VII Maratona UnBalloon de Programação

Caderno de Problemas

16 de maio de 2026

Realização

UnB

Organização

unballoon

Comissão Organizadora

Alberto Neto (UnB)
Arthur Botelho (UnB)
Daniel Porto (UnB)
Eduardo Freire (UnB)
Eduardo Quirino (UnB)
Gustavo Leal (UFG)
Pedro Gallo (UnB)
José Leite (UnB)
Lucas Sala (UnB)
Maxwell Oliveira (UnB)
Ruan Petrus (UnB)
Wilson Guimarães (UnB)

Lembretes

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova, entretanto, o mesmo não vale para materiais dispostos eletronicamente.
- A correção é automatizada, portanto, siga atentamente as exigências da tarefa quanto ao formato da entrada e saída conforme as amostras dos exemplos. Deve-se considerar entradas e saídas padrão;
- Para cada problema, além dos testes públicos, o juiz executará a sua submissão contra uma série de testes secretos para fornecer um parecer sobre a correção do programa.
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. Lembre-se que as soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas;
- Utilize a aba *clarification* para dúvidas da prova. Os juízes podem opcionalmente atendê-lo com respostas acessíveis a todos;

C/C++

- Seu programa deve retornar zero, executando, como último comando, `return 0` ou `exit 0`.

Java

- Não declare `package` no seu programa Java.
- Note que a convenção para o nome do arquivo fonte deve ser obedecida, o que significa que o nome de sua classe pública deve ser uma letra maiúscula igual a letra que identifica o problema.

Python

- Tenha cuidado ao selecionar a versão correta na submissão.

(Este caderno contém 14 problemas)

Problem A. Árvore Colorida

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Luiz, Carol e Henrique combinaram de se encontrar no parque após o encontro do clube de corrida da UnBalloon para se organizarem como time para a VII Maratona UnBalloon, a corrida mais esperada do ano. Porém, Luiz, como um exímio corredor, terminou o percurso antes do restante do time, chegando em 3° lugar do clube (atrás do Maxwell e do Arthur).

Tendo que esperar a Carol, que está mais lenta devido ao seu joelho, e o Henrique, que ainda aguarda o ônibus na parada, Luiz começou a observar as árvores do parque. Ele notou que uma árvore era bem diferente das demais, possuindo várias partes de cores distintas. Sabendo que, ao mencionar isso, seu time preferiria investigar as cores em vez de focar na próxima maratona, ajude Luiz a responder às diversas perguntas de seus colegas.

A árvore observada por Luiz pode ser modelada como um grafo bidirecional conexo e acíclico com N nós e $N-1$ arestas, onde a raiz está fixada no vértice 1. Cada nó i possui uma cor associada, representada por um valor inteiro a_i.

Sua tarefa é ajudar o time a responder Q perguntas. Cada pergunta consiste em um nó x_i, e você deve determinar a quantidade de cores distintas presentes no caminho único entre o nó x_i e a raiz da árvore.

Input

A primeira linha contém dois inteiros N e Q ($1 \leq N, Q \leq 10^5$), representando, respectivamente, a quantidade de nós na árvore e a quantidade de perguntas que Carol e Henrique farão.

A segunda linha contém N inteiros $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$), indicando a cor de cada nó da árvore.

As próximas $N-1$ linhas contêm, cada uma, dois inteiros u_i e v_i ($1 \leq u_i, v_i \leq N, u_i \neq v_i$), representando uma aresta entre os nós u_i e v_i na árvore.

Por fim, as próximas Q linhas contêm, cada uma, um inteiro x_i ($1 \leq x_i \leq N$), representando o nó de origem da i-ésima pergunta para a qual se deseja saber a quantidade de cores distintas até a raiz.

Output

A saída deve conter Q linhas. A i-ésima linha deve apresentar um único inteiro representando a quantidade de cores distintas encontradas no caminho entre o nó x_i e a raiz da árvore.

Examples

standard input	standard output
4 2 1 2 3 4 1 2 2 3 3 4 3 4	3 4
4 2 1 2 3 2 1 2 2 3 3 4 3 4	3 3

Problem B. Beats

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Nirva é um dos mais novos talentos da Unballoon. Como todos sabem, um dos principais conselhos que ouvimos no início da carreira em programação competitiva é focar em melhorar o raciocínio e deixar para aprender conteúdos mais avançados, como autômato de sufixos, fluxo, FFT etc., para depois. Como todos também sabem, quase sempre esse conselho é ignorado completamente. Por isso, Nirva decidiu aprender a "Segment Tree Beats". O próprio autor dessa estrutura de dados tem isso a dizer sobre ela:

"In China, all of the 15 candidates for the Chinese National Team are asked to write a simple research report about algorithms in informatics Olympiad, and the score will be counted in the final selection. There are many interesting ideas and algorithms in these reports. And I find that some of them are quite new for competitors in CF although they are well known in China [...].

This blog is about my report which is written about two years ago. [...] the name "Segment tree beats" is given by C_SUNSHINE which is from a famous Japanese anime Angel Beats."

Como esperado, Nirva entendeu facilmente o funcionamento da Segment Tree Beats e fez sua própria implementação. Para testar se ela está correta, ela achou que seria interessante comparar a implementação dela com a sua.

Você recebe n inteiros não negativos $a_1, \ldots, a_n$. Seu programa deve processar q operações de três tipos:

1. chmod(l, r, x) — Para todos os índices i tais que $l \leq i \leq r$, substitua o valor de a_i por $a_i \,\%\, x$.
2. chmax(l, r, x) — Para todos os índices i tais que $l \leq i \leq r$, substitua o valor de a_i por $\max(a_i, x)$.
3. soma(l, r) — Imprima a soma dos elementos a_i tais que $l \leq i \leq r$, isto é, $a_l + a_{l+1} + \cdots + a_r$.

Input

A primeira linha contém dois inteiros n e q ($1 \leq n, q \leq 1000$). A segunda linha contém n inteiros $a_1, \ldots, a_n$ ($0 \leq a_i \leq 10^6$).

Cada uma das próximas q linhas descreve uma operação em um dos formatos:

- $1\ l\ r\ x$ — representando a operação chmod(l, r, x);
- $2\ l\ r\ x$ — representando a operação chmax(l, r, x);
- $3\ l\ r$ — representando a operação soma(l, r).

É garantido que, em cada operação, $1 \leq l \leq r \leq n$ e $1 \leq x \leq 10^6$.

Output

Para cada operação do tipo 3 (soma), imprima um único inteiro representando o resultado dessa operação.

Example

standard input	standard output
6 5 1 2 3 4 5 6 3 1 5 1 2 4 3 3 1 5 2 1 4 2 3 2 3	15 9 4

Problem C. Caça ao Tesouro

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Tiagodfs, um lendário competidor de programação competitiva, decidiu esconder suas soluções mais valiosas em um terreno retangular de dimensões $b \times h$.

O esconderijo foi escolhido de forma totalmente aleatória: qualquer ponto dentro do terreno tem a mesma probabilidade de conter o tesouro.

Depois de muito investigar, a equipe *Chappell Roan, vc tá jurada aqui em São Sebastião* finalmente descobriu a existência desse esconderijo. Para tentar encontrar os códigos secretos, elas decidiram usar uma máquina de escavação especial, capaz de cavar uma região em formato de um hexágono regular de lado l. A escavação será feita de modo que o hexágono esteja completamente contido dentro do terreno.

Sua tarefa é descobrir a probabilidade da equipe *Chappell Roan, vc tá jurada aqui em São Sebastião* encontrar o tesouro fazendo uma escavação dessa forma.

Input

A entrada contém três inteiros b, h e l, tais que $1 \leq b, h, l \leq 10^3$.

É garantido que algum hexágono regular de lado l caiba completamente dentro do retângulo.

Output

Imprima a probabilidade do tesouro estar dentro da área escavada.

Sua resposta será considerada correta se o erro absoluto ou relativo não exceder 10^{-4}.

Examples

standard input	standard output
10 10 5	0.6495190528
29 23 11	0.4713151748

Problem D. Dominando Shurikens

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Na Vila Oculta dos Balões, aqueles que buscam melhorar suas habilidades estão sempre próximos do campo de treinamento. A área do campo é vasta, possuindo uma seção com N troncos enfileirados, todos com N metros de altura, numerados de 1 a N. Para usá-los como uma espécie de escala, cada tronco de número i recebeu riscos exatamente a cada i metros (até N). Por exemplo: se $N = 3$, o tronco 1 tem riscos a 1, 2 e 3 metros, o 2 tem um risco a 2 metros e o 3 tem um risco a 3 metros.

Dentre os ninjas no campo de treinamento em um certo dia, estava Lucas Sala — um especialista em shurikens, sendo capaz de atirar mais de um bilhão de shurikens simultaneamente! Ele estava buscando aperfeiçoar seu jutsu Estilo Vento: Múltiplas Shurikens Aéreas, que pode atingir vários oponentes com uma grande quantidade de shurikens. Para isso, ele estava usando os troncos do campo de treinamento da seguinte forma: ele escolhia dois inteiros l e r, com $1 \leq l \leq r \leq N$, e atirava x shurikens em cada um dos troncos de l a r (inclusive) — acertando todas, obviamente.

No entanto, a proeza de Lucas Sala não havia sido suficiente para impressionar seu sensei, José. Por isso, ele propôs um desafio para seu aluno: José escolhia inteiros l, r e k ($1 \leq l, r, k \leq N$), e, dentre os troncos de l a r, Lucas Sala deveria acertar x shurikens apenas naqueles com exatamente k riscos. Esse desafio foi feito várias vezes, com valores diferentes de l, r e k. No entanto, José não era capaz de acompanhar todas as shurikens disparadas para verificar a mira de Lucas Sala, então ele parou em alguns momentos entre os desafios para escolher outros inteiros l e r e contar as quantidades de shurikens acertadas em cada tronco de l a r.

Assim, sendo bastante cuidadoso, José anotou em seu pergaminho dois tipos de eventos relacionados ao treinamento desse dia, na ordem em que ocorreram:

1. l r: José conta a quantidade total de shurikens acertadas em todos os troncos de l a r.
2. l r k x: Lucas Sala acerta x shurikens em todos os troncos i tais que $l \leq i \leq r$ e o tronco i possui exatamente k riscos.

Esse dia foi há muitos anos, e José se lembra com felicidade daquela época. Olhando em seu pergaminho, ele percebeu que não anotou, para os eventos do tipo 1, a quantidade total de shurikens que ele contou, o que é uma pena. Ajude-o a reviver os velhos tempos e recupere essa informação para ele!

Como cada quantidade contada pode ser muito grande, você deve imprimir o resto da divisão de cada uma delas por $10^9 + 7$.

Input

A primeira linha contém dois números: N ($1 \leq N \leq 10^{12}$), o número de troncos e também a altura de cada um deles, em metros, e Q ($1 \leq Q \leq 10^5$), o número de eventos que ocorreram no treinamento.

Cada uma das próximas Q linhas contém um evento, em um dos seguintes formatos:

- 1 l r ($1 \leq l \leq r \leq N$), representando um evento no qual José contou shurikens.
- 2 l r k x ($1 \leq l \leq r \leq N, 1 \leq k \leq N, 0 \leq x < 10^9 + 7$), representando um evento em que Lucas Sala jogou shurikens.

Output

A saída deve consistir em uma linha para cada evento do primeiro tipo, cada uma contendo um único número x ($0 \leq x < 10^9 + 7$): o resto, na divisão por $10^9 + 7$, do número de shurikens contadas por José no respectivo evento.

Examples

standard input	standard output
5 4 2 1 5 1 10 2 2 4 2 5 1 1 5 1 2 3	35 15
10 7 2 1 10 2 1000000000 2 1 10 2 1000000000 1 4 5 2 1 7 1 50 1 6 8 2 8 10 5 100 1 2 2	999999979 100 0
1000000000000 2 2 1 1000000000000 2 5 1 400000000000 600000000000	999996512

Problem E. EZ

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A sociedade secreta UnBalloon tem muitos membros, muitos contatos e muitos acordos. Por causa de uma aposta perdida no pedra, papel e tesoura durante uma viagem, a UnBalloon ficou responsável por enviar, toda lua cheia, um suprimento de ratos para alimentar a cobra de estimação da bruxa da floresta.

Em determinado mês, a lua cheia caiu na quarta-feira, dia de folga do trabalho de Caleb (um dos membros da UnBalloon). Por isso, ele foi escolhido para entregar o suprimento de ratos, em vez de beber Martinis no bar como de costume. Sendo bastante metódico, Caleb escolheu levar N ratos, sendo que o i-ésimo mais pesado desses ratos pesa exatamente i quilos.

Ao entrar na cabana da bruxa para fazer a entrega, ele a ouviu falando algo sobre "...não dar asa à cobra...". Talvez por isso, assim que recebeu o suprimento de ratos, ela enfeitiçou-os da seguinte forma: "Ao comer um rato de i quilos, não será possível comer ratos que pesem $i - 1$ ou $i + 1$ quilos.".

Caleb ficou espantado com isso e pensou: "Qual será então a maior quantidade, em quilos, que a cobra poderá comer com essa restrição?". Para sua surpresa, a cobra ouviu seus pensamentos e respondeu (telepaticamente): "EZ...". Caleb sabe que EZ é uma abreviação de easy, que significa fácil em inglês. Mas será que comer a maior quantidade é tão fácil assim?

Calcule para Caleb a maior quantidade, em quilos, que a cobra poderá comer no total, considerando a restrição imposta pelo feitiço da bruxa.

Input

A entrada consiste de uma única linha contendo um único inteiro N: a quantidade de ratos ($1 \leq N \leq 1000$).

Output

Imprima, em uma única linha, um único inteiro: a maior quantidade, em quilos, que a cobra poderá comer no total.

Examples

standard input	standard output
1	1
3	4
4	6
1000	250500

Note

No primeiro caso de teste, com somente um rato, a cobra só irá comê-lo (total: 1 quilo).

No segundo caso de teste, a cobra pode comer os ratos de pesos 1 e 3, resultando em 4 quilos comidos (que é o máximo possível).

No terceiro caso de teste, a cobra pode comer os ratos de pesos 2 e 4, resultando em 6 quilos comidos (que é o máximo possível).

Problem F. Formas de Formar Times

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A organização secreta UnBalloon possui N agentes, numerados de 1 a N. Utilizando um algoritmo secreto, a organização calculou, para cada um deles, seu valor como ser humano, de modo a poder ordená-los por esse valor (nenhum agente possui o mesmo valor que o outro). A partir disso, foi criada a lista L, onde L_i é o número do i-ésimo agente mais valioso.

A organização precisa formar um time de agentes para a próxima missão, que consiste em se infiltrar na Pizza Super Ultra Secreta. Para que a missão seja bem-sucedida, os agentes escolhidos devem obedecer a uma propriedade especial. Seja S o conjunto dos números dos agentes escolhidos e P o conjunto das posições desses agentes na lista L. A partir desses conjuntos, definimos os polinômios[1] $A(x) = \sum_{i \in S} x^i$ e $B(x) = \sum_{i \in P} x^i$. Para que o time de agentes seja bem-sucedido, é necessário que S não seja um conjunto vazio e que exista um polinômio $C(x)$ com todos os seus coeficientes inteiros não negativos tal que $A(x) \cdot C(x) = B(x)$.

Como parte dos analistas da organização, você deve calcular de quantas maneiras é possível formar um time de agentes que seria bem-sucedido na missão. Como essa quantidade pode ser muito grande, calcule o resto da divisão dessa quantidade por 998244353.

[1] Um polinômio é uma função da forma $P(x) = \sum_{i=0}^{\infty} a_i \cdot x^i$ tal que o número de valores não nulos a_i é finito. Chamamos esses valores de a_i de coeficientes.

Input

A primeira linha da entrada contém o inteiro N: o número de agentes ($1 \leq N \leq 5 \cdot 10^3$).

A segunda linha da entrada contém N inteiros distintos: a lista L ($1 \leq L_i \leq N$).

Output

Imprima, em uma única linha, o resto na divisão por 998244353 da quantidade de maneiras de escolher um time de agentes para uma missão bem sucedida.

Examples

standard input	standard output
1 1	1
4 2 1 4 3	6
4 1 3 2 4	8
10 1 4 5 3 2 10 9 8 7 6	49

Problem G. Grande Maratona de Programação

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Durante a preparação para a Grande Maratona de Programação da UnBalloon, vários treinadores estão testando estratégias para montar o melhor supertime possível.

Os organizadores da prova permitem formar um supertime a partir de um subconjunto de n equipes disponíveis. Cada equipe possui quatro tipos de participantes, correspondendo a diferentes perfis de habilidades, com suas quantidades dentro da equipe i sendo representadas por quatro inteiros (a_i, b_i, c_i, d_i). Para montar um supertime, os organizadores podem escolher qualquer subconjunto dessas equipes.

Após escolher um supertime, os organizadores juntam todos os participantes das equipes escolhidas e definem:

- A, como o número total de participantes do tipo a;
- B, como o número total de participantes do tipo b;
- C, como o número total de participantes do tipo c;
- D, como o número total de participantes do tipo d.

Em seguida, é realizado um treinamento com os integrantes do supertime, com uma penalidade numérica associada. O treinamento ocorre em duas etapas independentes:

- Na primeira etapa, os participantes dos tipos a e b do supertime são pareados entre si para uma rodada de confrontos. Cada participante enfrenta exatamente um oponente, sempre que possível. Ao final, restam exatamente $|A - B|$ participantes sem oponente, que irão contribuir para o cálculo da penalidade;
- Na segunda etapa, o mesmo processo ocorre entre os participantes dos tipos c e d.

Logo depois dessas duas etapas, são utilizados dois parâmetros (k, l) para calcular a penalidade desse treinamento. Os valores k e l representam o peso de cada etapa do treinamento, e o valor da penalidade é dado pela soma das contribuições de cada etapa.

- A contribuição da primeira etapa é a quantidade de participantes restantes após os confrontos entre os tipos a e b, multiplicada por k.
- A contribuição da segunda etapa é a quantidade de participantes restantes após os confrontos entre os tipos c e d, multiplicada por l.

Os treinadores querem testar diferentes estratégias. Para isso, eles avaliarão q diferentes pares de pesos (k, l).

Sua tarefa é ajudar os treinadores: para cada par de pesos (k, l), determine qual seria a **maior penalidade** que pode ser atingida escolhendo um subconjunto das equipes.

Input

A primeira linha contém dois inteiros n e q ($1 \leq n, q \leq 10^5$) — o número de equipes e o número de consultas.

Cada uma das próximas n linhas contém quatro inteiros a_i, b_i, c_i, d_i ($0 \leq a_i, b_i, c_i, d_i \leq 10^4$), descrevendo a i-ésima equipe.

Cada uma das próximas q linhas contém dois inteiros k e l ($0 \leq k, l \leq 10^4$), representando um par de pesos.

Output

Para cada um dos q pares de pesos, imprima um inteiro: a maior penalidade possível de ser atingida.

Examples

standard input	standard output
2 3 5 3 4 1 6 1 3 4 1 1 2 0 0 3	9 14 9
5 4 5 10 0 3 9 3 2 3 7 4 2 2 2 8 5 6 4 12 9 8 1 1 3 1 9 2 7 4	22 60 177 145

Problem H. Hora da Aula

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

AnaLu passou o dia todo no LINF estudando Teoria dos Números, então decidiu botar os novos conhecimentos em prática! Por isso, ao voltar para casa, ela foi direto para seu quarto fazer questões. Então, depois de muitos ACs, ela acabou dormindo. E acabou tendo um sonho...

Neste sonho, AnaLu era novamente uma aluna do 6° ano do ensino fundamental, e a aula do dia era sobre multiplicação, exponenciação e divisibilidade. A professora escreveu no quadro as seguintes definições:

1. Considere A e X inteiros positivos.
2. $A \cdot X$: A multiplicado por X.
3. A^X: A elevado a X (que é o mesmo que fazer $A \cdot A \cdot A \cdot \ldots \cdot A$, com X fatores A).
4. $A|X$: significa que A divide X, ou seja, que existe algum inteiro positivo B tal que $A \cdot B = X$.

Mas, não satisfeita, a professora decidiu desafiar seus alunos. Ela pediu que cada um mantivesse em seu caderno o valor do número A, que inicia sendo igual a 1. A professora fará Q **queries** sobre esse número A, nas quais um número X será escrito no quadro e todos os alunos ou devem atualizar o valor de A de acordo ou então responder a uma pergunta, de acordo com o tipo da query. Há três tipos:

1. Todos os alunos devem atualizar o valor de A para $A = A \cdot X$ em seu caderno.
2. Todos os alunos devem atualizar o valor de A para $A = A^X$ em seu caderno.
3. Todos os alunos devem responder (em inglês, com "Yes" ou "No"): $X|A$?

AnaLu percebeu então que estava em um sonho, já que, dependendo das queries da professora, o número A poderia ficar tão grande que não daria pra escrevê-lo nem se juntassem todos os cadernos do universo. Mesmo assim, ela percebeu que ainda seria possível responder as queries de divisibilidade (tipo 3) da professora corretamente com a ajuda de um computador.

Mas AnaLu não trouxe nenhum computador para o sonho, então você deve ajudá-la a responder as queries!

Input

A primeira linha contém um inteiro Q ($1 \leq Q \leq 10^5$), o número de queries.

As próximas Q linhas contêm dois inteiros t ($1 \leq t \leq 3$) e X ($2 \leq X \leq 10^5$), o tipo da query e seu parâmetro. É garantido que existe ao menos uma query do tipo 3.

Output

Para cada query do tipo 3, escreva uma linha com a resposta dessa query: "Yes" caso X seja um divisor de A, ou "No", caso contrário.

Example

standard input	standard output
6 2 100000 1 3 2 3 1 2 3 12 3 18	No Yes

Note

Explicação do exemplo:

Inicialmente $A = 1$.

Após a primeira query, $A = 1^{100000} = 1$.

Após a segunda query, $A = 1 \cdot 3 = 3$.

Após a terceira query, $A = 3^3 = 27$.

Após a quarta query, $A = 27 \cdot 2 = 54$.

Para a quinta query, 12 não divide A, portanto, a resposta da query é "No".

Para a sexta query, 18 divide A, portanto, a resposta da query é "Yes".

Problem I. Inventando Comunicação

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Iasmim, Pedro e Emerson estão se preparando para apresentar um método de criptografia deles para a *International Committee for Poor Communication* (ICPC), a principal organização que estuda métodos de comunicação ruins que "funcionam".

Eles desenvolveram um método único e inovador de transmitir mensagens. Como a ideia é criar um método de comunicação ruim, eles enviam uma sequência de números criptografada. Para decriptografar e encontrar a mensagem original, deve ser encontrada a menor subsequência lexicográfica não vazia do vetor V tal que o seu hash seja estritamente menor que o hash de seu complemento. Se uma subsequência (de tamanho k) consiste das posições $p_0, p_1, \ldots, p_{k-1}$, seu complemento é a subsequência composta por todas as posições do vetor que não sejam alguma dessas k posições.

Dizemos que uma sequência A é lexicograficamente menor que uma sequência B se, na primeira posição em que elas divergem, o elemento de A for menor que o elemento de B. Caso as sequências sejam idênticas até que a menor delas termine, a sequência de menor comprimento é considerada a lexicograficamente menor.

O cálculo do hash utilizado pelo trio segue uma estrutura de hash polinomial. Dada uma subsequência (ou seu complemento) composta pelos números $x_0, x_1, \ldots, x_{k-1}$ na ordem em que aparecem no vetor V, o valor do hash H é calculado pela seguinte fórmula:

$$H = (x_0 \cdot P^N + x_1 \cdot P^{N-1} + \ldots + x_{k-1} \cdot P^{N-k+1}) \% M$$

Onde:

- x_i é o valor do i-ésimo elemento da subsequência;
- $P = 100019$ é a base polinomial;
- N é o tamanho total do vetor V;
- $M = 998244353$ é o módulo;
- O índice i (de 0 até $k-1$) representa a posição do número dentro da subsequência escolhida;
- Por convenção, o hash de uma sequência vazia ($k = 0$) é definido como 0.

Você receberá o vetor V e deve recuperar a subsequência que representa a mensagem comunicada por Iasmim, Pedro e Emerson.

Input

A primeira linha da entrada contém um único inteiro N ($1 \leq N \leq 10^6$), o tamanho do vetor V.

A segunda linha da entrada contém N inteiros V_i: os elementos do vetor V ($1 \leq V_i \leq 10^5$).

É garantido que tanto N quanto o vetor V são gerados de forma aleatória para todos os casos de teste não visíveis.

Output

Imprima, em uma única linha, a menor subsequência lexicográfica não vazia A do vetor V em que $H(A) < H(A^c)$, onde A^c é o complemento de A em V. Caso nenhuma subsequência satisfaça essa condição, imprima -1 em vez disso.

Note que devem ser impressos os valores da subsequência, e não as suas posições. Se duas ou mais subsequências lexicograficamente iguais ocorrem no vetor em posições diferentes de forma que atendem à condição, deve-se imprimir os seus valores apenas uma vez.

Examples

standard input	standard output
5 60522 14575 36426 79445 48772	14575
5 90081 33447 90629 3497 47202	3497

Note

É garantido que os casos de teste visíveis são os únicos casos de teste criados sem randomizar o tamanho N e o vetor V.

No primeiro caso de exemplo, a menor subsequência lexicográfica do vetor é [14575] e seu hash é 405057492. O seu complemento é a subsequência $[60522, 36426, 79445, 48772]$, cujo hash é 980706017. Como a menor subsequência lexicográfica já atende à condição, ela deve ser impressa.

Problem J. Jantar no RU???...

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	256 megabytes

Arthur tem aulas à noite na UnB todos os dias, de forma que ele tem que jantar antes de ir para a aula. Ele sabe que a maioria das pessoas costuma jantar no RU, e ele não tem nada contra o RU, mas ele acha que a vida é curta demais para deixar de aproveitar experiências gastronômicas diferenciadas... Porém, nem todos os amigos dele compartilham desse ponto de vista, então é necessário convencê-los.

Arthur e seus amigos já não comeram no RU no dia 1, porque "no primeiro dia tudo bem". Mas agora os amigos estão decididos a ir no RU nos próximos dias. Para impedir isso, Arthur teve uma ideia. Ele vai propor o seguinte: "podemos ir no RU todos os dias, menos os dias múltiplos de X", sendo X um inteiro positivo maior que 1. Isso parece razoável para os amigos em um primeiro momento, mas mal sabem eles que Arthur vai escolher vários valores de X para eles nunca irem no RU do dia 1 até o dia N...

Mas Arthur deve ser cuidadoso para que os amigos não percebam o que ele está fazendo! Para isso, ele deve falar a menor quantidade de valores X possível — caso contrário, seria suspeito demais e ele seria arrastado para o RU!

Ajude-o e calcule qual a menor quantidade possível de valores X que Arthur deve falar para seus amigos de forma que eles acabem não indo no RU em nenhum dos dias 1 a N.

Input

A entrada consiste em um único inteiro N: a quantidade de dias ($1 \leq N \leq 10^6$).

Output

Imprima, em uma única linha, a menor quantidade possível de valores X que Arthur deve falar para seus amigos de forma que eles acabem não indo no RU em nenhum dos dias 1 a N.

Examples

standard input	standard output
1	0
4	2
12	5
1000000	78498

Note

No primeiro caso de teste, como $N = 1$ e Arthur já não foi no RU no primeiro dia, não é necessário falar nenhum valor de X.

No segundo caso de teste, Arthur pode falar $X = 2$ e $X = 3$ para não ir em nenhum dos $N = 4$ dias (note que Arthur não irá no dia 4, porque 4 é múltiplo de 2).

Problem K. Kalculadora de Dano

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Gus e Caio Fleury são dois treinadores Pokémon experientes. Eles estão se preparando para a próxima edição da Liga Pokémon UnBalloon e, para isso, estão realizando simulações computadorizadas das batalhas.

O software de simulação escolhido por eles realiza o cálculo de dano de um único ataque feito por um Pokémon contra outro. Para isso, ele permite a escolha do tipo do Pokémon atacante, T_a, do poder do ataque utilizado, P, do tipo do Pokémon defensor, T_d, e dos pontos de vida do Pokémon defensor, H. Após esse ataque, os pontos de vida do Pokémon ficam reduzidos a $max(0, H - P \cdot M(T_a, T_d))$, onde M é a função de modificador de dano por tipo. Caso os pontos de vida sejam reduzidos a 0, o Pokémon é nocauteado.

Gus e Caio Fleury preferem usar os tipos mais clássicos: fogo (F), grama (G) e água (W). Para esses tipos específicos, a função M é definida da seguinte forma:

- Se $(T_a, T_d) = (F, G)$, (G, W) ou (W, F), então $M(T_a, T_d) = 2$.
- Caso contrário, $M = \frac{1}{2}$.

Caio Fleury estava usando extensivamente o simulador para escolher qual estratégia usar na próxima batalha. Enquanto isso, Gus estava pensando em contribuir para o simulador com uma mensagem customizada. Denotando por X os pontos de vida do Pokémon defensor após o ataque, ele pensou em imprimir "Nocaute!", caso $X = 0$, e "Sobraram X pontos de vida!", caso contrário.

Porém, Caio Fleury acabou de desafiar Gus para uma batalha, e agora não há tempo para implementar essa mensagem. Sobrou pra você implementar isso!

Input

A primeira linha da entrada contém uma única letra maiúscula, representando T_a, que pode ser apenas F, G ou W.

A segunda linha contém também uma única letra maiúscula, representando T_d, que também pode ser apenas F, G ou W.

A terceira linha contém um único inteiro, representando P ($2 \leq P \leq 10^6$). É garantido que P é par.

A quarta linha contém também um único inteiro, representando H ($1 \leq H \leq 10^6$).

Output

Imprima, em uma única linha, a mensagem proposta por Gus, de acordo com o enunciado.

Examples

standard input	standard output
G W 6 10	Nocaute!
G W 4 10	Sobraram 2 pontos de vida!
G G 2 2	Sobraram 1 pontos de vida!

Problem L. Lady Gaga e o Coeficiente Ecoante

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Lady Gaga é uma artista completa: inigualável nas performances, extremamente versátil e repleta de referências artísticas diversas. Ao longo de sua trajetória, ela se consolidou como uma das figuras mais essenciais da história da música pop e se estabeleceu como uma voz para muitos que não conseguem encontrar a sua própria. Para a felicidade dos *Little Monsters*, ela continuamente se reinventa, deixando sua legião de fãs sempre na expectativa por seus próximos passos.

Para aprimorar ainda mais suas habilidades de composição, Lady Gaga estudou uma propriedade das letras de músicas chamada Coeficiente Ecoante, que é simplesmente a soma dos valores de todos os Segmentos Ecoantes de uma letra. Um Segmento Ecoante é um trecho de uma letra que pode ser representado como um mesmo trecho repetido duas vezes, e seu valor é o produto entre a quantidade de versos que o compõem e a quantidade total de caracteres desses versos.

Como um exemplo, o seguinte trecho de Bad Romance é um Segmento Ecoante (porque pode ser representado pela repetição dos três primeiros versos):

Oh-oh-oh-oh-oh

Oh-oh-oh-oh-oh-oh-oh

Caught in a bad romance

Oh-oh-oh-oh-oh

Oh-oh-oh-oh-oh-oh-oh

Caught in a bad romance

Formalmente, a letra de uma música pode ser representada como uma lista L de versos, onde o i-ésimo verso pode ser representado como uma string L_i. Um trecho de uma letra é uma sublista contígua de L — se ele é composto por todos os versos do i-ésimo até o j-ésimo (com $j \geq i$), é denotado por L_i^j. Assim, um Segmento Ecoante de L é um trecho L_i^j tal que exista alguma lista de versos V de forma que $L_i^j = V + V$, onde $+$ representa a operação de concatenação de listas. Dessa forma, o valor de um Segmento Ecoante L_i^j é dado por $(j - i + 1) \cdot \sum_{k=i}^{j} |L_k|$, onde $|L_k|$ é o tamanho da string que representa o k-ésimo verso. Por fim, o Coeficiente Ecoante de uma letra é a soma dos valores de todos os Segmentos Ecoantes dessa letra.

Em seus estudos, Lady Gaga pretende analisar letras de músicas e seus respectivos Coeficientes Ecoantes. Para deixar esse processo mais conveniente, ela pediu à sua equipe um programa que receba uma letra de música simplificada (sem letras maiúsculas, pontuações e caracteres especiais) e imprima o Coeficiente Ecoante dessa letra.

Ajude a *Mother Monster* com um programa que realize esse cálculo de forma eficiente!

Input

A primeira linha da entrada contém um inteiro N: a quantidade de versos da letra da música ($1 \leq N \leq 5 \cdot 10^4$).

As próximas N linhas contêm, cada uma, uma string L_i, que representa o i-ésimo verso da letra ($1 \leq |L_i| \leq 25$). É garantido que L_i contém apenas caracteres minúsculos de a a z.

Output

Imprima, em uma única linha, um único valor: o Coeficiente Ecoante da letra de música representada pela entrada.

Examples

standard input	standard output
14 ohohohohoh ohohohohohohoh caughtinabadromance ohohohohoh ohohohohohohoh caughtinabadromance raraahahah romaromama gagaohlala wantyourbadromance raraahahah romaromama gagaohlala wantyourbadromance	1284
12 cantreadmycantreadmy nohecantreadmypokerface shesgotmelikenobody cantreadmycantreadmy nohecantreadmypokerface shesgotmelikenobody cantreadmycantreadmy nohecantreadmypokerface shesgotmelikenobody cantreadmycantreadmy nohecantreadmypokerface shesgotmelikenobody	8184
2 a aaa	0
4 a b a b	16

Note

Note que a definição de um Segmento Ecoante se refere à concatenação de listas, e não à concatenação de todas as strings das listas. Por isso, no terceiro caso de exemplo, não há Segmentos Ecoantes: mesmo que “a” + “aaa” = “aaaa” e que “aaaa” possa ser representada como “aa” + “aa”, a lista {“a”, “aaa”} não possui uma representação como concatenação de duas listas iguais.

Problem M. Motorista Impaciente

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2.5 seconds
Memory limit:	256 megabytes

Henrique quer estacionar seu carro no estacionamento da Universidade de Brasília (UnB).

As vagas são enumeradas de 1 até n. A vaga de número i é caracterizada por dois números: a sua distância d_i da entrada principal e a probabilidade p_i de estar ocupada. Henrique percorre as vagas sequencialmente, começando da vaga 1. Ao chegar à vaga i, duas situações podem ocorrer:

1. Com probabilidade p_i, a vaga i estará vazia. Nesse caso, Henrique tem duas opções: estacionar na vaga i (Henrique é extremamente rápido, então podemos considerar que ele leva 0 segundos para estacionar), ou seguir em frente para a próxima vaga, que será a vaga $i+1$ caso $i \neq n$ e será a vaga 1 caso $i = n$.
2. Com probabilidade $1 - p_i$, a vaga i estará ocupada. Nesse caso, Henrique é forçado a continuar para a vaga seguinte.

Cada vez que Henrique segue em frente para a próxima vaga, ele gasta 1 segundo. Após visitar a vaga i uma vez, é possível que, em visitas subsequentes, o seu estado de ocupação seja diferente, já que o estacionamento da UnB é muito movimentado e um motorista pode ter ocupado ou desocupado a vaga no tempo em que Henrique levou para voltar a visitá-la. Esse comportamento será ditado pela probabilidade p_i, que permanece fixa ao longo de todo o problema.

Henrique quer agir de tal forma que o valor esperado da soma do tempo total gasto para selecionar uma vaga com a distância da vaga em que ele finalmente estacionou seja o menor possível. Se Henrique agir de maneira ótima, qual será esse valor?

Input

A primeira linha da entrada contém um inteiro n ($2 \leq n \leq 3 \cdot 10^5$).

A segunda linha da entrada contém n inteiros $d_1, d_2, \ldots, d_n$ ($1 \leq d_i \leq 10^6$).

Por fim, a terceira linha da entrada contém n números reais $p_1, p_2, \ldots, p_n$ ($0.001 \leq p_i \leq 1$). Cada p_i é dado com no máximo 3 casas decimais.

Output

Imprima um único número real, o valor esperado da soma do tempo total gasto para selecionar uma vaga com a distância da vaga estacionada na melhor estratégia. Sua resposta será considerada correta se o erro absoluto ou relativo não exceder 10^{-6}. Ou seja, se sua resposta foi a e a resposta correta foi b, sua resposta será aceita se e somente se $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-6}$.

Examples

standard input	standard output
7 8 9 9 4 6 3 2 1 1 1 1 1 1 1	7
2 1 1000000 0.5 1	3
6 31 415 92 6535 89 79323 0.461 0.884 0.500 0.972 0.149 0.111	38.01518438177870962136

Note

No primeiro caso de teste, Henrique deve estacionar na vaga de número 4. Ele leva um total de 3 segundos para alcançá-la, e ela está a uma distância de 4 metros da entrada, logo a soma do tempo gasto com a distância é $3 + 4 = 7$. Como todas as vagas sempre estão livres, Henrique sempre consegue atingir esse valor.

Problem N. Não Ter Medo de Cair

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Luisa gosta de treinar no site UnBalloonforces, no qual faz problemas e participa de contests. Ela segue a filosofia "Não Ter Medo de Cair", que prega que deve-se fazer todo contest, mesmo que haja o risco de cair de título (equivalente à cor no Codeforces), porque essa é a melhor maneira de treinar a habilidade e o mental. Isso funcionou tão bem que hoje ela possui o título de **Super Programmer**, que é o segundo mais alto no UnBalloonforces.

No UnBalloonforces, todo problema tem um rating inteiro entre 1 e M. Como uma forma de treinamento para Luisa, seu coach, Ruan, preparou uma lista com N problemas de ratings variados. Ele também enviou uma mensagem de voz, dizendo: "Cara, na verdade, faz só os com rating maior ou igual a..." (Luisa não conseguiu ouvir o rating especificado por Ruan).

Luisa já resolveu todos os problemas, e agora ela está entediada. Lembrando da mensagem, ela se perguntou: "Se o rating escolhido por Ruan fosse i, qual seria a soma dos ratings dos problemas que eu deveria resolver?". Ajude a satisfazer a curiosidade de Luisa calculando essa soma para todo i de 1 a M!

Input

A primeira linha da entrada contém dois inteiros: N e M, a quantidade de problemas na lista de Luisa e o rating máximo de um problema no UnBalloonForces ($1 \leq N, M \leq 5 \cdot 10^5$).

A segunda linha da entrada contém N inteiros r_i: o i-ésimo deles representa o rating do i-ésimo problema da lista ($1 \leq r_i \leq M$).

Output

Imprima, em uma única linha, M inteiros: o i-ésimo deles deve ser a soma dos ratings dos problemas a serem resolvidos por Luisa caso Ruan tivesse especificado o rating i.

Examples

standard input	standard output
5 5 2 4 2 4 2	14 14 8 8 0
3 2 1 1 2	4 2
8 9 5 9 1 2 3 4 7 6	37 36 34 31 27 22 16 9 9

Note

No primeiro caso de teste, caso o rating especificado por Ruan tivesse sido 1 ou 2, Luisa precisaria resolver todos os problemas, resultando em uma soma de ratings igual a 14. Já se o rating especificado fosse 3 ou 4, Luisa não precisaria resolver os problemas de rating 2, apenas os dois de rating 4, e a soma seria apenas 8. Por fim, se o rating especificado fosse 5, Luisa não precisaria resolver nenhum problema (soma 0).