 2 brasil 1 china 0 brasil 1 coreia 1 alemanha 3 china 2	brasil coreia

Problema L

Dominó

Tempo limite: 1 segundo
Limite de memória: 256 megabytes

Jogar dominó em Salvador é algo tão ou mais famoso do que ir atrás do trio no Carnaval. Suas esquinas, repletas de cadeiras de metal e tábuas de madeira apoiadas sobre as pernas de verdadeiros profissionais do esporte, revelam a paixão do soteropolitano pelo jogo. Rola até apostar abadá na mesa. Onde já se viu, rapaz?

Dominó baiano é um verdadeiro show. É buchada, é barrigada, é Chico Romero e, é claro, até lasque traque, do francês *lasquinê*. Entretanto, depois de dominar a linguagem complexa das mesas de dominó, uma das principais habilidades do jogo é a de contar peças.

Sua tarefa neste problema, muito mais simples que jogar dominó com um baiano raíz, é escrever um programa que conte quantas peças de cada número estão em jogo.

Entrada

A entrada contém 7 linhas com 7 números cada – uma matriz composta por zeros e uns. O número na j-ésima coluna da i-ésima linha indica se a peça com os números i e j (numerados a partir de 0) está em jogo (1, um) ou não (0, zero).

Note que as peça i, j e j, i são na verdade as mesmas, e que isso estará refletido na matriz.

Saída

Imprima 7 linhas com a quantidade de peças de cada número que estão em jogo. A primeira linha deve conter a quantidade de peças com zero, a segunda de peças com um, a terceira de peças com dois, e assim por diante.

Exemplo

standard input	standard output
0 1 1 0 1 1 1	5
1 0 1 1 0 1 1	5
1 1 1 0 0 1 1	5
0 1 0 1 0 1 0	3
1 0 0 0 0 1 1	3
1 1 1 1 1 1 0	6
1 1 1 0 1 0 1	5