III Maratona Interna de Programação

3 de julho de 2016

Caderno de Problemas

Este caderno contém 8 problemas; as páginas estão numeradas de 1 a 11, não contando esta página de rosto. Verifique se o caderno está completo.

A) Sobre os nomes dos programas

1. Sua solução deve ser chamada *codigo_de_problema.c*, *codigo_de_problema.cpp* ou *codigo_de_problema.java*, onde codigo de problema é a letra maíuscula que identifica o problema. Lembre que em Java o nome da classe principal deve ser igual ao nome do arquivo.

B) Sobre a entrada

1. A entrada de seu programa deve ser lida da entrada padrão.
2. A entrada é composta de um único ou vários casos de teste, descrito em um número de linhas que depende do problema.
3. Quando uma linha da entrada contém vários valores, estes são separados por um único espaço em branco; a entrada não contém nenhum outro espaço em branco.
4. Cada linha, incluindo a última, contém exatamente um caractere final-de-linha.
5. O final da entrada coincide com o final do arquivo.

C) 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 saída não deve conter nenhum outro espaço em branco.
3. Cada linha, incluindo a última, deve conter exatamente um caractere final-de-linha.

Tabela 1: *Limites de Tempo*

Problema	Nome	C/C++	Java
A	Zerinho ou Um	1s	1s
B	Fórmula 1	1s	1s
C	Handebol	1s	1s
D	Apagando e Ganhando	1s	1s
E	Vá com Calma	3s	3s
F	Triângulos	1s	1s
G	Esquerda, Volver!	1s	1s
H	Remendo	3s	3s

Problema A
Zerinho ou Um

Arquivo: `zerinho.[c|cpp|java]`

Todos devem conhecer o jogo Zerinho ou Um (em algumas regiões também conhecido como Dois ou Um), utilizado para determinar um ganhador entre três ou mais jogadores. Para quem não conhece, o jogo funciona da seguinte maneira. Cada jogador escolhe um valor entre zero ou um; a um comando (geralmente um dos competidores anuncia em voz alta "Zerinho ou... Um!"), todos os participantes mostram o valor escolhido, utilizando uma das mãos: se o valor escolhido foi um, o competidor mostra o dedo indicador estendido; se o valor escolhido foi zero, mostra a mão com todos os dedos fechados. O ganhador é aquele que tiver escolhido um valor diferente de todos os outros; se não há um jogador com valor diferente de todos os outros (por exemplo todos os jogadores escolhem zero, ou um grupo de jogadores escolhe zero e outro grupo escolhe um), não há ganhador.

Alice, Beto e Clara são grandes amigos e jogam Zerinho a toda hora: para determinar quem vai comprar a pipoca durante a sessão de cinema, quem vai entrar na piscina primeiro, etc. Jogam tanto que resolveram fazem um plugin no Facebook para jogar Zerinho. Como não sabem programar, dividiram as tarefas entre amigos que sabem, inclusive você. Dados os três valores escolhidos por Alice, Beto e Clara, cada valor zero ou um, escreva um programa que determina se há um ganhador, e nesse caso determina quem é o ganhador.

Entrada

A entrada é composta de uma única linha, que contém três inteiros A, B e C, indicando respectivamente os valores escolhidos por Alice, Beto e Clara.

Saída

Seu programa deve produzir uma única linha, contendo um único caractere. Se o vencedor é Alice o caractere deve ser 'A', se o vencedor é Beto o caractere deve ser 'B', se o vencedor é Clara o caractere deve ser 'C' e se não há vencedor o caractere deve ser '*' (asterisco).

Restrições

A,B,C $\in \{0, 1\}$

Exemplo de entrada	Exemplo de saída
1 1 0	C
0 0 0	*
1 0 0	A

Problema B
Fórmula 1

Arquivo: `formula.[c|cpp|java]`

A temporada de Fórmula 1 consiste de uma série de corridas, conhecidas como Grandes Prêmios, organizados pela Federação Internacional de Automobilismo (FIA). Os resultados de cada Grande Prêmio são combinados para determinar o Campeonato Mundial de Pilotos. Mais especificamente, a cada Grande Prêmio são distribuídos pontos para os pilotos, dependendo da classificação na corrida. Ao final da temporada, o piloto que tiver somado o maior número de pontos é declarado Campeão Mundial de Pilotos.

Os organizadores da Fórmula 1 mudam constantemente as regras da competição, com o objetivo de dar mais emoção às disputas. Uma regra modificada para a temporada de 2010 foi justamente a distribuição de pontos em cada Grande Prêmio. Desde 2003 a regra de pontuação premiava os oito primeiros colocados, obedecendo a seguinte tabela:

Colocação	1	2	3	4	5	6	7	8
Pontos	10	8	6	5	4	3	2	1

Ou seja, o piloto vencedor ganhava 10 pontos, o segundo colocado ganhava 8 pontos, e assim por diante.

Na temporada de 2010, os dez primeiros colocados receberão pontos obedecendo a seguinte tabela:

Colocação	1	2	3	4	5	6	7	8	9	10
Pontos	25	18	15	12	10	8	6	4	2	1

A mudança no sistema de pontuação provocou muita especulação sobre qual teria sido o efeito nos Campeonatos Mundiais passados se a nova pontuação tivesse sido utilizada nas temporadas anteriores. Por exemplo, teria Lewis Hamilton sido campeão em 2008, já que a diferença de sua pontuação total para Felipe Massa foi de apenas um ponto? Para acabar com as especulações, a FIA contratou você para escrever um programa que, dados os resultados de cada corrida de uma temporada determine Campeão Mundial de Pilotos para sistemas de pontuações diferentes.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém dois números inteiros G e P separados por um espaço em branco, indicando respectivamente o número de Grandes Prêmios ($1 \leq G \leq 100$) e o número de pilotos ($1 \leq P \leq 100$). Os pilotos são identificados por inteiros de 1 a P. Cada uma das G linhas seguintes indica o resultado de uma corrida, e contém P inteiros separados por espaços em branco. Em cada linha, o i-ésimo número indica a ordem de chegada do pilodo i na corrida (o primeiro número indica a ordem de chegada do piloto 1 naquela corrida, o segundo número indica a ordem de chegada do piloto 2 na corrida, e assim por diante). A linha seguinte contém um único número inteiro S indicando o número de sistemas de pontuação ($1 \leq \mathrm{S} \leq 10$), e após, cada uma das S linhas seguintes contém a descrição de um sistema de pontuação. A descrição de um sistema de pontuação inicia com um inteiro $K(1 \leq K \leq P)$, indicando a última ordem de chegada que receberá pontos, seguido de um espaço em branco, seguido de K inteiros $k_0, k_1, ..., k_{n-1}(1 \leq k_i \leq 100)$ separados por espaços em branco, indicando os pontos a serem atribuídos (o primeiro inteiro indica os pontos do primeiro colocado, o segundo inteiro indica os pontos do segundo colocado, e assim por diante).

O último caso de teste é seguido por uma linha que contém apenas dois números zero separados por um espaço em branco.

Saída

Para cada caso de sistema de pontuação da entrada seu programa deve imprimir uma linha, que deve conter o identificador do Campeão Mundial de Pilotos. Se houver mais de um Campeão Mundial Pilotos (ou seja, se houver empate), a linha deve conter todos os Campeões Mundiais de Pilotos, em ordem crescente de identificador, separados por um espaço em branco.

Exemplo de entrada	**Exemplo de saída**
1 3	3
3 2 1	3
3	1 2 3
3 5 3 2	3
3 5 3 1	3
3 1 1 1	2 4
3 10	4
1 2 3 4 5 6 7 8 9 10	
10 1 2 3 4 5 6 7 8 9	
9 10 1 2 3 4 5 6 7 8	
2	
5 5 4 3 2 1	
3 10 5 1	
2 4	
1 3 4 2	
4 1 3 2	
2	
3 3 2 1	
3 5 4 2	
0 0	

Problema C
Handebol

Arquivo: `handebol.[c|cpp|java]`

Frustrado e desanimado com os resultados de sua equipe de futebol, o Super Brasileiro Clube (SBC) resolveu investir na equipe de handebol. Para melhor avaliar os atletas, os técnicos identificaram que seria útil analisar a regularidade dos jogadores. Especificamente, eles estão interessados em saber quantos jogadores fizeram gols em todas as partidas.

Como o volume de dados é muito grande, eles gostariam de ter um programa de computador para realizar essa contagem.

Entrada

A primeira linha da entrada contém dois inteiros N e M ($1 \leq N \leq 100$ e $1 \leq M \leq 100$), indicando respectivamente o número de jogadores e o número de partidas. Cada uma das N linhas seguintes descreve o desempenho de um jogador: a i-ésima linha contém M inteiros X_j ($0 \leq X_j \leq 100$, para $1 \leq j \leq M$), informando o número de gols do i-ésimo jogador em cada partida.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número de jogadores que fizeram gols em todas as partidas.

Exemplo de entrada	**Exemplo de saída**
5 3 0 0 0 1 0 5 0 0 0 0 1 2 1 1 0	0

Problema D
Apagando e Ganhando

Arquivo: `apagando.[c|cpp|java]`

Juliano é fã do programa de auditório Apagando e Ganhando, um programa no qual os participantes são selecionados através de um sorteio e recebem prêmios em dinheiro por participarem.

No programa, o apresentador escreve um número de N dígitos em uma lousa. O participante então deve apagar exatamente D dígitos do número que está na lousa; o número formado pelos dígitos que restaram é então o prêmio do participante.

Juliano finalmente foi selecionado para participar do programa, e pediu que você escrevesse um programa que, dados o número que o apresentador escreveu na lousa, e quantos dígitos Juliano tem que apagar, determina o valor do maior prêmio que Juliano pode ganhar.

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém dois inteiros N e D ($1 \leq D < N \leq 10^5$), indicando a quantidade de dígitos do número que o apresentador escreveu na lousa e quantos dígitos devem ser apagados. A linha seguinte contém o número escrito pelo apresentador, que não contém zeros à esquerda.

O final da entrada é indicado por uma linha que contém apenas dois zeros, separados por um espaço em branco.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha na saída, contendo o maior prêmio que Juliano pode ganhar.

Exemplo de entrada	Exemplo de saída
4 2 3759 6 3 123123 7 4 1000000 7 3 1001234 6 2 103759 0 0	79 323 100 1234 3759

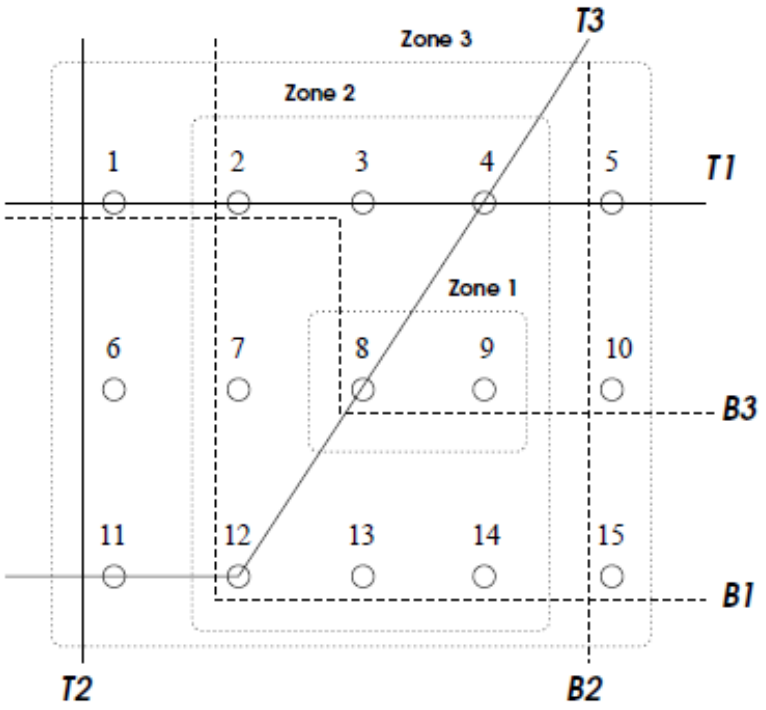
Problema E
Vá com Calma

Arquivo: `calma.[c|cpp|java]`

O prefeito de uma cidade pretende introduzir um novo sistema de transporte para simplificar a vida de seus habitantes. Isso será feito através da utilização de um cartão de débito, que o prefeito nomeou "GoEasy". Há dois meios de transporte na cidade: trens e ônibus. O sistema de trem é "baseado em zonas", enquanto o sistema de ônibus é "baseado em jornadas". A tarifa para a viagem é calculada como segue:

- Primeiramete há uma taxa de duas unidades monetárias para entrar no sistema de transporte, independentemente do meio inicial de transporte.
- Quando viajar de trem um cliente paga quatro unidades monetárias para cada mudança de zona.
- Ao viajar de ônibus a cliente paga uma unidade monetária a cada vez que ele/ela embarca num ônibus.

Um mapa do sistema de transporte irá proporcionar informações sobre as estações pertencentes a cada zona, e a seqüência de estações para cada itinerário de ônibus e trem. Ônibus e trens se movem em ambas as direções em cada itinerário, e nenhum trem ou ônibus passa pela mesma estação duas vezes durante uma única viagem através de um itinerário. É sempre possível ir de qualquer estação a qualquer outra estação usando trens e/ou ônibus. As regras para as tarifas de computação são rígidas: se durante uma viagem de trem um cliente entra em uma determinada zona duas vezes, ele/ela é cobrado(a) duas vezes, da mesma forma, se durante uma viagem de ônibus o cliente utiliza duas vezes o ônibus para o mesmo itinerário, ele/ela é cobrado(a) duas vezes.



No mapa do transporte acima um cliente pode viajar da estação 2 para a estação 4 pagando apenas duas unidades monetárias, usando a linha T1, uma vez que elas estão na mesma zona.

Mas se o cliente precisa ir da estação 2 à 5, então o melhor é tomar o ônibus B3 para a estação 10 e, em seguida, tomar o ônibus B2 para a estação 5, pagando um total de quatro unidades monetárias. Ao invés de rastrear toda a viagem de cada passageiro, a idéia do prefeito é que máquinas sejam colocadas em todas as estações, e os viajantes devem passar seu cartão pessoal GoEasy apenas no começo e término de toda a viagem. Uma vez que todas as máquinas são interligadas em rede, com base na saída e entrada do sistema de estações, pode-se calcular o custo mínimo possível para a viagem, e qual o valor será cobrado do cartão de débito do viajante. Tudo o que falta é um sistema de computador para fazer os cálculos para a tarifa a ser deduzida. Assim, dado o mapa do sistema de transporte na cidade, você deve escrever um programa para calcular a tarifa mínima que o cliente deve pagar para viajar entre duas paradas/estações dadas.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém dois inteiros Z e S, que indicam, respectivamente, o número de zonas ($1 \leq Z \leq 30$) e o número de estações de trem/ônibus na cidade ($1 \leq S \leq 100$). Cada estação tem um único número de identificação variando de 1 a S, e cada estação pertence exatamente a uma zona. Cada uma das seguintes Z linhas descreve as centrais pertencentes a uma zona. A descrição de uma zona começa com um K inteiro que indica o número de estações ($1 \leq K \leq S$) na zona, seguido de K inteiros representando as estações na zona. Depois disso vem uma linha com dois números inteiros T e B, representando, respectivamente, o número de itinerários de trem ($1 \leq T \leq 50$) e o número de itinerários de ônibus ($1 \leq B \leq 50$). Em seguida, vem T linhas descrevendo itinerários de trem, seguido por B linhas descrevendo itinerários de ônibus. A descrição de cada itinerário é composto de uma linha contendo L um inteiro que indica o número de estações ($2 \leq L \leq S$) no itinerário, seguido por L inteiros especificando a seqüência de estações no itinerário. Finalmente, vem uma linha com dois inteiros X e Y ($1 \leq X \leq S, 1 \leq Y \leq S e X \neq Y$), especificando que o cliente viajou da estação X para a estação Y. O final da entrada é indicado por $Z = S = 0$.

Saída

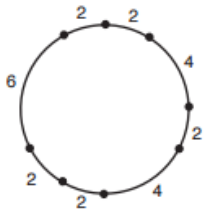
Para cada caso de teste seu programa deve imprimir uma linha, contendo um inteiro representando o valor a ser deduzido do cartão GoEasy do viajante.

Exemplo de entrada	**Exemplo de saída**
3 15 2 8 9 7 2 3 4 7 12 13 14 6 1 5 6 10 11 15 3 3 5 1 2 3 4 5 3 1 6 11 4 4 8 12 11 6 2 7 12 13 14 15 3 5 10 15 6 1 2 3 8 9 10 11 6 3 15 2 8 9 7 2 3 4 7 12 13 14 6 1 5 6 10 11 15 3 3 5 1 2 3 4 5 3 1 6 11 4 4 8 12 11 6 2 7 12 13 14 15 3 5 10 15 6 1 2 3 8 9 10 11 5 0 0	2 4

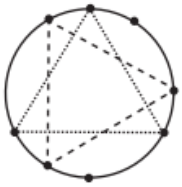
Problema F
Triângulos

Arquivo: `triangulos.[c|cpp|java]`

São dados N pontos em uma circunferência. Você deve escrever um programa que determine quantos triângulos equiláteros distintos podem ser construídos usando esses pontos como vértices. A figura abaixo ilustra um exemplo; (a) mostra um conjunto de pontos, determinados pelos comprimentos dos arcos de circunferência que têm pontos adjacentes como extremos, e (b) mostra os dois triângulos que podem ser construídos com esses pontos.



(a)



(b)

Entrada

A primeira linha da entrada contém um número inteiro N, o número de pontos dados. A segunda linha contém N inteiros X_i, representando os comprimentos dos arcos entre dois pontos consecutivos na circunferência: para $1 \leq i \leq (N-1), X_i$ representa o comprimento do arco entre os pontos i e $i+1$; X_N representa o comprimento do arco entre os pontos N e 1.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número de triângulos equiláteros distintos que podem ser construídos utilizando os pontos dados como vértices.

Restrições

- $3 \leq N \leq 10^5$;
- $1 \leq X_i \leq 10^3$, para $1 \leq i \leq N$.

Exemplos

Exemplo de entrada	Exemplo de saída
8 4 2 4 2 2 6 2 2	2
6 3 4 2 1 5 3	1

Problema G
Esquerda, Volver!

Arquivo: `esquerda.[c|cpp|java]`

Este ano o sargento está tendo mais trabalho do que de costume para treinar os recrutas. Um deles é muito atrapalhado, e de vez em quando faz tudo errado – por exemplo, ao invés de virar à direita quando comandado, vira à esquerda, causando grande confusão no batalhão.

O sargento tem fama de durão e não vai deixar o recruta em paz enquanto este não aprender a executar corretamente os comandos. No sábado à tarde, enquanto todos os outros recrutas estão de folga, ele obrigou o recruta a fazer um treinamento extra. Com o recruta marchando parado no mesmo lugar, o sargento emitiu uma série de comandos "esquerda volver!" e "direita volver!". A cada comando, o recruta deve girar sobre o mesmo ponto e dar um quarto de volta na direção correspondente ao comando. Por exemplo, se o recruta está inicialmente com o rosto voltado para a direção norte, após um comando de "esquerda volver!" ele deve ficar com o rosto voltado para a direção oeste. Se o recruta está inicialmente com o rosto voltado para o leste, após um comando "direita, volver!" ele deve ter o rosto voltado para o sul.

No entanto, durante o treinamento, em que o recruta tinha inicialmente o rosto voltado para o norte, o sargento emitiu uma série tão extensa de comandos, e tão rapidamente, que até ele ficou confuso, e não sabe mais para qual direção o recruta deve ter seu rosto voltado após executar todos os comandos. Você pode ajudar o sargento?

Entrada

A primeira linha da entrada contém um inteiro N que indica o número de comandos emitidos pelo sargento ($1 \leq N \leq 1.000$). A segunda linha contém N caracteres, descrevendo a série de comandos emitidos pelo sargento. Cada comando é representado por uma letra: 'E' (para "esquerda, volver!") e 'D' (para "direita, volver!").

Saída

Seu programa deve produzir uma única linha, indicando a direção para a qual o recruta deve ter sua face voltada após executar a série de comandos, considerando que no início o recruta tem a face voltada para o norte. A linha deve conter uma letra entre 'N', 'L', 'S' e 'O', representando respectivamente as direções norte, leste, sul e oeste.

Exemplo de entrada	**Exemplo de saída**
3 DDE	L
2 EE	S

Problema H
Remendo

Arquivo: `remendo.[c|cpp|java]`

Carlão é muito preocupado com o meio ambiente. Sempre que possível, ele tenta utilizar meios de transporte menos poluentes. Recentemente ele conseguiu um emprego próximo de casa e agora está utilizando sua bicicleta para ir ao trabalho.

Infelizmente, no caminho entre sua casa e seu emprego, há uma fábrica de pregos, que frequentemente deixa alguns pregos caírem de seus caminhões que acabam furando os pneus de da bicicleta de Carlão. Por isso, ele acaba tendo que fazer diversos remendos nos pneus de sua bicicleta.

Para fazer os consertos, Carlão usa dois tipos diferentes de remendos. Ambos os tipos têm a largura do pneu da bicicleta, mas diferem no comprimento. Como o valor do remendo é proporcional ao seu comprimento, Carlão está tentando encontrar uma maneira de economizar, gastando o menor comprimento total possível de remendos para fazer os consertos, mas sem precisar cortá-los.

O primeiro passo para efetuar o conserto é fazer uma marca com giz em uma posição do pneu e depois anotar as distâncias, medidas no sentido horário, de cada um dos furos em relação à marca de giz. Todos os furos devem ser cobertos por um remendo. Carlão gostaria de sua ajuda para determinar, a partir das posi¸cões dos furos, a forma mais econômica de efetuar o conserto.

Entrada

A entrada contém vários casos de teste. Cada caso de teste consiste de duas linhas. A primeira linha contém quatro inteiros N $(1 \leq N \leq 1000)$, C $(1 \leq C \leq 10^6), (1 \leq T1)$ e $T2$ $(T2 \leq C)$. O inteiro N corresponde ao número de furos no pneu e C corresponde ao comprimento da circunferência do pneu, em centímetros. Os comprimentos dos remendos, em centímetros, são dados pelos inteiros $T1$ e $T2$. A segunda linha da entrada contém N inteiros $F_i (0 \leq F_i \leq C-1)$, um para cada furo, descrevendo a distância no sentido horário da marca de giz até o furo i $(1 \leq i \leq N)$, em centímetros. O final da entrada é determinado por EOF (fim de arquivo).

Obs: Se a distância entre dois furos é exatamente k centímetros, um remendo de comprimento k centímetros é suficiente para cobrir ambos os furos.

Saída

Para cada caso de teste, seu programa deve imprimir uma única linha contendo um inteiro indicando o menor comprimento total de remendos que é suficiente para consertar todos os furos do pneu.

Exemplo de entrada	Exemplo de saída
5 20 2 3 2 5 8 11 15 4 20 12 9 1 2 3 13	8 12