CADERNO DE PROBLEMAS

PROBLEMA A: ÁLBUM DE FIGURINHAS

Arquivo: `album.[cpp/c/java]`

Cor: azul

Limite de tempo: 1s

Descrição do problema

Aldo e Beto colecionam figurinhas do maior álbum de figurinhas do mundo! Esse álbum tem um bilhão de figurinhas diferentes para colecionar.

Devido ao tamanho do álbum, Aldo e Beto decidiram juntar forças para terminar a coleção o quanto antes. Eles querem descobrir quantas figurinhas distintas eles possuem juntos, mas, devido ao volume de figurinhas que já possuem, essa tarefa se tornou difícil demais.

Para ajuda-los você resolveu criar um programa que, dada as listas de figurinhas de cada um, mostre a quantidade de figurinhas distintas que possuem juntos.

Entrada

A entrada é composta de vários casos de teste. Cada caso de teste possui três linhas. A primeira linha de um caso de entrada contém dois inteiros A e B ($1 \leq A, B \leq 10^5$) que representam a quantidade de figurinhas que Aldo e Beto possuem, respectivamente. A segunda linda de um caso de teste contém A inteiros separados por espaço que representam os identificadores das figurinhas que Aldo possui. A terceira linda de um caso de teste contém B inteiros separados por espaço que representam os identificadores das figurinhas que Beto possui. Um identificador F de uma figurinha pertence ao intervalo $1 \leq F \leq 10^9$. A entrada termina quando A = B = 0.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha contendo a quantidade de figurinhas distintas que Aldo e Beto possuem juntos.

Exemplos de testes

Entrada	**Saída**
3 4 1 2 3 4 5 6 7 3 4 1 2 3 1 2 3 4 3 4 2 1 2 1 2 1 2 0 0	7 4 2

PROBLEMA B: KITKATA

Arquivo: `kitkata.[cpp/c/java]`

Cor: preto

Limite de tempo: 1s

Descrição do problema

2014 passou, tivemos uma Copa no Mundo no Brasil após 64 anos e sabe o que ganhamos com isso? Um belo chocolate da campeã mundial, a Alemanha. E põe chocolate nisso!!! Não bastasse perder de goleada (7x1) ainda vimos um time dominar o meio de campo inteiro, sem dar a mínima chance para o Brasil jogar!

Acreditem ou não, isso frustrou muitos treinadores, jogadores e torcedores. Dado este histórico recente, um treinador inovador chamado Silva, decidiu olhar para o futebol vencedor das duas últimas copas, e após assistir todas os jogos da Espanha de 2010 e da Alemanha de 2014 ele decidiu criar um método inovador para dominar o meio de campo. Não reparem na demora, seu objetivo era apresentar este método a seleção brasileira em 2014, mas Silva sempre foi um treinador muito metódico e cauteloso.

O KitKata consiste em fazer com que a bola seja invertida de um lado a outro do campo de forma que os jogadores deem o passe mais curto possível, diminuindo assim a chance de erro no passe, e ao mesmo tempo fazendo com que a bola chegue o mais rápido possível do outro lado do campo.

Seu trabalho é ajudar Silva a mostrar a seus jogadores, a partir de imagens do campo como o time deve se comportar. Considere que a bola deve ser tocada do jogador mais acima em uma extremidade para o outro mais abaixo (eixo y).

Entrada

A entrada é constituída de um inteiro N representando o número de casos de teste ($0 <= N <= 1000$). Para cada caso você receberá 11 linhas contendo 3 inteiros indicando o número da camisa do jogador ($1 <= C <= 11$) e as coordenadas ($0 <= X <= 150$) ($0 <= Y <= 120$). Após isto você receberá a distância máxima aceitável entre 2 jogadores para que um passe seja executado ($30 <= D <= 60$).

Saída

Para caso de teste, imprima o caminho mais rápido que a bola pode percorrer passando pelo pé dos jogadores para sair de uma extremidade do campo e chegar a outra. Quando não houver caminho para fazer a bola chegar a outra extremidade deve ser exibida a mensagem “Jogadores mal posicionados! ”

Exemplos de testes

Entrada	**Saída**
2 1 37 0 2 11 18 3 30 15 4 39 20 5 29 43 6 10 38 7 59 29 8 49 63 9 52 63 10 58 60 11 36 99 30 1 36 7 2 27 21 3 52 24 4 44 14 5 19 39 6 23 45 7 45 44 8 57 77 9 47 89 10 23 76 11 48 91 56	Jogadores mal posicionados! 11 7 1

PROBLEMA C: YO HO YO HO

Arquivo: `pirata.[cpp/c/java]`

Cor: cinza

Limite de tempo: 3s

Descrição do problema

P piratas de diferentes idades possuem um tesouro com M moedas.

No navio, eles decidem dividir as moedas usando o seguinte modelo: o pirata mais velho propõe como dividir as moedas e todos os piratas (incluindo o mais velho) votam a favor ou contra a proposta.

Se 50% ou mais dos piratas votarem a favor então as moedas serão divididas daquela maneira. Se a proposta não for aceita então o pirata que a fez será lançado ao mar e o processo será repetido com os piratas remanescentes.

Dada uma proposta, se a quantidade de moedas que um pirata receberá for a mesma independentemente de ele votar a favor ou contra, ele votará contra pois, devido a sua natureza sanguinária, desejará ver seu companheiro ser atirado ao mar.

Assumindo que todos os piratas são inteligentes, racionais, gananciosos e não desejam ser lançados ao mar (e também excelentes matemáticos), crie um programa que, dados os valores de P e M, descreva qual proposta será aprovada.

Entrada

A entrada é composta de vários casos de teste. Cada caso de teste possui uma linha contendo dois inteiros P e M ($1 \leq P < M \leq 10^6$) que representam a quantidade de piratas e a quantidade de moedas, respectivamente. A entrada termina com final de arquivo (EOF).

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha contendo P inteiros separados por espaço representando a quantidade de moedas que cada pirata receberá ordenados pelas idades dos piratas.

No terceiro caso abaixo o pirata mais velho receberá 98 moedas, o segundo e o quarto mais velhos não receberão moedas, o terceiro mais velho e o mais novo receberão uma moeda cada.

Entrada	**Saída**
1 10 2 3 5 100	10 3 0 98 0 1 0 1

PROBLEMA D: MÉDIA

Arquivo: `media.[cpp/c/java]`

Cor: laranja

Limite de tempo: 1s

Descrição do problema

Carlinhos precisa calcular a média aritmética de suas notas para saber se vai passar de ano.

Crie um programa que, dada as notas de Carlinhos, calcula e exiba a média aritmética de suas notas

Entrada

A entrada é composta de um caso de teste que contém quatro inteiros que representam as notas de Carlinhos. Cada nota N está no intervalo $0 \leq N \leq 10$.

Saída

O seu programa deve imprimir a média aritmética das notas de Carlinhos com duas casas decimais.

Exemplos de testes

Entrada	**Saída**
1 1 1 1	1.00

Entrada	**Saída**
6 9 7 5	6.75

PROBLEMA E: BAZINGA

Arquivo: `bazinga.[cpp/c/java]`

Cor: roxo

Limite de tempo: 1s

Descrição do problema

Sheldon Cooper é um ser bastante excêntrico! A cada novo dia surge uma mania que deixa seus amigos loucos da vida! Especialmente Penny! A loira que mora no apartamento em frente ao seu tem um estilo de vida e intelectual bem diferente do de Sheldon.

Mas ainda assim, eles são amigos, se é que podemos definir as amizades do Sheldon como amizades normais! Infelizmente na última semana Penny aprontou uma daquelas coisas que irritam demais o Sheldon. Ela deixou uma roupa na máquina de lavar, Sheldon não viu e se sentiu contaminado pelas suas roupas terem sido lavadas junto a uma única peça de roupa da Penny.

Como vingança, Sheldon invadiu o e-mail de Penny e criptografou suas mensagens em letras minúsculas deixando pistas que Penny jamais conseguiria seguir para recuperar suas mensagens. Acontece que a Penny te encontrou no Facebook e pediu uma ajuda. Será que você pode recuperar os e-mails dela?

A propósito, Sheldon achou muito fácil criptografar os dados da Penny, já que além do alfabeto ela utiliza apenas os caracteres vírgula, ponto, interrogação, exclamação e espaço em branco.

Entrada

A entrada é constituída de vários casos de teste. Cada caso de teste inicia com dois inteiros representando o número de pistas deixadas por Sheldon (0 <= N <= 100) e o número de mensagens a recuperar (0 <= M <= 100). A entrada é seguida por N linhas com as pistas. Cada pista é composta de duas palavras separadas pelo caractere '_' sendo a primeira o texto criptografado e a segunda o texto limpo. Em seguida são recebidas M mensagens. O programa é encerrado com M = N = 0.

Tanto as mensagens como as pistas possuem no máximo 200 caracteres cada.

Saída

A saída consiste em apresentar a mensagem recuperada.

Exemplos de testes

Entrada	**Saída**
5 2 uck!_nah. wnr?_r o? od p_uwdp mxef_yel, y_! pxuumfnpxuumfnpxuumyyy exrucw !!!dkxwxncwxnmro? 0 0	penny, penny, penny!!! leonard...where are you?

PROBLEMA F: ERA DE ULTRÃO

Arquivo: `ultrao.[cpp/c/java]`

Cor: rosa

Limite de tempo: 1s

Descrição do problema

O filme Era de Ultron está sendo um sucesso de crítica e bilheteria. Os fatos narrados no filme, entretanto, não passam de pura ficção. Na vida real, o terrível vilão artificial não foi criado por Tony Stark, nem envolve código alienígena. Na verdade, o criador de Ultron foi o Dr. Henry Pym. Diferente da ficção, Ultron ganhou a luta contra a humanidade e dominou o mundo.

A única saída foi apelar, como sempre os heróis fazem em situações extremas: Wolverine entrou na máquina do tempo do Dr. Destino e matou Pym antes dele criar o Ultron. Infelizmente, o mundo acabou pior, Wolverine volta de novo e impede ele mesmo de matar o Pym (que rolo!). Outra solução precisava surgir. Finalmente decidiu-se colocar tentar mudar a programação de Ultron para que ele não se tornasse o terrível vilão.

Como o Dr. Pym é suspeito para inserir alterações, (apesar de tudo, ele ama sua criação!), um conjunto de programadores foi convocado para salvar o mundo. Esse projeto foi chamado de iniciativa Ultrão. Você e sua equipe fazem parte desse seleto grupo, e estão responsáveis por atuar no módulo de humor do Ultron, para que ele não sinta tanta raiva da humanidade.

Dr. Pym forneceu detalhes sobre o módulo de humor do Ultron. Basicamente, o módulo é um cubo tridimensional de lado N. O centro do cubo é a coordenada (0,0,0). A ideia desse modelo é que no centro do cubo Ultron esteja com raiva e na superfície do cubo ele esteja calmo. Conforme ele vai vivenciando experiências, seu estado de humor vai caminhando para o centro ou superfície do cubo. Para simular mais ainda a humanidade do robô, Dr Pym resolveu criar este cubo de forma aleatória, sorteando seu tamanho e que tipo de conexão existe entre os elementos do cubo. O menor caminho entre o centro e a superfície do cubo é traçado e esse caminho se torna a máquina de estado de humor para o robô. O problema é que o ilustre Dr Pym não contava com o fato de que certos cubos gerados tornavam impossível acalmar o robô, se não existisse caminho do centro para a superfície ou, pior, o caminho mínimo fosse infinito...

Sua equipe deve, então, trabalhar em um módulo de pré-avaliação de cubos de humor do Ultron. Dado um cubo gerado aleatoriamente, você deve retornar o menor caminho do centro (0,0,0) para qualquer elemento da superfície. Caso não exista caminho ou exista um caminho infinito, retorne uma mensagem de aviso para descartar este cubo, que levará a extinção da humanidade!

Entrada

A entrada será composta de vários casos de teste. O primeiro inteiro T ($10 \leq T \leq 200$) indica quantos casos de teste existem. Cada caso de teste se inicia com um inteiro ímpar C ($3 \leq C \leq 11$), indicando o tamanho do lado do cubo. Em seguida, um inteiro A ($0 \leq A \leq 1000$) indica quantas conexões entre elementos dos cubos existem. Finalmente, uma quantidade A de 7 inteiros é apresentada, correspondendo as ligações. Os 3 primeiros inteiros são as coordenadas do elemento O de origem da ligação, os 3 inteiros seguintes são as coordenadas do elemento D de destino da ligação, e o último inteiro P ($-100 \leq C \leq 100$) indica o custo para se percorrer essa ligação. Importante notar que a ligação é unidirecional (do elemento O para o elemento D).

Saída

Para cada entrada existirá uma saída indicando o menor caminho do centro do cubo para qualquer elemento da superfície, com as coordenadas dos elementos separados por vírgula. Se não existir caminho ou um caminho infinito for encontrado, exiba uma mensagem “melhor nao usar este modulo de humor!”. Utilize um espaço em branco para separar a vírgula dos números.

Exemplos de testes

Entrada	**Saída**
4 5 4 0 0 0 0 0 1 -10 0 0 1 0 0 2 8 0 0 0 0 1 0 1 0 1 0 0 2 0 2 3 2 0 0 0 0 0 0 -2 0 0 0 0 0 1 3 5 6 0 0 0 0 0 1 10 0 0 1 0 0 2 2 0 0 0 0 1 0 1 0 1 0 0 1 1 3 0 1 1 0 0 1 2 0 0 1 0 0 0 4 5 6 0 0 0 0 0 1 5 0 0 1 0 0 2 2 0 0 0 0 1 0 1 0 1 0 0 1 1 -8 0 1 1 0 0 1 2 0 0 1 0 0 0 4	0 0 0 , 0 0 1 , 0 0 2 melhor nao usar este modulo de humor! 0 0 0 , 0 1 0 , 0 1 1 , 0 0 1 , 0 0 2 melhor nao usar este modulo de humor!

PROBLEMA G: GAME OF THRONES

Arquivo: `game.[cpp/c/java]`

Cor: verde

Limite de tempo: 1s

Descrição do problema

Game of Thrones é o primeiro livro da série de fantasia épica, as crônicas de gelo e fogo, escrita pelo norte-americano George R. R. Martin. O livro dá nome a vários itens derivados baseados na saga, incluindo uma série televisiva, intitulada Game of Thrones que estreou em 2011 nos Estados Unidos.

A saga traz disputas entre casas (família) para assumir o trono de ferro e liderar os reinos, algumas das importantes famílias na série são: Stark (Lobos), Lannister (Leões), Targaryen (Dragões), Baratheon (Veados), Greyjoy (Lulas), Tully (Peixes), Frey (Duas Torres), Arryn (Águias), Martell (Sol) e Tyrrel (Flores)**.**

Para assumir o trono de ferro e governar os sete reinos, as casas entram em guerra constantemente com o objetivo de impor sua força.

Em meio a tanta guerra, sua equipe, neutra nas batalhas, foi convocada para desenvolver um mecanismo que ajude a controlar e medir qual foi a casa com maior número de sobreviventes e qual foi a casa derrotada. Para facilitar a interpretação dos resultados, foi proposto identificar os resultados através de uma matriz onde as linhas representarão a casa desafiante e as colunas a casa desafiada. O conteúdo da matriz representa a quantidade de sobreviventes da casa desafiante. Com diversas batalhas ocorrendo, a casa vencedora é a com o maior número de sobreviventes em uma batalha e a segunda casa mais forte será a que conseguiu deixar a casa vencedora com o menor número de sobreviventes.

Entrada

A entrada é constituída de vários casos de teste. Cada caso possui um número inteiro (0 <= K <= 10) que indica a quantidade de casas que irão travar uma batalha, em seguida serão informados uma lista de nomes das casas e por fim a uma matriz indicando o número de sobreviventes das batalhas (0 <= S <= 20000). O programa deverá ser encerrado quando informado uma quantidade de casas menor que 3.

Saída

A saída será composta pelos números 1 e 2 seguido pelo nome da primeira e da segunda casa mais fortes entre as batalhas.

Exemplos de testes

Entrada	**Saída**
3 Stark Lannister Greyjoy 0 30 80 20 0 100 50 60 0 0	1 - Lannister 2 - Stark

PROBLEMA H: PERSISTÊNCIA

Arquivo: `persiste.[cpp/c/java]`

Cor: vermelho

Limite de tempo: 1s

Descrição do problema

Persistência de um número é o número de passos necessários para reduzi-lo a um único dígito multiplicando todos os seus algarismos para obter um segundo número, depois multiplicando todos os dígitos deste número para se obter um terceiro número, e assim por diante, até que um número de um dígito é obtido. Por exemplo, 77 tem uma persistência de quatro, porque requer quatro etapas para reduzi-lo a um dígito: 77, 49, 36, 18, 8. Números com apenas um dígito, tem persistência 0, pois não é preciso fazer nenhuma multiplicação para reduzi-los.

Você deve fazer um programa que, dado um número inteiro, ele deve dizer qual o menor número com aquela persistência. Por exemplo, menor número de persistência 1 é 10, o menor de persistência 2 é 25 e o menor de persistência 3 é 39.

Entrada

A entrada é composta de vários casos de teste. Cada caso é composto de um número inteiro de 1 dígito (0 a 9) indicando a persistência desejada. Os casos de testes terminam com o fim de arquivo (EOF).

Saída

Para cada entrada, a saída deve apresentar um número inteiro não negativo representando o menor valor com aquela persistência.

Exemplos de testes

Entrada	**Saída**
3 2 1	39 25 10

PROBLEMA I: ESTRELAS

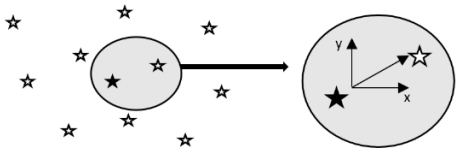
Arquivo: `estrelas.[cpp/c/java]`

Cor: amarelo

Limite de tempo: 1s

Descrição do problema

A BRUM agencia do Uzbequistão identificou por meio de ondas de rádio que existem portais para outros planetas dentro de algumas estrelas chamadas supernovas, mas estão com dificuldade de dado uma posição em uma dimensão X saber qual portal está mais acessível baseando-se na menor distância. Você foi contratado para resolver esse problema, mas não se preocupe se você encontrar nomes de planetas repetidos, afinal o espaço ainda está sendo catalogado pela BRUM.



Entrada

A entrada é constituída de vários casos de teste. Cada caso de testes possui a quantidade de estrelas ($1 \leq Q \leq 1000$) e a dimensão em que elas estão ($3 \leq D \leq 10$). Esta entrada é seguida por Q linhas com D inteiros cada (100 <= P <= 1000) e do nome do destino (com no máximo 200 caracteres). Em seguida são realizadas várias consultas. Cada consulta é representada por D inteiros (100 <= R <= 1000). A última consulta é representada por D inteiros iguais a 0 e não deve ser processada; encerrando o caso de teste. O programa encerra com uma entrada Q = D = 0.

Saída

Para cada consulta, seu programa deve imprimir o planeta mais próximo às coordenadas passadas.

Exemplos de testes

Entrada	**Saída**
2 3 -2 -5 -7 Mercurio 22 15 2 Marte -2 -5 -1 21 10 -1 0 0 0 4 4 -2 -5 -7 3 Venus 52 5 2 -2 Jupiter -2 5 17 3 Saturno 51 25 23 -12 Urano 6 7 -1 -4 10 -5 20 0 56 11 10 -1 50 20 23 -10 0 0 0 0 0 0	Mercurio Marte Venus Saturno Jupiter Urano

PROBLEMA J: POKER

Arquivo: `poker.[cpp/c/java]`

Cor: branco

Limite de tempo: 1s

Descrição do problema

Quem não quer ficar rico fazendo o que gosta? Quem não gostaria de ter a oportunidade de ganhar uma boa grana em apenas um final de semana? Pois é, com certeza muitos gostariam de uma oportunidade como essa, e muitos veem nos campeonatos de Poker esta oportunidade. Porém não é um jogo fácil já que envolve diversos aspectos, tanto estatísticos quanto psicológicos.

Hoje em dia o Poker se tornou muito acessível e existem diversas casas de Poker online nas quais é possível jogar com dinheiro real. Uma empresa decidiu abrir sua própria casa de poker online e precisa da sua ajuda para validar os vencedores de uma partida. Esta casa de Poker irá trabalhar com o Texas Hold'em, jogo no qual cada jogador recebe 2 cartas e existem 5 cartas na mesa, sendo o objetivo encontrar o melhor jogo de 5 cartas entre as 7 disponíveis.

O Poker é jogado utilizando um baralho convencional de 52 cartas, sendo 13 valores distintos (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K) e 4 naipes (Copas, Ouros, Paus e Espadas). A tabela abaixo apresenta todos os possíveis jogos, na ordem em que são avaliados durante uma partida (do melhor para o pior jogo, assim como seus critérios de desempate).

Jogo	**Regra**	**Critério de Desempate**
Straight Flush	Sequência de 5 cartas do mesmo naipe podendo iniciar em A (A, 2, 3, 4, 5) e terminar também em A (10, J, Q, K, A)	A maior carta da sequência desempata dois jogadores
Quadra	Quatro cartas com valores iguais + a maior carta que não pertença a quadra	A maior quadra ou em caso de continuar empatado, a maior carta restante
Full House	Uma trinca + uma dupla	A maior trinca ou em caso de continuar empatado, a maior carta dupla
Flush	5 cartas de um mesmo naipe	A maior carta do jogo, em caso de empate segue verificando até que todas as 5 cartas sejam validadas
Straight	Sequência de 5 cartas de naipes independente do naipe podendo iniciar em A (A, 2, 3, 4, 5) e terminar também em A (10, J, Q, K, A)	A maior carta da sequência
Trinca	3 cartas iguais (naipes diferentes) + 2 maiores cartas que não pertençam a trinca	A maior trinca ou em caso de empate verifica a maior carta e a segunda maior carta
Duas Duplas	Duas duplas + a maior carta que não pertença a uma das duplas	A maior dupla, a segunda maior dupla ou a maior carta
Dupla	2 cartas iguais (naipes diferentes) + 3 maiores cartas que não pertençam a dupla	A maior dupla ou a maior carta, segunda maior carta e a terceira maior carta
Maior Carta	5 maiores cartas do jogo	A maior carta do jogo, em caso de empate segue verificando até que todas as 5 cartas sejam validadas

Entrada

A entrada é constituída de vários casos de teste. Cada caso de testes inicia com o número de jogadores (0 <= N <= 23), seguido por 5 cartas separadas por espaço em branco. Após isto são recebidas duas cartas para cada um dos N jogadores. Os naipes são representados pela primeira letra de seu nome. O programa encerra com uma entrada N = 0.

Saída

Para caso de teste, imprima o número do jogador vencedor, ou empate quando houver empate.

Exemplos de testes

Entrada	**Saída**
2 10O 5P 7E 6O 9C AO 2P AC AE 2 10O 5P 7E 6O 9C AO AP AC AE 2 10O 10P 10C QP JP AC KE AP 9C 0	Jogador 2 Empate Jogador 1