acm International Collegiate Programming Contest 2013 IBM event sponsor

Maratona de Programação da SBC 2013

Sub-Regional Brasil do ACM ICPC

14 de Setembro de 2013

Caderno de Problemas

Informações Gerais

Este caderno contém 10 problemas; as páginas estão numeradas de 1 a 19, não contando esta página de rosto. Verifique se o caderno está completo.

A) Sobre a entrada

1) A entrada de seu programa deve ser lida da *entrada padrão*.
2) A entrada é composta de um único caso 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 o caractere final-de-linha.
5) O final da entrada coincide com o final do arquivo.

B) 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 o caractere final-de-linha.

Promoção:

SBC Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

v1.1

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

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

Exemplos

Entrada	**Saída**
`1 1 0`	`C`

Entrada	**Saída**
`0 0 0`	`*`

Entrada	**Saída**
`1 0 0`	`A`

Problema B

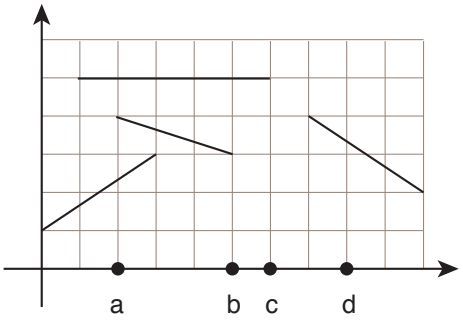
Balão

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

Uma das principais dificuldades de organizar uma Maratona de Programação é recolher os balões que escapam e ficam presos no teto do salão: muitas vezes o contrato com o dono do salão exige que este seja entregue limpo logo após o evento, sob pena de multa.

Este ano a organização da Maratona está mais previdente: ela tem o desenho do teto do salão, e quer sua ajuda para determinar o que pode acontecer com um balão, dependendo da posição no solo onde ele é solto (isto é, se é bloqueado pelo teto ou se escapa para o exterior do salão).

O teto do salão é formado por vários planos que, vistos de lado, podem ser descritos por segmentos de reta, como mostrado na figura abaixo:



O balão pode ser considerado pontual. Quando um balão toca um segmento do teto que é horizontal, ele fica preso. Quando um balão toca um segmento que é inclinado, o balão desliza até o ponto mais alto do segmento e escapa, podendo escapar completamente do salão ou podendo tocar em mais segmentos. Não há pontos em comum entre os segmentos que formam o teto.

Por exemplo, se o balão for solto nas posições marcadas como `a` ou `b`, será bloqueado na posição de coordenadas $(2, 5)$; se o balão for solto na posição marcada como `c`, será bloqueado na posição de coordenadas $(6, 5)$; e se o balão for solto na posição marcada como `d`, não será bloqueado e escapará para fora do salão na posição de coordenada $x = 7$.

Escreva um programa que, dada a descrição do teto do salão como segmentos de reta, responde a uma série de consultas sobre a posição final de balões soltos do piso do salão.

Entrada

A primeira linha da entrada contém dois inteiros N e C indicando, respectivamente, o número de segmentos de reta do teto e o número de consultas. Cada uma das N linhas seguintes contém quatro inteiros X_1, Y_1, X_2, Y_2, descrevendo um segmento de reta do perfil do teto, com extremos de coordenadas (X_1, Y_1) e (X_2, Y_2).

Cada uma das C linhas seguintes descreve uma consulta e contém um inteiro X, indicando que a consulta quer determinar o que acontece com um balão solto no ponto de coordenada $(X, 0)$.

Saída

Para cada consulta da entrada, seu programa deve imprimir uma única linha. Se o balão escapar do salão, a linha deve conter um único inteiro X, indicando a coordenada x pela qual o balão escapa do salão. Caso contrário, a linha deve conter dois inteiros X e Y indicando a posição (x, y) em que o balão fica retido no teto.

Restrições

- $1 \leq N \leq 10^5$
- $1 \leq C \leq 10^5$
- $0 \leq X_1, X_2 \leq 10^6$, $0 < Y_1, Y_2 \leq 10^6$, $X_1 \neq X_2$
- não há dois valores de coordenadas x iguais, considerando todos os segmentos.
- $0 \leq X \leq 10^6$

Exemplos

Entrada	**Saída**
4 4 0 1 3 3 1 5 6 5 5 3 2 4 7 4 10 2 2 5 8 6	2 5 2 5 7 6 5

Entrada	**Saída**
4 3 1 3 4 2 10 3 7 4 2 3 8 3 3 5 5 4 4 9 8	1 7 8 3

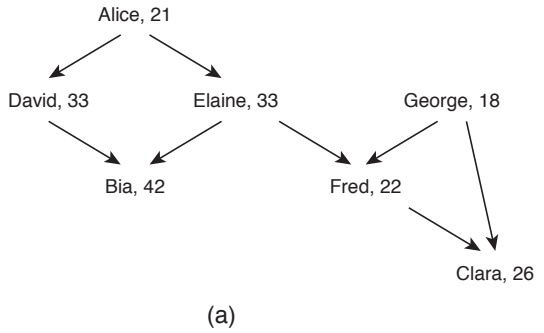
Problema C

Chefe

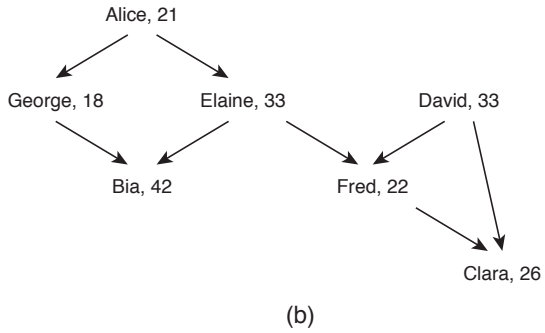
Arquivo: `chefe.[c|cpp|java]`

Todos conhecem Iks, a última moda em redes sociais, que fez tanto sucesso que competidores como Facebook e Google+ estão começando a ter dificuldades financeiras. Assim como muitas companhias ".com", Iks surgiu em uma pequena garagem, mas hoje emprega milhares de pessoas no mundo todo.

O sistema de gerência utilizado em Iks é bem diferente do padrão. Por exemplo, não há diretorias ou superintendências. No entanto, como é usual em outras companhias, há uma cadeia (ou melhor, várias cadeias) de comando: uma pessoa pode gerenciar outras pessoas, e pode ser gerenciada por outras pessoas. As figuras abaixo mostra a cadeia de comando para alguns empregados, junto com suas idades.



(a)



(b)

Uma pessoa P_1 pode gerenciar outra pessoa P_2 diretamente (quando P_1 é o superior imediato de P_2) ou indiretamente (quando P_1 gerencia diretamente uma pessoa P_3 que gerencia P_2 direta ou indiretamente). Por exemplo, na figura (a) acima, Alice gerencia David diretamente e Clara indiretamente. Uma pessoa não gerencia a si própria, nem direta nem indiretamente.

Um folclore que apareceu em Wall Street é que Iks é tão bem sucedido porque em sua rede de comando um(a) gerente é sempre mais jovem do que as pessoas que ele(a) gerencia. Como podemos ver na figura acima, isso não é verdade. Mas esse folclore incentivou Iks a desenvolver uma ferramenta para analisar o seu sistema de gerenciamento, e estudar se tem alguma influência no sucesso da empresa. Você foi contratado para trabalhar nessa ferramenta.

Dadas a descrição da cadeia de comando na Iks e as idades de seus empregados, escreva um programa que execute uma série de instruções. Instruções podem ser de dois tipos: trocas de gerência e perguntas. Uma instrução de troca de gerência faz dois empregados A e B trocarem suas posições na cadeia de comando. Como exemplo, a figura (b) acima mostra a cadeia de comando resultante quando David e George trocam suas respectivas posições na cadeia de comando. Uma instrução de pergunta identifica um empregado A e deseja saber a idade do mais jovem gerente (direto ou indireto) de A na cadeia de comando. Por exemplo, no cenário da figura (a) acima a idade do(a) gerente mais jovem de Clara é 18 anos; já no cenário da figura (b), a idade do(a) gerente mais jovem de Clara é 21 anos.

Input

A entrada é composta de várias linhas. A primeira linha contém três inteiros N, M e I, indicando respectivamente o número de empregados, o número de relações de gerência direta e o número de instruções. Empregados são identificados por números de 1 a N. A segunda linha contém N inteiros K_i, onde K_i indica a idade do empregado de número i.

Cada uma das M linhas seguintes contém dois inteiros X e Y, indicando que X gerencia Y diretamente. Seguem-se I linhas, cada uma descrevendo uma instrução. Uma instrução de troca de gerência é descrita em uma linha contendo o identificador `T` seguido de dois inteiros A e B, indicando os dois empregados que devem trocar seus lugares na cadeia de comando. Uma instrução de pergunta é descrita em uma linha contendo o identificador `P` seguido de um inteiro E , indicando um empregado. A última instrução será sempre do tipo pergunta.

Output

Para cada instrução de pergunta seu programa deve imprimir uma linha contendo um único inteiro, a idade da pessoa mais jovem que gerencia (direta ou indiretamente) o empregado nomeado na pergunta. Se o empregado nomeado não possui um gerente, imprima o caractere '`*`' (asterisco).

Restrições

- $1 \leq N \leq 500$
- $0 \leq M \leq 60 \times 10^3$
- $1 \leq I \leq 500$
- $1 \leq K_i \leq 100$, para $1 \leq i \leq N$
- $1 \leq X, Y \leq N$, $X \neq Y$
- $1 \leq A, B \leq N$
- $1 \leq E \leq N$

Exemplos

Entrada	**Saída**
7 8 9 21 33 33 18 42 22 26 1 2 1 3 2 5 3 5 3 6 4 6 4 7 6 7 P 7 T 4 2 P 7 P 5 T 1 4 P 7 T 4 7 P 2 P 6	18 21 18 18 * 26

Entrada	**Saída**
6 5 6 10 20 30 40 50 60 1 5 1 4 3 6 2 5 4 5 P 1 P 5 P 6 T 1 6 P 1 P 4	* 10 30 30 60

Problema D

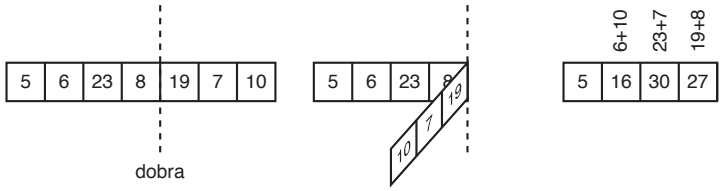
Máquina Dobradora

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

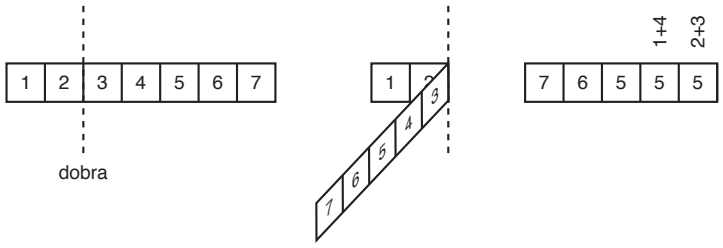
Uma das principais ferramentas de uma Máquina de Turing, que possibilita que seu poder de computação seja maior do que de outros modelos mais simples, é uma fita infinita, dividida em células, onde informações de um alfabeto ficam armazenadas.

Uma Máquina Dobradora é uma máquina inspirada na Máquina de Turing, onde a fita é finita, os dados armazenados são números inteiros e, ao invés do mecanismo de funcionamento tradicional de Turing, a máquina utiliza operações de dobras da fita para fazer computações.

Para efetuar uma dobra, a máquina escolhe uma posição entre células adjacentes e, ao realizar a dobra, ela soma os valores das células que se sobrepuseram, como pode ser visto na figura abaixo.



Observe também que a dobra pode ser feita em uma posição anterior ao centro da fita, como ilustrado a seguir. Note também que, com isso, podem ser feitas dobras também no início e no final da fita, invertendo a ordem desta.



A empresa Science of Bends Company vem desenvolvendo versões comerciais da Máquina Dobradora e a produção tem aumentado recentemente. Infelizmente o último lote de Máquinas Dobradoras produzidas está com problemas e algumas máquinas não estão funcionando corretamente. Assim, testes são necessários para evitar a venda de produtos com defeito, o que poderia denegrir a imagem da empresa.

Para testar as máquinas, um conjunto de testes é dado e, para cada fita, a máquina devolve o resultado da computação. Assim os engenheiros responsáveis pelos testes tomam nota do resultado e podem verificar se este está correto. Mas os engenheiros esqueceram-se de tomar nota de qual computação foi feita em cada conjunto de teste. Para evitar a necessidade de testar todas as máquinas novamente, os engenheiros estariam satisfeitos em descobrir se pelo menos existe uma sequência de dobras coerente para um par de fitas de entrada e saída. Para isso, eles contrataram você para desenvolver um programa que verifique, para cada fita de entrada, se existe uma sequência de dobraduras que leve a uma fita de saída.

Entrada

Cada caso de teste é composto por 4 linhas. As primeiras duas linhas referem-se à entrada fornecida à Máquina Dobradora e as duas seguintes referem-se à saída fornecida pela Máquina. A primeira

linha da entrada contém um único inteiro N, descrevendo o tamanho da fita de entrada. A linha seguinte conterá N inteiros $v_1, \ldots, v_N$, correspondentes ao conteúdo da fita de entrada. A terceira linha contém um inteiro M, o tamanho da fita de saída e a última linha conterá inteiros $w_1, \ldots, w_M$, correspondentes ao conteúdo da fita de saída.

Saída

A saída de cada caso de teste conterá uma única linha contendo a letra "S" caso exista uma sequência de dobraduras que transforme a fita de entrada na fita de saída e "N" em caso contrário.

Restrições

- $1 \leq M \leq N \leq 15$.
- $0 \leq v_i, w_j \leq 10^8$, para $1 \leq i \leq N$ e $1 \leq j \leq M$.

Exemplos

Exemplos

Entrada	**Saída**
7 5 6 23 8 19 7 10 4 5 16 30 27	S

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

Entrada	**Saída**
4 1 2 3 4 1 10	S

Entrada	**Saída**
6 19 23 3 51 2 0 2 34 64	N

Entrada	**Saída**
6 1 2 3 4 5 6 6 1 2 3 4 5 6	S

Entrada	**Saída**
6 1 2 3 4 5 6 6 6 5 4 3 2 1	S

Problema E

Mergulho

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

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 é composta de duas linhas. A primeira linha contém dois inteiros N e R, 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, 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).

Restrições

- $1 \leq R \leq N \leq 10^4$

Exemplos

Entrada	**Saída**
`5 3` `3 1 5`	`2 4`

Entrada	**Saída**
`6 6` `6 1 3 2 5 4`	`*`

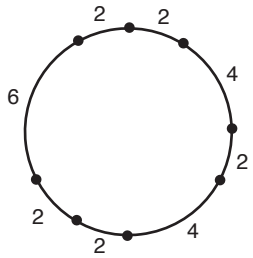
Problema F

Triângulos

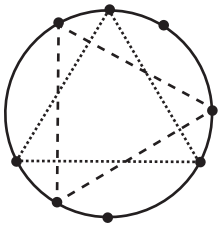
Arquivo: `triangulos.[c|cpp|java]`

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

Entrada	**Saída**
8 4 2 4 2 2 6 2 2	2

Entrada	**Saída**
6 3 4 2 1 5 3	1

Problema G

Linhas de contêineres

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

Um carregamento de Nlogs, principal produto de exportação de Nlogônia, está no porto, em contêineres, pronto para ser embarcado. Todos os contêineres têm as mesmas dimensões e são cubos. Os contêineres estão organizados no pátio do porto em L linhas e C colunas, num total de LC contêineres. Cada contêiner está marcado com um número de identificação distinto, de 1 a LC.

Cada uma das L linhas de contêineres será embarcada em um navio distinto. Para facilitar o desembarque nos divesos países em que serão entregues, os containeres de uma linha devem estar organizados de forma que os números de identificação estejam ordenados. Mais precisamente, a linha 1 foi organizada no pátio de forma a conter os contêineres identificados de 1 a C ordenados crescentemente, a linha 2 de forma a conter os contêineres de $C + 1$ a $2C$ (ordenados crescentemente), e assim por diante, até a linha L, organizada de forma a conter os contêineres de $(L - 1)C + 1$ a LC (ordenados crescentemente). A figura (a) abaixo mostra a organização de um carregamento com 5 linhas e 4 colunas de contêineres.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

(a)

13	14	15	16
5	6	7	8
9	10	11	12
1	2	3	4
17	18	19	20

(b)

13	15	14	16
5	7	6	8
9	11	10	12
1	3	2	4
17	19	18	20

(c)

O guindaste de embarque é capaz de movimentar ou uma linha completa ou uma coluna completa de contêineres, não sendo capaz de movimentar outros tipos de agrupamentos ou contêineres individuais.

Na noite anterior ao embarque, um grupo de estivadores operou os guindastes para trocar linhas e colunas do carregamento, como forma de protestar quanto aos baixos salários. A figura (b) acima mostra a configuração dos contêineres após a troca das linhas 1 e 4; a figura (c) mostra a configuração após mais uma troca, entre as colunas 2 e 3.

O carregamento precisa ser embarcado ainda hoje, mas antes disso é necessário que os contêineres sejam reorganizados da forma descrita. Você deve escrever um programa que, dada a informação sobre a posição de cada contêiner após o protesto, determine se é possível recolocar os contêineres na forma originalmente prevista utilizando apenas os guindastes, e nesse caso calcular o menor número de trocas de linhas e colunas necessário para esse fim.

Entrada

A primeira linha de um programa contém dois inteiros L e C indicando respectivamente o número de linhas e o número de colunas do carregamento. As L linhas seguintes descrevem a posição dos contêineres depois do protesto dos estivadores. Cada uma dessas L linhas contém C números inteiros $X_{l,c}$ indicando a posição de um contêiner. Cada número inteiro entre 1 e LC aparece na entrada, em alguma das L linhas. É garantido que cada número na configuração apareça uma única vez cada e que todos os números entre 1 e LC aparecerão na mesma.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número mínimo de trocas de linhas e colunas que devem ser realizadas pelo guindaste para recolocar os contêineres na posição original. Se não for possível colocar os contêineres na posição original, utilizando apenas trocas de linhas e colunas, imprima o caractere '*'.

Restrições

- $1 \leq L \leq 300$
- $1 \leq C \leq 300$
- $1 \leq X_{l,c} \leq LC$

Exemplos

Entrada	**Saída**
2 2 3 4 1 2	1

Entrada	**Saída**
3 3 9 2 4 5 8 7 6 1 3	*

Entrada	**Saída**
5 4 13 15 14 16 5 7 6 8 9 11 10 12 1 3 2 4 17 19 18 20	2

Problema H
Ônibus

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

Competições de programação normalmente exigem infraestrutura e organização por parte dos responsáveis. Um problema que frequentemente deve ser resolvido é em relação ao transporte. Ao participar de uma competição recente, Ricardinho ficou observando os ônibus e micro-ônibus utilizados no transporte dos competidores, todos enfileirados um atrás do outro enquanto os competidores desembarcavam. Os veículos eram todos de uma mesma empresa, embora tivessem pinturas distintas. Ricardinho começou a se perguntar de quantas maneiras aquela fila poderia ser formada, usando ônibus e micro-ônibus daquela empresa.

Cada ônibus tem 10 metros de comprimento. Já os micro-ônibus possuem 5 metros de comprimento. A partir de um dado comprimento total a ser alcançado com ônibus e micro-ônibus enfileirados, e das quantidades de cores diferentes para ônibus e micro-ônibus, Ricardinho quer saber de quantas formas uma fila pode ser formada.

Entrada

A entrada é composta por apenas uma linha, contendo três inteiros N, K e L, separados por espaço. O inteiro N representa o comprimento total, em metros, da fila que Ricardinho está considerando. K e L representam o número de cores distintas disponíveis para micro-ônibus e ônibus, respectivamente. Note que, como os inteiros N, K e L podem ser muito grandes, recomenda-se o uso de inteiros de 64 bits.

Saída

Como o número de formas diferentes de se formar a fila pode ser muito grande, Ricardinho está interessado nos últimos 6 dígitos da quantidade. Assim, seu programa deve produzir uma única linha contendo exatamente 6 dígitos, correspondentes aos últimos dígitos da solução.

Restrições

- $5 \leq N \leq 10^{15}$ e N é múltiplo de 5
- $1 \leq K \leq 10^{15}$
- $1 \leq L \leq 10^{15}$

Exemplos

Entrada	**Saída**
`25 5 5`	`006000`

Entrada	**Saída**
`5 1000 1000`	`001000`

Entrada	**Saída**
`20 17 31`	`111359`

Entrada	**Saída**
15 9 2	000765

Problema I

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 posições dos furos, a forma mais econômica de efetuar o conserto.

Entrada

A entrada consiste de duas linhas. A primeira linha contém quatro inteiros N, C, T_1 e T_2. 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 T_1 e T_2. A segunda linha da entrada contém N inteiros F_i, um para cada furo, descrevendo a distância no sentido horário da marca de giz até o furo i, em centímetros.

Saída

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.

Restrições

- $1 \leq N \leq 1000$
- $1 \leq C \leq 10^6$
- $1 \leq T_1, T_2 \leq C$
- $0 \leq F_i \leq C - 1$, $1 \leq i \leq N$
- Se a distância entre dois furos é exatamente k centímetros, um remendo de comprimento k centímetros é suficiente para cobrir ambos os furos.

Exemplos

Entrada	**Saída**
`5 20 2 3` `2 5 8 11 15`	`8`

Entrada	**Saída**
4 20 12 9 1 2 3 13	12

Problema J

Caminhão

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

A Sociedade de Balões Coloridos (SBC) é a principal fornecedora de balões para competições de programação; ela dispõe de grandes fábricas e depósitos, além de uma extensa frota de caminhões para garantir a alegria dos competidores.

Há várias sedes regionais na Nlogônia, todas as quais contrataram a SBC para o fornecimento de balões para a prova. A Nlogônia é um arquipélago ligado por várias pontes. Cada ilha do arquipélago pode conter várias sedes regionais e vários depósitos da SBC.

Ao planejar as rotas, a SBC se deparou com um problema: por razões de segurança, cada ponte da Nlogônia tem um limite máximo de peso permitido para os veículos que trafegam sobre ela. Devido ao grande número de pontes na Nlogônia, e ao elevado peso da mercadoria transportada, o diretor de operações da SBC pediu que você escrevesse um programa que determina o maior peso bruto que pode ser transportado entre os depósitos e os locais de prova.

Entrada

A primeira linha contém três inteiros N, M e S, indicando, respectivamente, o número de ilhas da Nlogônia, o número de pontes que ligam as ilhas e o número de sedes. As ilhas nlogonianas são numeradas de 1 a N.

Cada uma das M linhas seguintes descreve uma ponte. A descrição de cada ponte consiste de uma linha contendo três inteiros A, B e P, indicando as duas ilhas ligadas por aquela ponte e o peso máximo permitido naquela ponte, em toneladas.

Todas as pontes são de mão dupla; cada par de ilhas é ligado por no máximo uma ponte; é possível ir de qualquer ilha para qualquer outra ilha utilizando apenas as pontes do arquipélago (mas pode ser preciso passar por outras ilhas primeiro).

Cada uma das S linhas seguintes descreve uma sede. A descrição de cada sede consiste de uma linha contendo dois inteiros A e B, indicando, respectivamente, a ilha onde está a sede e a ilha onde está o depósito que irá fornecer os balões àquela sede.

Saída

Para cada sede, na ordem em que elas foram descritas na entrada, seu programa deve imprimir uma linha contendo um único inteiro, indicando o maior peso bruto, em toneladas, que pode ser transportado por caminhão do depósito que irá fornecer os balões até ela.

Restrições

- $2 \leq N \leq 2 \times 10^4$
- $1 \leq M \leq 10^5$
- $1 \leq S \leq 5 \times 10^4$
- $1 \leq A, B \leq N$, $A \neq B$
- $0 \leq P \leq 10^5$

Exemplos

Entrada	**Saída**
4 5 4 1 2 9 1 3 0 2 3 8 2 4 7 3 4 4 1 4 2 1 3 1 4 3	7 9 8 7

Entrada	**Saída**
4 5 2 1 2 30 2 3 20 3 4 10 4 1 40 2 4 50 1 3 1 2	20 40