PPCI 2024 - Maratona de Programação

19 de outubro de 2024

Instruções Importantes

- Use a opção **Runs** para enviar suas soluções. Os problemas podem resolvidos em qualquer ordem e linguagem (dentre C, C++ e Python, independentemente do problema);
- Suas soluções serão testadas com várias entradas, além das dadas como exemplo. Por isso, sua solução pode não ser aceita mesmo se funcionar para os exemplos dados. Certifique-se que ela funciona para todas as entradas possíveis;
- A saída gerada deve ser *exatamente* conforme especificada. Em particular, **não** imprima instruções ("digite um número", "a resposta é", etc);
- É garantido que todas as entradas usadas para teste estarão de acordo com o enunciado, não sendo necessário testar se são válidas;
- Ao enviar uma solução, o sistema irá responder uma das seguintes respostas:
  - `Not answered yet`: a solução está sendo corrigida. Aguarde um pouco e atualize a página;
  - `YES`: solução aceita. Parabéns!
  - `Wrong Answer`: a saída impressa pelo seu programa não é a saída correta esperada, para alguma entrada de teste;
  - `Presentation Error`: a saída impressa está correta, exceto por espaços em branco e/ou quebras-de-linha faltando/sobrando;
  - `Time Limit Exceeded`: o tempo de execução do seu programa ultrapassou o tempo limite estipulado para o problema (ver tabela abaixo). O tempo de execução da sua solução precisa ser menor;
  - `Runtime Error`: seu programa gerou algum erro em tempo de execução ("crashou");
  - `Compile Error`: seu programa não compila.
- Todas as linhas, tanto na entrada quanto na saída, terminam com o caractere de fim-de-linha ($\backslash n$), mesmo quando houver apenas uma única linha na entrada e/ou saída;

- Sua solução deve processar cada arquivo de entrada no tempo máximo estipulado para cada problema, dado pela seguinte tabela:

Problema	**Nome**	**Tempo Limite (segundos)**
A	Arremesso de Triângulo	1
B	Bloons	1
C	Cofre Break	1
D	Double Casting	1
E	Estresse	1
F	Futebol	1
G	Graffiti	1
H	Herança	1
I	Ímpar e Impares	1
J	Joias do Tesouro	1
K	Kleber e a Convenção	1
L	Lâmpada Maldita	1

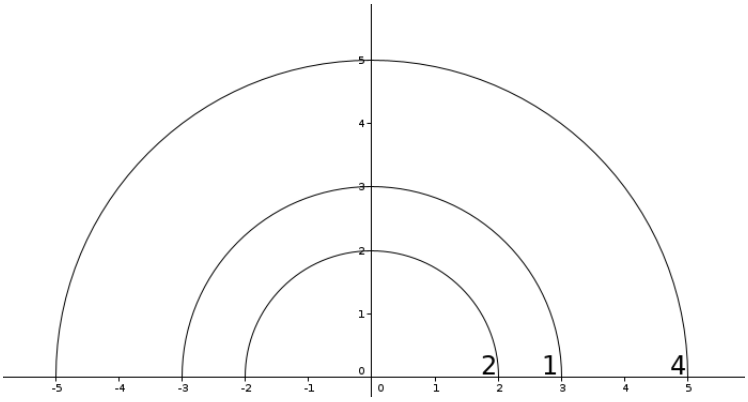
A: Arremesso de Triângulo

`Arquivo: arremesso.[c|cpp|py]`

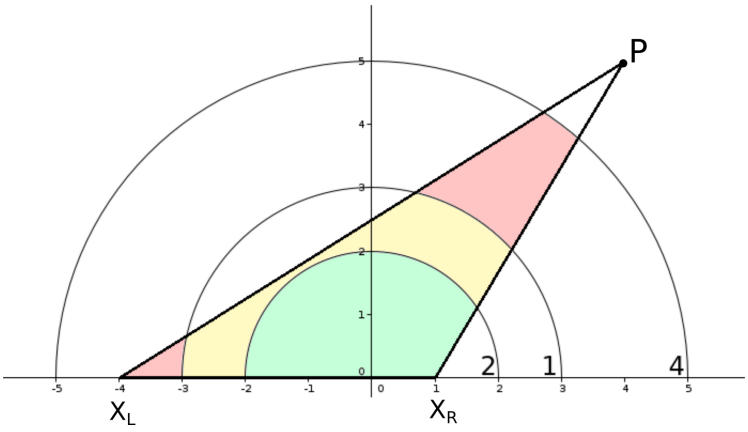
As Olimpíadas de Verão Nlogônia 2024 estão sendo um sucesso! Inúmeros atletas estão em busca da tão sonhada medalha de ouro em diversos esportes.

Um dos esportes disputados é o *arremesso de trinângulo*. Este esporte é jogado em um campo que pode ser representado em um plano cartesiano. O campo contém N semicircunferências. Seus raios são r_1, r_2, ..., r_N em ordem crescente, todas são centradas na origem, e cada semicircunferência tem seu *valor* associado.

A figura abaixo exemplifica um campo com $N = 3$ semicircunferências, de raios 2, 3 e 5, e valores 2, 1 e 4, respectivamente:



Em uma jogada, um atleta arremessa um ponto em uma posição P. Em seguida, é traçado um triângulo cujos vértices são os pontos P, $(X_L, 0)$ e $(X_R, 0)$ (os valores de X_L e X_R são definidos pelos juizes). Como exemplo, considere que um atleta arremessou um ponto na posição $P = (4, 5)$, que $X_L = -4$ e $X_R = 1$. A figura abaixo mostra o triângulo traçado:



A *região de pontuação* de uma semicircunferência é a área do seu semicírculo (no caso da primeira semicircunferência) ou a área entre uma semicircunferência e a próxima semicircunferência (no caso das demais).

A *pontuação* feita em uma semicircunferência é dada pelo produto entre seu valor e a área de intersecção entre o triângulo e sua região de pontuação. A *pontuação total* da jogada é dada pela soma da pontuação feita em todas as semicircunferências.

No exemplo dado, a pontuação total da jogada será, aproximadamente, 2×5.662733 (a área em verde na figura) $+ 1 \times 3.579953$ (a área em amarelo na figura) $+ 4 \times 2.774575$ (a área em vermelho na figura) ≈ 26.003719.

Dada a descrição do campo e da jogada, determine sua pontuação total.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 100$), o número de semicircunferencias no campo. A próxima linha contém N inteiros r_i ($1 \leq r_i \leq 100$), os raios das semicircunferências, em ordem crescente. A próxima linha contém N inteiros v_i ($0 \leq v_i \leq 10$), os valores de cada semicircunferência, na ordem em que são dados na entrada. A próxima linha contém dois inteiros X_L e X_R ($-100 \leq X_L \leq 0 \leq X_R \leq 100$) indicando as coordenadas definidas pelos juizes. A última linha contém dois inteiros X_P e Y_P ($-100 \leq X_P \leq 100$, $0 \leq Y_P \leq 100$) indicando a coordenada do ponto arremessado.

Note que tanto os pontos $(X_L, 0)$ e $(X_R, 0)$ quanto o ponto P podem ou não estar dentro de algum semi-circulo.

Saída

Imprima uma única linha contendo a pontuação total da jogada, arredondada com exatamente duas casas decimais.

Exemplo de entrada	**Exemplo de saída**
3 2 3 5 2 1 4 -4 1 4 5	26.00

Exemplo de entrada	**Exemplo de saída**
4 1 5 7 8 2 1 0 1 -10 10 0 10	55.02

B: Bloons

`Arquivo: bloons.[c|cpp|py]`

No jogo **Bloons Tower Defense**, existem macacos que defendem o território deles, que é constantemente atacado por uma sequência de balões com diferentes camadas de resistência. Cada macaco pode arremessar dardos que dão dano e destroem um número específico de camadas de um balão. Cada balão, por sua vez, tem um número de camadas a serem destruídas. Um balão só é totalmente destruído quando o último dardo que o acertar der dano **exatamente** igual à quantidade de camadas restantes do balão.

Dada uma sequência de N balões, onde cada balão tem h_i camadas (ou seja, o balão i tem resistência h_i), temos disponíveis K tipos de macacos, onde o macaco do tipo j pode arremessar um dardo que destrói exatamente d_j camadas de um balão.

Você deve determinar o número mínimo de dardos necessários para destruir completamente todos os balões da sequência. Se não for possível destruir todos os balões, retorne -1. E lembrando que um balão é totalmente destruído somente se os dano causado nele é **exatamente** o número de camadas dele. E ressaltando também que o mesmo macaco pode arremessar dardos quantas vezes quiser em qualquer balão.

Entrada

A primeira linha contém dois inteiros N ($1 \leq N \leq 1000$) e K ($1 \leq K \leq 100$), representando o número de balões e o número de macacos, respectivamente.

A segunda linha contém N inteiros, onde o i-ésimo inteiro representa h_i ($1 \leq h_i \leq 1000$), o número de camadas de resistência do i-ésimo balão.

A terceira linha contém K inteiros, onde o j-ésimo inteiro representa d_j ($1 \leq d_j \leq 1000$), o número de camadas que o dardo arremessado pelo j-ésimo macaco consegue destruir.

Saída

Imprima o número mínimo de dardos necessários para destruir todos os balões. Se não for possível destruir todos os balões, imprima -1.

Exemplo de entrada	Exemplo de saída
4 3 5 10 6 8 2 3 5	7

Exemplo de entrada	Exemplo de saída
2 2 7 11 2 5	6

Exemplo de entrada	Exemplo de saída
2 2 7 8 3 5	-1

C: Cofre Break

`Arquivo: cofre-break.[c|cpp|py]`

Carlinhos é um grande colecionador de balões e, para proteger sua preciosa coleção, decidiu trancá-la em um cofre com uma senha s, composta por n números inteiros. Porém, a memória de Carlinhos não é a das melhores, então ele decidiu escrever a senha em um papel e guardá-la em um lugar seguro. No entanto, para garantir que ninguém descubra a senha, Carlinhos resolveu anotá-la de maneira camuflada, seguindo uma regra especial:

- A maior sequência de números distintos, ou seja, a maior sequência contínua de números onde nenhum número aparece mais de uma vez, foi invertida.
- Caso haja mais de uma sequência de maior tamanho, apenas a primeira, da esquerda para a direita, será invertida.
- O restante dos números foi mantido na mesma ordem original.

Agora, anos depois, Carlinhos quer pegar seu balão vermelho favorito, mas sua memória falhou novamente. Ele não lembra qual sequência havia sido invertida, e tudo o que tem é a versão modificada da senha que ele escreveu no papel.

A sua tarefa é ajudar Carlinhos a pegar o seu balão favorito, descobrindo a senha correta do cofre.

Entrada

A primeira linha contém um inteiro n ($1 \leq n \leq 10^5$), que representa a quantidade de números da senha.

A segunda linha contém n inteiros s_i ($0 \leq s_i \leq 10^9$), que representam os dígitos da senha como foram escritos no papel.

Saída

Imprima a senha correta para o cofre, separando os números por espaços. Não imprima o espaço após o último número.

Exemplo de entrada	**Exemplo de saída**
7 1 1 2 3 4 5 5	1 5 4 3 2 1 5

Exemplo de entrada	**Exemplo de saída**
9 1 2 1 3 4 7 3 5 7	1 7 4 3 1 2 3 5 7

Exemplo de entrada	**Exemplo de saída**
6 42 42 43 43 44 44	42 43 42 43 44 44

D: Double Casting

`Arquivo: double_casting.[c|cpp|py]`

Mikio descobriu uma nova mecânica no seu jogo favorito *Lost in Time (LIT)*, o **double casting**, no qual ele pode usar duas magias ao mesmo tempo caso o dano delas seja igual.

Em *LIT* você deve comprar suas magias utilizando as LIT coins, o custo da magia é igual ao valor do dano dela em LIT coins e cada magia é de uso único. A loja de magias possui N magias.

Você pode comprar apenas um único intervalo ($1 \leq i \leq j \leq N$) de magias, que custará o valor da soma de todas as magias dentro deste intevalo, ou seja $\textbf{soma}[i; j] = (a_i + a_{i+1} + a_{i+2} + ... + a_j)$.

Mikio possui apenas K LIT coins e deseja realizar o máximo de **double casting's** possíveis.

Entrada

A primeira linha contém dois inteiro N ($1 \leq N \leq 2 \cdot 10^5$) e K ($1 \leq k \leq 10^9$), o número de magias da loja e o total de LIT coins que Mikio possui, respectivamente.

A próxima linha contém N inteiros a_i ($1 \leq a_i \leq 10^9$), o custo das magias da loja.

Saída

Imprima o maior número de **double casting's** que Mikio pode realizar.

Exemplo de entrada	**Exemplo de saída**
`5 20` `1 3 2 1 3`	`2`

Exemplo de entrada	**Exemplo de saída**
`7 26` `2 20 2 3 10 10 3`	`2`

Exemplo de entrada	**Exemplo de saída**
`4 1000` `1 100000 1 13`	`0`

E: Estresse

`Arquivo: estresse.[c|cpp|py]`

Às vezes, é preciso ter aquela conversa difícil e dizer o que é necessário. É ou não é?

Yara e sua colega Karen compartilham o mesmo apartamento. A papagaia da Karen, Lori, fica o dia todo presa na gaiola, sem ninguém para conversar, pois Karen estuda o dia inteiro e só chega de noite para dar atenção à coitada. Yara tem dó da passarinha, e precisa mostrar para Karen como a passarinha se sente abandonada e que é preciso tomar uma providência.

Yara é estudante de biologia, e por observação chegou à hipótese de que papagaios falam muito quando estressados, e às vezes também podem falar ao contrário, de trás para frente. Então, ela gravou a papagaia ao longo de um dia e obteve uma string S, que contém letras correspondentes a sons, os quais podem ter sido falas da Lori ou ruídos de fundo. A bichana sempre fala a mesma palavra P, podendo ser de frente para trás ou de trás para frente.

É preciso que você conte em quantos momentos, ao longo da gravação, houve um som que possa ter sido a Lori falando. Dessa forma, Yara poderá estudar o caso, e quem sabe mostrar para Karen como o comportamento da Lori é consequência da tristeza e estresse de estar sozinha e sem interação o dia inteiro.

Entrada

A primeira linha contém uma string S, de tamanho $|S|$ ($1 \leq |S| \leq 2 \cdot 10^5$).

A segunda linha contém uma string P, de tamanho $|P|$ ($1 \leq |P| \leq 2 \cdot 10^5$).

Ambas strings são formadas por caracteres minúsculos do alfabeto inglês (i.e. de `'a'` a `'z'`).

Saída

Imprima a quantidade de momentos distintos da gravação em que houve um som que possa ser a papagaia falando.

Exemplo de entrada	Exemplo de saída
zzzgrrauauzzz uauarrg	1

Exemplo de entrada	Exemplo de saída
ororolorolouolorololol loro	5

Exemplo de entrada	Exemplo de saída
arararara arara	3

F: Futebol

`Arquivo: futebol.[c|cpp|py]`

O futebol é o esporte mais popular do mundo, com entusiastas espalhados globalmente. Rafael, além de ser um grande fã do esporte, também é analista de desempenho de um clube de futebol profissional chamado Universitários FC.

O técnico do time solicitou que Rafael desenvolvesse um programa para ajudar o time a melhorar suas chances de marcar gols. Contudo, Rafael não é especialista em programação, e por isso ele decidiu pedir a sua ajuda para resolver este problema.

A sua tarefa é criar um programa que analise se é possível que o Universitários FC troque passes de maneira eficiente, de modo que a bola saia dos pés do goleiro e chegue até o centroavante, o jogador responsável por finalizar as jogadas.

A entrada do programa consiste nas possíveis opções de passes entre os jogadores. Cada passe indica que a bola pode ser passada de um jogador X para um jogador Y (e vice-versa). Note que as opções de passes podem ser limitadas, o programa deve encontrar o tamanho do menor caminho possível que a bola fará entre os jogadores. Em outras palavras, o programa deve mostrar a quantidade mínima de jogadores que participarão na sequência de passes, de forma que o número total de jogadores envolvidos seja o menor possível.

Entrada

A primeira linha contém um número inteiro N ($1 \leq N \leq 55$), representando o número de passes possíveis.

As próximas N linhas descrevem as opções de passe, onde a bola pode ser passada do jogador X para o jogador Y e vice-versa. Os jogadores são numerados de 1 a 11, sendo que o jogador 1 é o goleiro e o jogador 9 é o centroavante.

Saída

Imprima a quantidade mínima de jogadores que participarão da troca de passes, incluindo o goleiro (jogador 1) e o centroavante (jogador 9).

Caso não seja possível completar a sequência de passes até o centroavante, imprima -1.

Exemplo de entrada	Exemplo de saída
11 1 3 1 4 3 2 4 6 3 5 5 10 5 8 10 8 2 7 10 11 11 9	6

Exemplo de entrada	Exemplo de saída
9 1 3 1 4 3 2 4 6 4 5 5 10 5 8 11 9 9 7	-1

Exemplo de entrada	Exemplo de saída
13 1 3 1 4 3 2 4 6 3 5 4 5 5 10 5 8 10 11 8 7 1 9 11 9 7 9	2

G: Graffiti

`Arquivo: graffiti.[c|cpp|py]`

Guilherme e Gabriela são dois irmãos apaixonados por arte. Desde pequenos, sempre competiram para ver quem conseguia criar o melhor desenho. Desta vez, eles decidiram transformar o muro de casa em uma obra de arte colorida, utilizando graffiti. Cada um tem uma cor favorita: Gabriela gosta de laranja, enquanto Guilherme prefere o roxo. No entanto, essa simples discordância transformou a atividade em uma competição. Como em qualquer disputa entre irmãos, o objetivo não é apenas grafitar, mas ver quem consegue cobrir a maior área do muro com sua cor favorita.

O muro possui uma altura fixa de 1 metro e inicialmente está em branco. Gabriela e Guilherme se revezam para grafitar. Como Gabriela é mais velha, ela sempre começa. A cada turno, eles escolhem uma posição e uma distância horizontal no muro, e toda a área dentro dessa distância é grafitada.

Como ambos são extremamente competitivos, eles podem grafitar por cima das cores já preenchidas pelo outro, e a cor mais recente prevalece em cada posição.

Entrada

A primeira linha contém dois inteiros n e m ($1 \leq n \leq 10^5, 1 \leq m \leq 10^5$), representando o comprimento do muro em metros e o número de turnos, respectivamente.

As próximas m linhas descrevem as ações dos irmãos em cada turno.

Cada uma dessas linhas contém dois inteiros p_i e d_i ($0 \leq p_i < n, 1 \leq d_i \leq n - p_i$), onde p_i é a posição inicial, começando de 0, e d é a distância grafitada a partir dessa posição.

Saída

Imprima o nome do irmão que grafitou a maior área da parede. Se ambos grafitaram a mesma área, imprima "Empate".

Exemplo de entrada	Exemplo de saída
8 4 0 4 4 4 0 8 3 3	Gabriela

Exemplo de entrada	Exemplo de saída
10 7 0 10 0 8 2 7 3 4 4 2 8 2 5 1	Guilherme

Exemplo de entrada	Exemplo de saída
12 8 0 1 1 11 5 6 7 3 2 2 0 5 0 8 2 3	Empate

H: Herança

`Arquivo: heranca.[c|cpp|py]`

Você acaba de receber uma notícia inesperada: seu tio-avô, morador de Nlogônia, faleceu e deixou uma herança especial para você. Porém, sabendo da sua paixão por problemas de lógica e desafios matemáticos, ele deixou um enigma como parte de sua herança.

Na casa dele, você encontra uma mochila mágica com um limite de peso W que pode ser carregado dentro dela. Junto à mochila, há uma carta explicando que ela pode ser preenchida com uma série de itens valiosos, cada um com um peso e um valor específico. No entanto, você não precisa pegar os itens inteiros – pode escolher levar apenas uma fração de cada item, se assim desejar.

Seu desafio é simples: escrever um algoritmo que defina o valor máximo dos itens para o peso w de sua mochila.

Entrada

A primeira linha contém 2 inteiros N ($1 \leq N \leq 10^5$) e W ($1 \leq W \leq 10^3$) que indicam, respectivamente, o número de itens disponíveis e o peso máximo suportado pela mochila.

As próximas N linhas contêm 2 inteiros P ($1 \leq P \leq 10^6$) e V ($1 \leq V \leq 10^8$) indicando o peso e o valor de cada item.

Saída

Imprima um número com 2 casas após a vírgula, indicando o valor máximo.

Utilize números com precisão dupla.

Exemplo de entrada	**Exemplo de saída**
3 50 10 60 20 100 30 120	240.00

Exemplo de entrada	**Exemplo de saída**
2 50 20 10 50 3	11.80

I: Ímpar e Impares

`Arquivo: impar_e_impares.[c|cpp|py]`

Eiji tem um vetor com N números, porém ele odeia os números pares e não quer que eles estraguem seu vetor. Ele pode fazer a seguinte operação quantas vezes quiser: Ele escolhe dois números de índices i e j $(1 \leq i, j \leq N)$ e substitui o menor deles pelo valor da soma entre eles, formalmente:

$$\begin{cases} \text{Caso } a_i > a_j : a_j = a_j + a_i \\ \text{Senão } a_i = a_i + a_j \end{cases}$$

Porém ele não quer desperdiçar seu tempo realizando mais operações do que o necessário para se livrar de todos o pares.

Entrada

A primeira linha contém um inteiro N $(1 \leq N \leq 2 \cdot 10^5)$, o número de elementos do vetor

A próxima linha contém N inteiros a_i $(1 \leq a_i \leq 10^9)$, os valores de cada elemento do vetor.

Saída

Imprima o número mínimo de operações que Eiji terá que realizar para se livrar de todos os pares em seu vetor, caso não seja possível se livrar de todos os pares imprima -1.

Exemplo de entrada	**Exemplo de saída**
3 3 1 9	0

Exemplo de entrada	**Exemplo de saída**
5 3 1 2 2 10	4

Exemplo de entrada	**Exemplo de saída**
2 2 4	-1

J: Joias do Tesouro

`Arquivo: joias.[c|cpp|py]`

Você é um explorador em uma perigosa expedição nas selvas de Zathura, uma terra lendária conhecida por esconder um misterioso tesouro. Diz a lenda que esse tesouro é protegido por enigmas e armadilhas que só podem ser vencidos por aqueles que sabem o caminho mais curto até ele.

Determine o número mínimo de movimentos que você precisa para sair de sua posição inicial (sx, sy) e alcançar a posição do tesouro (tx, ty). Lembre-se de que a selva é cheia de desafios, e você só pode mover-se nas quatro direções: para cima, para baixo, para a esquerda e para a direita. Não é possível atravessar as bordas da selva.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 1000$), representando o tamanho do tabuleiro $N \times N$.

A segunda linha contém dois inteiros sx e sy ($0 \leq sx, sy < N$), que representam as coordenadas de sua posição inicial no tabuleiro.

A terceira linha contém dois inteiros tx e ty ($0 \leq tx, ty < N$), que representam as coordenadas do tesouro.

Saída

Imprima o número mínimo de movimentos que o explorador precisa fazer para alcançar o tesouro.

Exemplo de Entrada e Saída

Exemplo de entrada	**Exemplo de saída**
5 0 0 3 3	6

Exemplo de entrada	**Exemplo de saída**
7 1 1 5 5	8

K: Kleber e a Convenção

`Arquivo: kleber_e_a_convencao.[c|cpp|py]`

Kleber está participando da sua primeira convenção de colecionadores de números. Nesta convenção cada colecionador pode doar ou receber números de outros colecionadores, alterando assim sua coleção.

Há um estande de confirmação de coleção que dá uma insígnia ao colecionador se o $\oplus^*$(operação binária xor) de algum dos E números distintos presentes no estande com TODOS os N números da coleção atual de um colecionador resultarem em um número cuja representação binária contém apenas bits 1, desconsiderando zeros a esquerda. O estande dará apenas uma insígnia por coleção, independente de quantos números do estande atendam o requisito.

Kleber fará e/ou receberá Q doações de números durante a convenção e deseja saber qual o número máximo de insígnias que ele consegue obter. Ele pode ir ao estande para confirmar sua coleção quantas vezes quiser, essa verificação pode ser feita sempre que a coleção for alterada.

Veja a explicação do exemplo teste:

Suponha que Kleber tem uma coleção: $1, 2, 8$ e que o estande de confirmação tem os seguintes números: $4, 5, 7$.

Durante a convenção aconteceram as seguintes doações:

$+$ 3 deixando sua coleção como: $1, 2, 3, 8$.

$+$ 4 deixando sua coleção como: $1, 2, 3, 4, 8$.

$-$ 8 deixando sua coleção como: $1, 2, 3, 4$.

$-$ 1 deixando sua coleção como: $2, 3, 4$.

($+$ simbolizam que Kleber recebeu uma doação e $-$ simboliza que Kleber fez uma doação).

Durante esta convenção Kleber conseguiu 2 insígnias:

$1 \oplus 2 \oplus 8 \oplus \mathbf{4} = 15_{10} = 1111_2$.

$1 \oplus 2 \oplus 3 \oplus 8 \oplus \mathbf{7} = 15_{10} = 1111_2$.

* $\oplus$ (xor) é a operação binária na qual: $1 \oplus 0 = 1$, $0 \oplus 1 = 1$, $0 \oplus 0 = 0$ e $1 \oplus 1 = 0$.

Entrada

A primeira linha contém três inteiros N e E ($1 \leq N, E \leq 2 \cdot 10^5$), quantos números Kleber tem em sua coleção, quantos números distintos o estande de confirmação de coleção possui, respectivamente.

A segunda linha contém N números a_i ($0 \leq a_i \leq 10^9$), a coleção de Kleber.

A terceira linha contém E números distintos do estande de confirmação de coleção a_j ($0 \leq a_j \leq 10^9$).

A quarta linha contém um inteiro Q ($1 \leq Q \leq 2 \cdot 10^5$), quantas doações serão feitas e/ou recebidas por Kleber

As próximas Q linhas contém operações $-$ e/ou $+$ de números a_k ($0 \leq a_k \leq 10^9$), a descrição das doações que aconteceram durante a convenção.

Saída

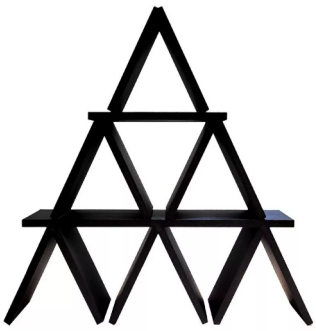
Imprima o números máximo de insígnias que Kleber conseguirá receber.

Exemplo de entrada	**Exemplo de saída**
3 3 1 2 8 4 5 7 4 + 3 + 4 - 8 - 1	2

L: Lâmpada Maldita

`Arquivo: lampada.[c|cpp|py]`

Oh Céus! No multiverso das maldições, Aladdin se perdeu no deserto e quando encontrou o gênio da lâmpada, foi amaldiçoado a montar pirâmides de cartas eternamente! Para facilitar seu trabalho, Aladdin decidiu criar um algoritmo que ajudasse-o a coletar a quantidade suficiente de cartas para uma pirâmide de altura qualquer. A pirâmide é estruturada da seguinte forma:



Entrada

A entrada possui somente uma linha e contém a altura h ($1 \leq h \leq 10^9$) da pirâmide que Aladdin precisa montar.

Saída

Imprima 1 linha com a quantidade de cartas necessárias para montar uma pirâmide de altura h.

Exemplo de entrada	**Exemplo de saída**
1	2

Exemplo de entrada	**Exemplo de saída**
10	155