Desafio de Programadores

25 de setembro de 2015

Este caderno contém 10 problemas em páginas numeradas de 1 a 16.

Verifique se o caderno está completo.

Informações Gerais

Sua solução deve ser chamada de *letra_do_problema.c*, *letra_do_problema.cpp* ou *letra_do_problema.java*, onde *letra_do_problema* é a letra maiúscula que identifica o problema. Lembre que em Java o nome da classe principal deve ser igual ao nome do arquivo. Exceto quando explicitamente indicado, as seguintes condições valem para todos os problemas:

Entrada

- A entrada deve ser lida da entrada padrão.
- A entrada pode ser composta por um ou mais casos de testes. Cada caso de teste é descrito usando um número de linhas que depende do problema.
- Quando uma linha de dados contém vários valores, estes são separados por um único espaço. Não aparecem outros espaços na entrada. Não existem linhas vazias.
- Cada linha, incluindo a última, possui um indicador de fim de linha.
- O fim da entrada é especificado no problema, podendo ser indicado por uma linha contendo valores específicos. Neste caso, esta linha não deve ser processada como um caso de teste.

Saída

- A saída deve ser escrita na saída padrão.
- O resultado de cada teste deve aparecer na saída usando um número de linhas que depende do problema.
- Quando uma linha de saída contém vários valores, estes devem ser separados por um único espaço. Não podem aparecer outros espaços na saída. Não devem existir linhas vazias.
- Cada linha, incluindo a última, deve possuir um indicador de fim de linha.
- O formato da saída deve seguir rigorosamente o padrão descrito no problema.

Agradecimentos

Altamir G. Bispo Jr., Delano M. Beder, Mario A. S. Liziér, Paulo R. Cerioni e Tiago B. Reis

PROBLEMA A

Brincando com Strings

Nome do arquivo: A.c, A.cpp ou A.java

Pedro é uma criança que gosta muito de brincar com strings. Sempre que ele vê uma string ***s***, ele escreve todas as suas subsequências, isto é, todas as strings que podem ser obtidas a partir de ***s*** apagando algum número de caracteres (possivelmente nenhum), sem mudar a ordem entre os que sobram. Às vezes, Pedro fica particularmente empolgado e escreve também todas as subsequências para todas as permutações da string que lhe é dada.

Agora, Pedro acaba de ver duas strings, ***s*** e ***t***. Em uma folha de papel ele escreve todas as subsequências de **s** e, em outra, ele escreve todas as subsequências de todas as permutações de **t**. Ele quer saber qual é o tamanho da maior subsequência que está escrita em ambas as folhas e, como ele escreveu muitas subsequências, ele pede a sua ajuda.

Entrada

A entrada consiste de duas linhas. A primeira contém a string ***s*** e, a segunda, a string ***t***. Todos os caracteres de ambas as strings são letras minúsculas e $1 \leq |s|, |t| \leq 100.000$.

Saída

A saída deve ser um único inteiro representando o tamanho da maior subsequência de ***s*** que também é subsequência de alguma permutação de ***t***.

Exemplos

Entrada:	**Saída:**
abcd ad	2

Entrada:	**Saída:**
ufscar desafiodeprogramadores	4

PROBLEMA B

Mais um Jogo de Tabuleiro

Nome do arquivo: B.c, B.cpp ou B.java

Paulo e Bernardo criaram um jogo e agora passam horas disputando um contra o outro. Esse jogo é jogado em um tabuleiro retangular com N linhas e M colunas, em que cada linha tem uma peça branca à direita de uma peça preta e os jogadores alternam turnos para mover suas peças. Bernardo controla as peças brancas e sempre começa o jogo, enquanto Paulo é o que controla as peças pretas. A cada turno, o jogador deve escolher uma das peças que ele controla e movê-la para uma outra posição na mesma linha, de forma que a peça branca continue à direita da peça preta. Caso seja o turno de um certo jogador e ele não possa fazer nenhum movimento válido, ele perde o jogo e o outro jogador ganha.

Depois de muito praticar, ambos os garotos aprenderam a jogar de forma ótima, eles não cometem nenhum erro. Sabendo disso, você deve escrever um programa que leia a configuração inicial de um tabuleiro e determine quem é o vencedor daquela partida.

Na figura abaixo, por exemplo, temos um tabuleiro em que N=1 e M=6. Como Bernardo começa, uma jogada ótima para ele seria mover a peça branca duas casas para a esquerda. Assim, Paulo só tem a opção de mover sua peça uma casa para a esquerda e, no próximo turno, Bernardo também move a sua uma casa para a esquerda, não deixando nenhum movimento válido para Paulo e, portanto, vencendo o jogo.

Entrada

A primeira linha da entrada contém dois inteiros N e M ($1 \leq N \leq 1.000$ e $2 \leq M \leq 1.000$) representando as dimensões do tabuleiro. A i-ésima das N linhas seguintes contém dois inteiros p_i e b_i ($1 \leq p_i < b_i \leq$ M), indicando que na linha i do tabuleiro a peça preta está na coluna p_i, contada da direita para a esquerda a partir de um, e a peça branca está na coluna b_i, contada da mesma forma.

Saída

Seu programa deve imprimir “Bernardo”, caso Bernardo seja o vencedor para a configuração inicial dada, “Paulo”, caso Paulo seja o vencedor, ou “Empate”, caso a instância do jogo nunca termine, sempre assumindo que ambos os jogadores jogam de maneira ótima.

Exemplos

Entrada:	**Saída:**
1 6 2 5	Bernardo

Entrada:	**Saída:**
2 3 2 3 2 3	Paulo

PROBLEMA C

Eleições

Nome do arquivo: C.c, C.cpp ou C.java

Numerolândia é um pequeno país famoso pela paixão nacional: matemática. Todas as leis desse país são baseadas em alguma propriedade matemática. Nesse contexto, as eleições nesse pequeno país se aproximam e o TSE (Tribunal Superior Eleitoral) desse país determinou que:

- Cada número de candidato é composto por ***D*** dígitos decimais;
- Cada número de candidato é não-decrescente. Um número é não-decrescente se cada dígito (exceto o primeiro dígito) é maior ou igual ao dígito anterior. Por exemplo: 223, 4455567 e 899 são números não-decrescentes.

Você é o encarregado por, dado o número de dígitos possíveis, calcular a quantidade de candidatos possíveis.

Entrada

A primeira linha de entrada contém um inteiro positivo N ($0 < N \leq 10000$) indicando o número de casos de teste. A segunda linha contém o inteiro D ($0 < D \leq 200$) indicando a quantidade de dígitos decimais dos números de candidatos.

Saída

Para cada caso de teste, a saída consiste em uma linha indicando a quantidade de candidatos possíveis.

Exemplos

Entrada:	**Saída:**
5 1 2 3 4 5	10 55 220 715 2002

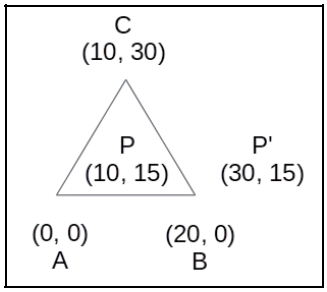
PROBLEMA D

O Jogo Geométrico de Pedro

Nome do arquivo: D.c, D.cpp ou D.java

Pedro simplesmente adora matemática, em especial geometria. Seu maior passatempo é ficar inventando jogos ou atividades que o envolvam para brincar com seus amiguinhos. Agora, Pedro inventou um pequeno passatempo que envolve 4 pontos no plano cartesiano: dados 3 pontos *A*(*Ax,Ay*), *B*(*Bx, By*) e *C*(*Cx,Cy*) que representam os vértices de um triângulo, verifique que um dado ponto *P*(*Px, Py*) encontra-se ou não dentro do triângulo *ABC*. Na figura a seguir ilustra-se que:

- O ponto *P*(10,15) encontra-se dentro do triângulo *ABC* formado pelos pontos *A*(0,0), *B*(20,0) e *C*(10,30).
- O ponto *P'*(30,15) não se encontra dentro do triângulo *ABC* formado pelos pontos *A*(0,0), *B*(20,0) e *C*(10,30).



Entrada

A primeira linha de entrada contém um inteiro positivo N ($0 < N \leq 10^6$) indicando o número de casos de teste. As demais N linhas representam os casos de teste. Cada caso de teste é uma sequência de 8 inteiros *Ax, Ay, Bx, By, Cx, Cy, Px* e *Py* ($-10^9 < Ax, Ay, Bx, By, Cx, Cy, Px, Py \leq 10^9$) que representam os 4 pontos (*A, B, C* e *P*) no plano cartesiano.

Saída

Para cada caso de teste, deve ser impressa uma linha indicando se o ponto *P* encontra-se ('S') ou não ('N') dentro do triângulo *ABC*.

Exemplos

Entrada:	**Saída:**
2 0 0 20 0 10 30 10 15 0 0 20 0 10 30 30 15	S N

PROBLEMA E

Algodão-doce

Nome do arquivo: E.c, E.cpp ou E.java

Flávia é uma vendedora de algodão-doce. Ela repôs o estoque de açúcar ***D*** dias atrás, comprando ***Q*** quilos deste produto e pagando um custo ***C*** por quilo.

Uma unidade de algodão-doce é preparada através do processo a seguir: gira-se uma vareta sobre a máquina de algodão-doce durante um período ***S*** de tempo e, consequentemente, uma quantidade ***A*** de gramas de açúcar se enrola na vareta a cada segundo. Cada unidade produzida é vendida por um preço ***P*** e Flávia somente produz as unidades de algodão-doce que vende.

Como a empresária eficiente que é, Flávia deseja saber o quanto lucrou com as vendas (isto é, o valor do dinheiro ganho com as vendas menos o valor gasto com a compra do açúcar) e, além disso, ela deseja saber quanto açúcar resta no estoque passados os ***D*** dias.

Entrada

A primeira linha de um caso de teste contém, nesta ordem, ***S*** ($1 \leq S \leq 100$), a quantidade média de segundos passados durante o preparo de uma unidade do algodão doce na máquina, ***A*** ($1 \leq A \leq 100$), a quantidade média de gramas de açúcar que é adicionada a cada segundo durante o preparo, ***P*** ($1 \leq P \leq 1000$), o preço de cada algodão doce, ***Q*** ($1 \leq Q \leq 1000$), o número de quilos de açúcar adquiridos, ***C*** ($1 \leq C \leq 1000$), o custo do quilo de açúcar e ***D*** ($1 \leq D \leq 1000$), o número de dias durante os quais Flávia saiu à rua com o objetivo de realizar suas vendas.

A linha a seguir contém as quantidades ***Di*** de algodão-doce que foram vendidas durante os ***D*** dias ($0 \leq Di \leq 100$).

Saída

Imprima duas linhas, a primeira no formato “SALDO: **x**”, em que **x** representa o valor do dinheiro ganho com as vendas menos o valor gasto com a compra do açúcar e, a segunda, no formato “ACUCAR RESTANTE: **y** QUILOS”, em que **y** representa a quantidade de açúcar restante no estoque, em quilos. Ambos são valores inteiros.

Exemplo

Entrada:	**Saída:**
4 2 10 300 5 4 50 46 38 42	SALDO: 260 ACUCAR RESTANTE: 298 QUILOS

PROBLEMA F

Transportando Caramelos

Nome do arquivo: F.c, F.cpp ou F.java

A União Federativa dos Sublimes Caramelos (UFSCar) é um pequeno país famoso pelos deliciosos caramelos que produz. Recentemente, entretanto, o sistema viário da União vem passando por certas mudanças que têm dificultado a distribuição dos caramelos para todas as cidades do país.

A UFSCar possui ***N*** cidades, identificadas por números de 1 a ***N***, ligadas por ***N***-1 estradas. Antigamente, todas as estradas eram bidirecionais, e era possível viajar de uma dada cidade para qualquer outra somente usando essas estradas. Agora, entretanto, visando deixar o tráfego pelas vias da UFSCar mais seguro, todas as estradas passaram a ser unidirecionais, isto é, elas só podem ser trafegadas em um sentido.

Mesmo com essas recentes mudanças, o governo da UFSCar deseja instalar um Depósito de Caramelos (DC) em alguma cidade do país. Da cidade onde o DC for instalado, deve ser possível alcançar qualquer outra cidade do país, apenas viajando pelas estradas na direção permitida, para que todos os habitantes tenham acesso fácil aos saborosos caramelos. Para que esse critério seja obedecido, talvez algumas vias terão seu sentido invertido. Como o governo da UFSCar não quer fazer mais tantas mudanças, ele pede que você determine em qual cidade o DC deve ser instalado a fim de minimizar o número de inversões de sentidos nas vias. Caso mais de uma cidade obedeça a esse critério, o DC deve ser instalado naquela que tem menor identificador.

Entrada

A primeira linha da entrada contém o inteiro ***N*** ($1 \leq N \leq 100.000$), indicando o número de cidades da UFSCar. As ***N***-1 linhas seguintes descrevem as estradas da UFSCar, uma estrada por linha. Cada estrada é descrita por um par de inteiros a_i, b_i ($1 \leq a_i, b_i \leq N$) indicando que existe uma via que vai da cidade a_i à cidade b_i.

Saída

A saída deve consistir de uma única linha contendo um número inteiro, o índice da cidade que deve ser escolhida para instalação do DC.

Exemplos

Entrada:	**Saída:**
2 2 1	2

Entrada:	**Saída:**
3 1 2 2 3	1

Entrada:	**Saída:**
4 1 4 2 4 3 4	1

PROBLEMA G

Programa de Auditório

Nome do arquivo: G.c, G.cpp ou G.java

Você foi selecionado para participar de um programa de auditório e quer aproveitar a oportunidade para ganhar muitos prêmios em barras de ouro, que valem mais do que dinheiro. Para isso, você quer conseguir o máximo de pontos possível nas provas do programa.

A você serão oferecidas N provas, das quais você deve escolher exatamente K para participar. A cada prova é associado um valor e, se você ganha a prova, esse valor é somado ao seu total de pontos. Se você perde a prova, nenhum valor é somado e você continua com os mesmos pontos que tinha antes. Conhecendo essas regras, você deve escrever um programa que calcule o máximo de pontos possível que você pode somar em todas as provas.

Entrada

A entrada consiste de duas linhas. A primeira linha contém dois inteiros, N e K ($1 \leq K \leq N \leq 100.000$). A segunda linha contém N inteiros v_i ($-10.000 \leq v_i \leq 10.000$) indicando o valor da prova i.

Saída

Imprima uma única linha contendo um único inteiro representando a soma máxima de pontos que você pode conseguir no programa.

Exemplos

Entrada:	**Saída:**
3 3 10 20 30	60

Entrada:	**Saída:**
4 3 40 30 20 10	90

Entrada:	**Saída:**
4 3 -10 -20 -30 40	40

Entrada:	**Saída:**
5 2 -1 -1 -1 -1 -1	0

PROBLEMA H

Relação entre Apostas

Nome do arquivo: H.c, H.cpp ou H.java

Márcio gosta de apostar na loteria. Ele também gosta de ajuntar seus volantes antigos de apostas e guardá-los no porão. Em um certo dia durante as férias, Márcio desceu ao porão com a ideia de separar os volantes, mantendo agrupados apenas os volantes cujas apostas estão relacionadas entre si.

Uma aposta é uma lista de números mutuamente distintos. Um grupo de apostas relacionadas entre si satisfaz a seguinte definição: dentro do grupo, uma aposta *i* compartilha pelo menos um número com uma aposta *j*, para todo *i*, *j*. Márcio agora deseja saber qual é o valor de percentagem de apostas que um determinado grupo representa, em comparação ao grupo de apostas de maior tamanho possível, sendo que ambos grupos satisfazem a definição acima.

Por exemplo: o volante apresenta 6 números mutuamente distintos e 2 desses números são marcados em cada aposta do mesmo grupo. O grupo de apostas relacionadas entre si de Márcio é composto por 5 apostas. O grupo de maior tamanho é composto também por 5 apostas: (1,2), (1,3), (1,4), (1,5) e (1,6). Não há outro grupo de tamanho maior do que 5 no qual todas as suas apostas estejam relacionadas entre si. Portanto, a quantidade de apostas no grupo de Márcio corresponde a 100% da quantidade de apostas do maior grupo possível.

Considere que nenhuma aposta seja exatamente igual a alguma outra dentro do mesmo grupo. Considere também que os números selecionáveis nas apostas são distintos entre si e que são exatamente os mesmos para cada volante de apostas.

Entrada

A entrada é composta por uma linha com três valores inteiros: ***N***, a quantidade de números distintos entre si em cada volante de apostas ($1 \leq N \leq 50$), ***R***, a quantidade de números que foram selecionados em cada aposta ($1 \leq R \leq N$) e ***A***, a quantidade de apostas que estão relacionadas entre si ($1 \leq A \leq C(N,R)$, em que $C(N,R)$ significa o número de combinações possíveis de ***R*** números selecionados dentre ***N*** alternativas).

Saída

Uma linha contendo a correspondência, em termos de percentagem, da quantidade de apostas do grupo de Márcio *versus* a quantidade de apostas do maior grupo possível, para cada tripla (***N***,***R***,***A***), com arredondamento e duas casas decimais, terminando com o sinal de percentagem (%). Veja o exemplo de saída.

Exemplos

Entrada:	**Saída:**
12 3 15	27.27%

Entrada:	**Saída:**
6 2 5	100.00%

Entrada:	**Saída:**
50 25 6571092338254	10.40%

PROBLEMA I

Cupons

Nome do arquivo: I.c, I.cpp ou I.java

Todo mundo adora comprar com cupons. Muitas pessoas procuram por cupons de desconto quando resolvem comprar um determinado produto. Outras pessoas fazem compras apenas por possuírem cupons de desconto. Existem pessoas, que por não terem um cupom, tentam códigos aleatórios, na esperança de obter descontos.

Uma famosa loja da internet, adepta ao fornecimento de cupons, gera um lote de cupons no início de cada ciclo de promoções, com validade limitada cada um. Em cada compra, a loja precisa conferir se o código fornecido é válido ou não. Um código de cupom é considerado válido se ele constar na relação de cupons já gerados pela loja e se o código ainda não foi utilizado mais do que ***K*** vezes. O código de cada cupom é único e é formado apenas por letras (maiúsculas e/ou minúsculas), sem espaços.

A sua tarefa é a de levantar o número de códigos inválidos informados durante um ciclo de promoção. Para isso, irão te fornecer a listagem de todos os cupons gerados, incluindo o número máximo de vezes que cada cupom pode ser utilizado, e todos os códigos informados durante as compras.

Entrada

As entradas são formada por exatamente três linhas. A primeira linha de entrada contém dois números inteiros ***C*** ($1 \leq C \leq 100.000$) e ***T*** ($1 \leq T \leq 100.000$), indicando o número de cupons gerados pela loja e o número de cupons informados nas compras, respectivamente. Na segunda linha de entrada estão ***C*** pares, um para cada cupom gerado. Cada par é formado pelo código do cupom ($1 \leq |código| \leq 20$) e o número máximo ***K*** ($1 \leq K \leq 100$) de vezes que o código pode ser utilizado. Na terceira linha de entrada estão os ***T*** códigos de cupons informados durante as compras.

Saída

Seu programa deve produzir uma única linha, contendo o número total de tentativas inválidas de uso de cupons.

Exemplos

Entrada:	**Saída:**
3 5 ABC 2 DEF 2 GH 2 ABC HG ABC DEF ABC	2

Entrada:	**Saída:**
2 4 ABC 1 DEF 2 AB ABC DEFG ABC	3

Entrada:	**Saída:**
1 2 ASrfGTVDSXgyT 2 ASrfGTVDSXgyT ASrfGTVDSXgyT	0

PROBLEMA J

Encontro Interplanetário

Nome do arquivo: J.c, J.cpp ou J.java

Andromeda já foi uma galáxia outrora muito próspera e de avançada tecnologia, porém atualmente se encontra em um período de completa decadência, restando poucos recursos disponíveis em seus planetas.

Visto a gravidade da situação e a falta de alternativas, os representantes de cada planeta de Andromeda combinaram de se reunir em um dos planetas para propor medidas que possam reverter esse cenário e determinar o futuro da galáxia.

Cada planeta da galáxia dispõe de uma única máquina de teletransporte capaz de instantaneamente transportar um único representante por vez, de um dado planeta para outro. No entanto, as máquinas estão desgastadas e foi constatado que só poderão ser usadas por um determinado número de vezes para teletransportar um representante para fora do planeta. Para receber um representante vindo de outro planeta, o processo é mais simples e não desgasta a máquina do planeta receptor. Além disso, as máquinas possuem diferentes alcances máximos para o teletransporte, ou seja, é possível transportar um representante do planeta A para o planeta B, somente se a distância entre esses planetas for igual ou inferior ao alcance da máquina localizada em A. Nenhum planeta dispõe de algum outro meio de transporte.

Os líderes dessa galáxia logo notaram que a escolha do planeta que sediaria o encontro poderia impactar diretamente no número de representantes que participariam do evento e, consequentemente, nas suas chances de sucesso. Sendo assim, eles gostariam de saber quais planetas dessa galáxia deveriam ser escolhidos como sede do evento para que o número de representantes reunidos em um deles seja o maior possível.

Vale ressaltar que é possível que um representante inicialmente em um dado planeta faça múltiplas viagens por outros planetas a fim de alcançar o seu objetivo final.

Entrada

A primeira linha de entrada contém o inteiro P ($1 \leq P \leq 100$), representando o número de planetas dessa galáxia. As P linhas subsequentes fazem referência ao planeta de identificador i ($0 \leq i \leq P-1$) e cada uma contém os valores x_i, y_i, z_i, r_i, a_i e n_i, separados por espaço, onde:

- x_i, y_i e z_i ($-10000 \leq x_i, y_i, z_i \leq 10000$) são inteiros que denotam a coordenada desse planeta no espaço. Por simplificação, você deve considerar neste problema que os planetas são pontos;
- r_i ($0 \leq r_i \leq 10$) é um inteiro que denota o número de representantes inicialmente nesse planeta, dispostos a comparecer ao evento;
- a_i ($0 \leq a_i \leq 100000$) é um ponto flutuante com até duas casas decimais de precisão que denota o alcance máximo da máquina de teletransporte localizada nesse planeta;
- n_i ($1 \leq n_i \leq 200$) é um inteiro que denota o número de vezes que a máquina de teletransporte desse planeta pode ser utilizada para transportar um representante para fora dele.

Saída

A saída consiste em uma única linha contendo o número máximo de representantes que poderão comparecer à reunião após as viagens necessárias, seguido por um espaço e uma lista de identificadores, também separados por espaços, de todos os planetas capazes de sediar esse evento com tal quantidade de participantes. Os identificadores dos planetas devem aparecer em ordem crescente. Não imprima um espaço no final da linha.

Exemplos

Entrada:	**Saída:**
3 2 0 1 3 0.92 1 -3 4 -3 2 10.11 2 2 -1 0 4 14.73 2	7 0

Entrada:	**Saída:**
5 1 -3 0 0 0.79 2 2 -4 -2 2 1.8 3 4 3 1 0 1.95 3 2 -2 -4 2 0.92 1 2 3 1 1 2.22 3	2 1 3