



2023
Caderno de Problemas
X Maratona Mineira de Programação

Patrocínio



<ambevtech/>



netLex

Colaboração



Realização



Informações gerais

Este caderno de tarefas é composto por 25 páginas (não contando a folha de rosto), numeradas de 1 a 25. Verifique se o caderno está completo.

Nome do programa

Cada problema tem os possíveis nomes de arquivo fonte indicados abaixo do título. Soluções na linguagem C devem ser arquivos com sufixo `.c`; soluções na linguagem C++ devem ser arquivos com sufixo `.cc` ou `.cpp`; soluções na linguagem Java devem ser arquivos com sufixo `.java` e a classe principal deve ter o mesmo nome do arquivo fonte; e soluções na linguagem Python devem ser arquivos com sufixo `.py`.

Entrada

- A entrada deve ser lida da entrada padrão.
- A entrada consiste em exatamente um caso de teste, que é descrito usando uma quantidade de linhas que depende do problema. O formato da entrada é como descrito em cada problema. A entrada não contém nenhum conteúdo extra.
- Todas as linhas da entrada, incluindo a última, terminam com o caractere de fim de linha (`\n`).
- A entrada não contém linhas vazias.
- Quando a entrada contém múltiplos valores separados por espaços, existe exatamente um espaço em branco entre dois valores consecutivos na mesma linha.

Saída

- A saída deve ser escrita na saída padrão.
- A saída deve respeitar o formato especificado no enunciado. A saída não deve conter nenhum dado extra.
- Todas as linhas da saída, incluindo a última, devem terminar com o caractere de fim de linha (`\n`).
- Quando uma linha da saída apresentar múltiplos valores separados por espaços, deve haver exatamente um espaço em branco entre dois valores consecutivos.
- Quando um valor da saída for um número real, use pelo menos o número de casas decimais correspondente à precisão requisitada no enunciado.

Problema A. Elemento X

Nome do arquivo fonte: elemento.c, elemento.cpp, ou elemento.java

Açucar, tempero, e tudo que há de bom. Esses foram os ingredientes escolhidos pelo professor P. L. Utônio para criar a sobremesa perfeita, porém ele accidentalmente adicionou um quarto ingrediente: o Elemento X! Diferentemente de um famoso desenho animado, a adição do Elemento X não deu certo, e a sobremesa ficou completamente sem gosto. Como bons viajantes do tempo que somos, nossa tarefa é impedir essa catástrofe culinária.



A organização do laboratório do professor é incrivelmente eficiente e simples: o recipiente de cada um dos quatro ingredientes (colocados em ordem - r_1, r_2, r_3, r_4) é anotado com a distância entre ele e o recipiente do Elemento X. Ao chegarmos do futuro, porém, descobrimos o motivo do acidente: o pote do Elemento X está com defeito, e mostra um valor maior que zero, o que não faz o menor sentido! Nossa tarefa é identificar qual o recipiente danificado e informar o professor antes que um desastre culinário ocorra!

Entrada

A entrada é composta de uma única linha contendo quatro inteiros r_1, r_2, r_3, r_4 , cada um representando a distância do respectivo pote para o recipiente do Elemento X.

Saída

A saída deve conter um único inteiro $1 \leq I \leq 4$, que representa o índice do pote que contém o elemento X .

Restrições

- $1 \leq r_1, r_2, r_3, r_4 \leq 10^5$

Exemplos

Entrada	Saída
1 1 2 3	1

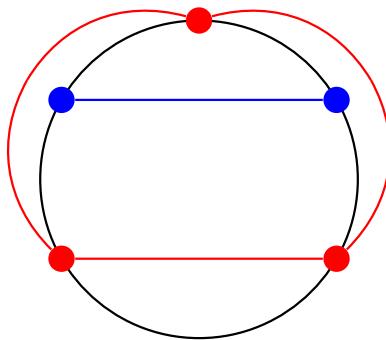
Problema B. Avenida do Contorno

Nome do arquivo fonte: `circulo.c`, `circulo.cpp`, ou `circulo.java`

A Avenida do Contorno é uma avenida circular de Belo Horizonte. Durante sua construção, foram instaladas vários estabelecimentos ao longo dela, como várias drogarias (Araújo), várias pastelarias (Rei do Pastel), etc. Por motivos logísticos, todo par de estabelecimentos do mesmo tipo deve ser conectado **diretamente** por uma rua (que pode passar por dentro ou por fora da Contorno).

Porém, para que a Contorno continue sendo a avenida principal, não podem existir esquinas em outros lugares da cidade (cruzamento de ruas). Além disso, não irão existir túneis ou viadutos na cidade.

Dados os tipos de estabelecimentos que existem em ordem ao longo da Contorno, sua tarefa é determinar se é possível construir as ruas conectando todo par de estabelecimentos de mesmo tipo, sem criar esquinas (cruzamentos de ruas) extras.



Possível configuração para o primeiro exemplo.

Entrada

A primeira linha contém um inteiro $1 \leq N \leq 10^3$. A segunda linha contém N inteiros $1 \leq c_i \leq N$, em que c_i representa o tipo do i -ésimo estabelecimento em ordem na Contorno.

Saída

Imprima S caso seja possível construir as estradas, e N caso contrário.

Restrições

- $1 \leq N \leq 10^3$.

Entrada	Saída
5 1 2 1 2 1	S

Entrada	Saída
9 1 2 1 1 2 1 2 2 3	N

Problema C. Dever de Casa

Nome do arquivo fonte: `dever.c`, `dever.cpp`, ou `dever.java`

Joãozinho acabou de aprender sobre divisão na escola. A professora passou o dever de casa, com N divisões que Joãozinho deve fazer, e entregar o resultado. Para cada uma delas, Joãozinho deve calcular o valor de a_i/b_i , a_i, b_i inteiros positivos e $1 \leq i \leq N$.

Porém, Joãozinho não é muito inteligente, e ele não entendeu muito bem como fazer quando há dízimas periódicas: ele acha que, se tiver que fazer $1/3 = 0.\overline{333333}\dots$, ele nunca mais vai conseguir terminar o dever! Joãozinho ficou desesperado.



Entretanto, Joãozinho é muito sagaz. Ele achou um furo: notou que a professora não especificou a base numérica em que os alunos devem representar o valor da divisão.

Sendo assim, ajude Joãozinho a descobrir a menor base $B \geq 2$ tal que, se escrevermos a_i/b_i na base B , para todo i , não teremos dízimas periódicas. Como a resposta pode ser muito grande, imprima o resto da divisão de B por $998\,244\,353$.

Entrada

A primeira linha contém um inteiro $1 \leq N \leq 10^5$.

As próximas N linhas contém dois inteiros separados por um espaço $1 \leq a_i, b_i \leq 10^5$.

Saída

Uma única linha com o menor valor da base $B \geq 2$ tal que nenhum dos a_i/b_i não forma dízima periódica na base B , módulo $998\,244\,353$.

Restrições

- $1 \leq N \leq 10^5$
- $1 \leq a_i, b_i \leq 10^5$.

Entrada	Saída
<pre>1 1 3</pre>	<pre>3</pre>

Entrada	Saída
2 9 3 5 2	2

Entrada	Saída
2 5 2 4 12	6

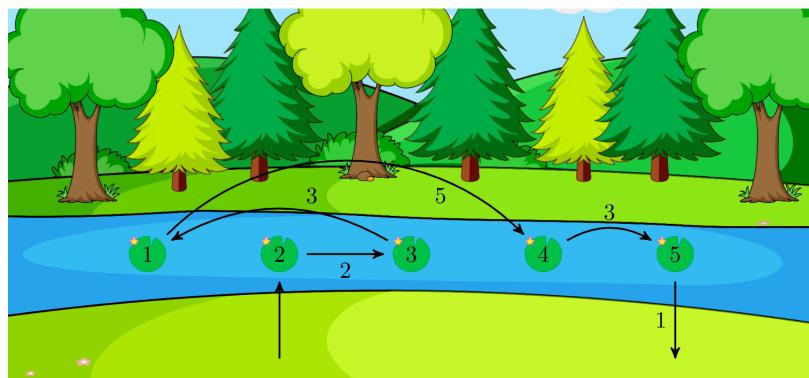
Explicação do Exemplo 1: Em base 2, $1/3$ é representado como $0.01010101\dots$. Mas, em base 3, $1/3$ é representado como 0.1 .

Problema D. Problema de Sapo

Nome do arquivo fonte: `sapo.c`, `sapo.cpp`, ou `sapo.java`

Um sapo está no pântano, na frente de um rio. No rio há N vitórias-régia, organizadas em uma sequência, da esquerda para a direita. O sapo sabe que, ao pular da vitória-régia A para a vitória-régia B , a A afunda para sempre no rio. Além disso, a i -ésima vitória-régia é tal que o sapo gasta x_i de energia para pular dela para a esquerda (**independente do tamanho do pulo**), e similarmente y_i para a direita.

O sapo, então, por diversão, irá escolher uma das vitórias-régia, pular nela, e depois pular para **todas as outras antes de sair do rio**, decidindo em cada momento pular para a esquerda ou direita. Como ele tem medo de cair no rio, em cada pulo, ele **sempre pula para a vitória-régia mais próxima na direção escolhida que ainda não afundou**. O sapo quer afundar todas as vitórias-régia dessa maneira, e, depois, pular para fora do rio. Se o caminho do sapo termina na posição i , o custo de pular dessa última vitória-régia para fora do rio é $\min(x_i, y_i)$.



Representação do exemplo. A energia gasta é $2 + 3 + 5 + 3 + 1 = 14$.

Ajude o sapo a descobrir alguma sequência de vitórias-régia que minimiza a energia total gasta por ele.

Entrada

A primeira linha da entrada contém um inteiro $1 \leq N \leq 10^5$, a quantidade de vitórias-régia no lago. As próximas N linhas possuem dois inteiros $1 \leq x_i, y_i \leq 10^9$, o quanto de energia que o sapo gasta para pular da i -ésima vitória-régia para a esquerda ou direita, respectivamente, independente do tamanho do pulo.

Saída

Imprima uma sequência de índices que descreve uma sequência de pulos que minimiza a energia total gasta pelo sapo. Se houver várias soluções, imprima qualquer uma delas.

Restrições

- $1 \leq N \leq 10^5$
- $1 \leq x_i, y_i \leq 10^9$

Entrada	Saída
5 3 5 2 2 3 4 3 3 4 1	2 3 1 4 5

Problema E. 14-bis

Nome do arquivo fonte: `14-bis.c`, `14-bis.cpp`, ou `14-bis.java`

Após inventar o avião, o grande mineiro Alberto Santos-Dumont precisava testar sua invenção.



14-bis, o primeiro avião da história a levantar voo

Para melhorar as chances de decolar sua lendária aeronave, o 14-bis, Santos-Dumont precisava de uma pista de decolagem que fosse bastante reta e o mais comprida possível. Uma pista de decolagem é descrita como uma sequência de números (representando a altura de cada trecho da pista) e esta é considerada reta se a diferença em valor absoluto entre duas posições adjacentes nessa sequência não é maior que um (não queremos estragar o trem de pouso da nossa querida aeronave de papel com terrenos acidentados!).

Dado um mapa topográfico da área, descrito como uma matriz $N \times M$ (em que cada valor da matriz guarda a altura da posição), imprima o tamanho da maior pista de decolagem possível, sendo que as pistas de decolagem podem ser no sentido norte-sul (subsequência de uma coluna da matriz) ou leste-oeste (subsequência de uma linha da matriz).

5	7	5	7	5	7
5	1	2	1	1	7
5	7	5	7	5	7

No exemplo 1 a maior pista de decolagem está no sentido leste-oeste e tem comprimento 4.

Entrada

A primeira linha conterá dois números N e M (tais que $N \times M \leq 2 \times 10^5$) que representam o número de linhas e colunas da matriz, respectivamente.

Depois disso teremos N linhas. Na i -ésima delas teremos M números $a_{i,1}, \dots, a_{i,M}$ onde $a_{i,j}$ representa o altura do terreno na linha i e coluna j da matriz e $0 \leq a_{i,j} \leq 10^9$.

Saída

Imprima um número que representa o tamanho da maior pista de pouso para o 14-bis.

Restrições

- $N \times M \leq 2 \times 10^5$
- $0 \leq a_{i,j} \leq 10^9$.

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

Entrada	Saída
1 10 1 3 5 4 5 6 7 8 9 7	7

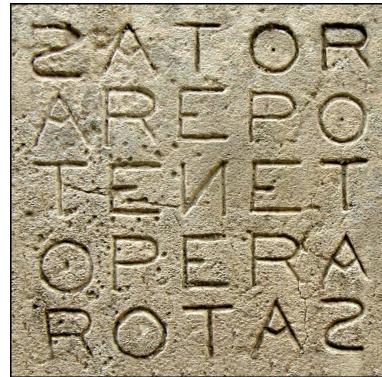
Entrada	Saída
5 5 1 3 2 4 5 3 2 4 1 3 2 3 5 7 2 3 4 1 9 8 5 5 3 0 4	5

Problema F. eenttt

Nome do arquivo fonte: `eenttt.c`, `eenttt.cpp`, ou `eenttt.java`

Emanuel estava estudando Programação Competitiva, resolvendo problemas sobre palíndromos. Um palíndromo é uma string que permanece igual quando lida de trás para frente, como `arara` e `tenet`. Além disso, Emanuel sabe que uma substring de uma string T é uma string que pode ser obtida apagando-se caracteres do início e fim de T . Por exemplo, `ara`, `ar` e `arara` são substrings de `arara`.

Ao se deparar com o problema clássico de computar quantas substrings de uma string S são palíndromos, Emanuel se perguntou como que esse problema poderia ser resolvido se pudessemos re-ordenar cada substring. Ou seja, dado uma string S , quantas substrings de S (contando repetições) podem ser re-ordenadas para formar um palíndromo.



Cansado após implementar o problema clássico, Emanuel pede sua ajuda para implementar a variação descrita acima para ele. Para simplificar o problema para você, ele garante que a string nunca conterá vogais (`a`, `e`, `i`, `o`, `u`, `y`).

Entrada

A primeira linha contém um inteiro N onde $1 \leq N \leq 10^6$.

A segunda linha contém uma string S com N caracteres. S é composta por letras minúsculas e não possui os caracteres `a`, `e`, `i`, `o`, `u`, `y`.

Saída

Imprima um único inteiro: o número de substrings de S que podem ser re-ordenadas para formar um palíndromo.

Restrições

- $1 \leq N \leq 10^6$.

Entrada	Saída
5 ffggh	12

Explicação do Exemplo 1: As 12 substrings são:

- | | |
|-------|-----------|
| 1. f | 7. gg |
| 2. f | 8. ffg |
| 3. g | 9. fgg |
| 4. g | 10. ggh |
| 5. h | 11. ffgg |
| 6. ff | 12. ffggh |

Problema G. O Andar do Trêbado

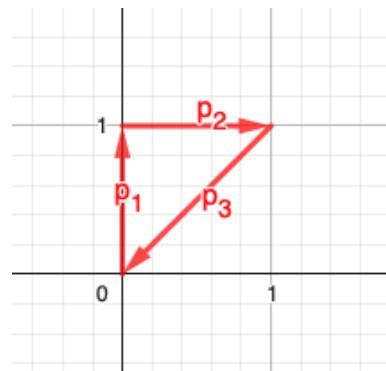
Nome do arquivo fonte: `trebado.c`, `trebado.cpp`, ou `trebado.java`

“O andar do bêbado” é um problema muito conhecido na matemática. Nesse problema temos um bêbado iniciando na origem $(0, 0)$ do plano 2D e, a cada momento, ele escolhe aleatoriamente uma direção, norte (N), sul (S), leste (L) ou oeste (O), e anda um metro nesta direção. Apesar do andar caótico, nesta versão original do problema é matematicamente garantido que depois de infinitos movimentos o bêbado sempre volta para sua casa a origem, alguma vez!

Apresentamos aqui um problema ligeiramente diferente, o bêbado pode fazer os movimentos:

- Norte (N): anda um metro para cima.
- Sul (S): anda um metro para baixo.
- Leste (E): anda um metro para a direita.
- Oeste (O): anda um metro para a esquerda.
- Nordeste (NE): anda um metro para cima e um metro para a direita.
- Sudeste (SE): anda um metro para baixo e um metro para a direita.
- Noroeste (NO): anda um metro para cima e um metro para a esquerda.
- Sudoeste (SO): anda um metro para baixo e um metro para a esquerda.

Dado que o bêbado inicia na origem $(0, 0)$, imprima de quantas formas ele pode fazer **exatamente K movimentos** de tal forma que após esses K movimentos ele esteja novamente em sua casa, a origem. Como o resultado pode ser grande, imprima a resposta módulo P , sendo P um primo entre 10^8 e 10^9 dado na entrada.



Exemplo do bêbado voltando para casa após três movimentos: N, L e SO

Entrada

A entrada é composta por dois inteiros K ($1 \leq K \leq 5000$) e P primo ($10^8 \leq P \leq 10^9$).

Saída

A saída deve ser composta por um único número: o número de formas que o bêbado pode voltar para a origem depois de K movimentos módulo P .

Restrições

- $1 \leq K \leq 5000$
- $10^8 \leq P \leq 10^9$

Entrada	Saída
3 100000007	24

Entrada	Saída
1 100001843	0

Entrada	Saída
100 100002031	70033170

Problema H. O Desfile das K-peias

Nome do arquivo fonte: `k-peia.c`, `k-peia.cpp`, ou `k-peia.java`

A fauna de Minas Gerais é muito diversa. Além dos animais mais conhecidos, tamanduás-bandeira, tatus, lobos guará e onças pintada, em Minas Gerais também se pode encontrar centopeias.

Ao andar pelo cerrado olhando atentamente para seus insetos, Elisa percebeu que a fauna de Minas é ainda mais diversa que o esperado, as "peias" aqui não se limitam a centopeias (100-peias). Aqui temos k -peias: bichos com k pés!

Percebendo essa maravilha da natureza, Elisa decidiu fazer um desfile de k -peias para mostrar ao mundo o quão majestosas elas são. Para deixá-las ainda mais belas, Elisa decidiu que apenas participarão dos desfiles as k -peias que tiverem sapatos em **todos os seus k pés**.



8-peia calçada.

Apesar de mais numerosos, os pés de k -peias são semelhantes aos pés humanos:

- É garantido que o número k de pés é par.
- Toda k -peia tem um tamanho de sapato t que é o mesmo para todos os pés.
- Metade dos pés de uma k -peia são esquerdos e, a outra metade, direitos.

Dada que toda k -peia possui um número de "majestosidade" (quão majestosa essa k -peia fica depois de calçada), ajude Elisa a distribuir seus sapatos de tal forma que a soma da majestosidade das k -peias seja maximizada.

Entrada

A primeira linha possui dois números $1 \leq N \leq 500$ e $1 \leq T \leq 500$ o número de k -peias encontradas por Elisa e o tamanho máximo de seus sapatos, respectivamente.

Depois temos N linhas descrevendo as k -peias. Na i -ésima delas temos três números k_i ($2 \leq k_i \leq 100$), t_i ($1 \leq t_i \leq T$) e m_i ($1 \leq m_i \leq 10^9$): o número de pés da i -ésima k -peia, o tamanho dos seus pés e sua majestosidade, respectivamente.

Por fim, teremos T linhas sendo que a j -ésima delas contém números e_j e d_j ($0 \leq e_j, d_j \leq 50000$): quantos sapatos esquerdos e direitos de tamanho j Elisa possui.

Saída

Imprima uma única linha contendo a maior soma das majestosidades das k -peias se Elisa distribuir seus sapatos de forma ótima.

Restrições

- $1 \leq N, T \leq 500$
- $2 \leq k_i \leq 100$
- $1 \leq t_i \leq T$
- $1 \leq m_i \leq 10^9$

Entrada	Saída
2 2 10 2 10 12 2 11 0 0 5 6	10

Problema I. Balões da Maratona

Nome do arquivo fonte: `baloes.c`, `baloes.cpp`, ou `baloes.java`

Chegou o dia da Maratona Mineira e a organização já tem as cores definidas para quase todos os problemas, faltando apenas um! Para facilitar a visualização do placar e do ambiente de prova, diminuindo a chance de confusões com balões de cores muito similares, a organização resolveu escolher esta última cor de forma que ela seja tão diferente das outras já selecionadas. As cores são codificadas no sistema RGB, através de triplas de inteiros que representam as quantidades de vermelho, verde e azul, respectivamente, na codificação da cor.



Dadas duas cores, (r_1, g_1, b_1) e (r_2, g_2, b_2) , calculamos a distância entre elas como $d = |r_1 - r_2| + |g_1 - g_2| + |b_1 - b_2|$. Sua tarefa é encontrar qual seria a **cor que maximiza a menor distância daquelas já existentes**. Caso mais de uma solução exista, aquela que minimiza a primeira coordenada da codificação deve ser escolhida. Persistindo o empate, deve ser escolhida aquela que minimiza a segunda coordenada. Caso mais de uma solução ainda exista aplicando estes critérios, deve ser escolhida aquela que minimiza a terceira coordenada. Em outras palavras, queremos a menor solução de forma lexicográfica.

Entrada

A primeira linha da entrada contém um inteiro $1 \leq N \leq 20$. Em seguida são dadas N linhas, de forma que a i -ésima linha, contém 3 números entre 0 e 255 (inclusivo), correspondendo ao código RGB da i -ésima cor já selecionada.

Saída

A saída contém três inteiros R, G, B , que correspondem ao código da cor que maximiza a distância para as demais cores.

Restrições

- $1 \leq N \leq 20$
- $0 \leq R, G, B \leq 255$.

Exemplos

Entrada	Saída
2 0 0 0 255 255 255	0 127 255

Entrada	Saída
2 100 100 255 100 100 0	255 255 127

Problema J. Jogo dos Copos

Nome do arquivo fonte: bolinha.c, bolinha.cpp, ou bolinha.java

Um famoso (e antigo) desafio jogado nas ruas consiste em esconder uma bolinha abaixo de um dentre três copos e dizer onde a bolinha ficou após o desafiante trocar os copos de posição.



Jogo da bolinha sendo jogado na era medieval.

Para garantir a vitória os desafiados costumam fazer movimentos muito rápidos. Entretanto, como você é esperto, você conseguiu ver todos os movimentos de "troca" dos copos.

Dado que os copos são numerados de 1 a 3 e a bolinha começa no copo do meio (copo 2), imprima qual é a posição final da bolinha após N movimentos de troca de copos.

Entrada

A primeira linha da entrada consiste em um inteiro N tal que $1 \leq N \leq 2 \times 10^5$. Depois seguem N linhas, cada uma contendo dois inteiros A, B ($1 \leq A, B \leq 3$), que indica que o copo A foi trocado com o copo B .

Saída

A saída deve consistir de um número de 1 a 3: qual foi a posição final da bolinha.

Restrições

- $1 \leq N \leq 2 \times 10^5$.

Entrada	Saída
<pre>3 1 3 2 3 3 1</pre>	1

Entrada	Saída
<pre>1 1 3</pre>	2

Problema K. Adedanha

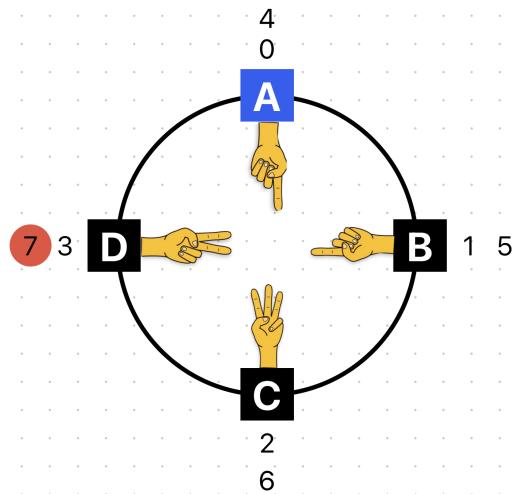
Nome do arquivo fonte: `adedanha.c`, `adedanha.cpp`, ou `adedanha.java`

Enquanto Alan e seus amigos estavam brincando, uma senhora os pediu para capinar seu lote. As crianças preferem brincar, mas, como são educadas, decidiram escolher uma das crianças para capinar o lote enquanto as outras se divertem. Para escolher quem capinará, jogarão adedanha.

Adedanha funciona da seguinte forma: todas as crianças esticam alguns dedos e o número de dedos esticados (um número de 0 a 20, dedos do pé também são dedos!) de cada criança são somados para um valor S . Depois disso conta-se **circularmente** as N crianças que são numeradas de 0 (Alan) a $N - 1$. Logo a contagem se dá na seguinte ordem: $0, 1, \dots, N - 2, N - 1, 0, 1, \dots$. Depois de contarmos S crianças, a criança em que a contagem parou será escolhida para a árdua tarefa de capinar o lote.

Alan é muito esperto e decidiu trapacear: ele só vai decidir o número de dedos que vai esticar logo depois de ver o número que cada uma das outras crianças esticou (Alan consegue contar dedos muito rápido, as outras crianças nem percebem sua trapaça!).

Dada a sua informação privilegiada Alan quer saber, para cada criança (incluindo ele próprio), se ele consegue escolher algum número de dedos para esticar de forma que esta criança capine (se vários números de dedos esticados forem possíveis para fazer certa criança capinar, imprima o menor deles) ou caso seja impossível, -1.



Como no Exemplo 1, se $N = 4$ e os amigos de Alan esticarem 1, 3 e 2 dedos, Alan pode esticar um dedo e assim o fim da contagem caia na criança 3 (D na figura).

Entrada

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 10^5$), o número de crianças participando da adedanha (incluindo Alan).

Depois disso temos $N - 1$ linhas com inteiros entre 0 e 20. A i -ésima linha contém o número de dedos que a criança $1, \dots, N - 1$ esticou (todas as crianças menos Alan).

Saída

A saída deve conter N linhas com números d_0, d_1, \dots, d_{N-1} . O número d_i deve ser o menor número de dedos que Alan deve esticar para que a criança i capine o lote e -1 caso seja impossível que i

capine.

Restrições

- $1 \leq N \leq 10^5$

Entrada	Saída
4	2
1	3
3	0
2	1

Entrada	Saída
3	2
5	0
20	1

Entrada	Saída
22	1
1	2
1	3
1	4
1	5
1	6
1	7
1	8
1	9
1	10
1	11
1	12
1	13
1	14
1	15
1	16
1	17
1	18
1	19
1	20
1	-1
1	0

Problema L. Bacon e a Árvore Mágica

Nome do arquivo fonte: `arvore.c`, `arvore.cpp`, ou `arvore.java`

Bacon, O Próspero, é reverenciado no reino das capivaras por seu glorioso reinado de paz e progresso, fruto de sua inabalável devoção ao Deboísmo, religião que hoje é seguida por mais de 90% dos habitantes da Bacônia. O Deboísmo prega que todos seus seguidores devem ficar o máximo possível de boa na lagoa, evitando assim os estresses e aflições que diminuem a qualidade de vida e geram o caos social. Além disso, ele patrocina várias festas populares e, certamente, as Festas da Cheia (para celebrar as chuvas de fim de ano) e suas árvores mágicas são as mais incríveis de todas.



Árvores mágicas possuem luz própria, são enraizadas, e funcionam da seguinte forma:

- A árvore inicia-se com todos os vértices apagados.
- Uma sub-árvore inteira pode se acender ou se apagar.
- Um vértice interno da árvore está aceso se e somente se todos os seus filhos estão acesos.

Além da magia da luz própria, a terceira propriedade também é bem misteriosa. Note que, se a sub-árvore enraizada em v é desligada, todos os ancestrais de v ligados também devem ser desligados; por outro lado se a sub-árvore de v é ligada, pode ser que tenhamos de ligar o pai de v , se todos seus filhos agora estiverem ligados, e assim sucessivamente.

Na 13^a Festa da Cheia de seu reinado, Bacon, O Próspero, levou seu bisneto, que viria a ser o rei Bacon, O Grafo, para ver a árvore mágica plantada no palácio. Como todo bom amante da combinatória, a pequena capivara perguntava incessantemente ao bisavô quantos vértices de uma dada sub-árvore estavam acesos. Infelizmente, nosso herói não é lá muito bom nessa área, e pediu a você, Grande Maratonista, para o ajudar a responder as perguntas do futuro monarca!

Entrada

A primeira linha contém um inteiro $1 \leq N \leq 10^5$, o número de vértices da árvore. As próximas $N - 1$ linhas contém as arestas $1 \leq a_i, b_i \leq N$ da árvore. É garantido que as arestas descrevem uma árvore, e a raiz é o vértice de número 1.

A próxima linha contém o um inteiro $1 \leq Q \leq 10^5$ que representa o número de eventos presenciados por Bacon, O Próspero. As próximas Q linhas contém 2 inteiros cada, t_i, v_i , em que $t_i \in \{1, 2, 3\}$ define o i -ésimo evento e $1 \leq v_i \leq N$ o vértice que enraiza a sub-árvore em questão.

- Se $t_i = 1$, então a sub-árvore enraizada em v_i se acende.
- Se $t_i = 2$, a sub-árvore enraizada em v_i se apaga.
- Se $t_i = 3$, o pequeno Bacon perguntou a seu bisavô quantos vértices na sub-árvore enraizada em v_i (inclusive v_i) estavam acesos.

Saída

Para eventos do tipo 3, imprima o número de vértices acesos na sub-árvore enraizada em v_i .

Restrições

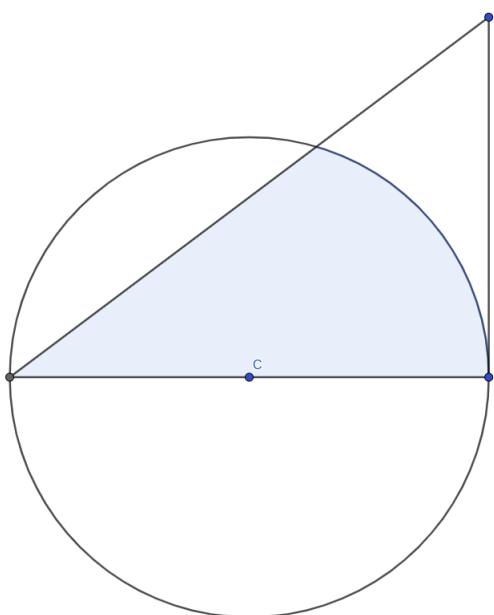
- $1 \leq N, Q \leq 10^5$.

Entrada	Saída
4	1
3 1	2
1 2	1
2 4	
8	
1 1	
2 4	
3 1	
1 4	
2 3	
3 1	
1 1	
3 3	

Problema M. Três, Quatro, Cinco

Nome do arquivo fonte: tresquatrocinco.c, tresquatrocinco.cpp, ou tresquatrocinco.java

Felipe Mota é especialista no ENEM. Ele notou que, no ENEM, sempre aparece o triângulo com lados 3, 4 e 5 nas questões. Ele se deparou com a seguinte questão do ENEM: dado uma circunferência de raio R , um triângulo semelhante ao com lados 3, 4 e 5 é colocado de tal forma que o lado proporcional ao 4 coincide com o diâmetro da circunferência. Qual é a área de interseção das figuras?



Ajude Felipe a resolver a questão.

Entrada

A entrada contém um único inteiro com o valor do raio $1 \leq R \leq 100$.

Saída

A saída deve ser a área de interseção das figuras. A saída será considerada correta se a diferença absoluta em relação à solução do juiz for no máximo 10^{-2} .

Restrições

- $1 \leq R \leq 100$

Entrada	Saída
1	1.123501108793

Problema N. Trapaça na Pastelaria

Nome do arquivo fonte: pastel.c, pastel.cpp, ou pastel.java

O Jogo possui 3 tabuleiros. Em cada tabuleiro, há uma peça e obstáculos. A peça inicialmente está na posição superior esquerda. Em um movimento, um jogador escolhe um dos 3 tabuleiros e move a peça uma quantidade não-nula de espaços para baixo ou para a direita de tal forma que ela não atravesse nem pare em um obstáculo. Jogam dois jogadores alternadamente, quem não tiver movimento, perde.

Você vai jogar O Jogo contra seu amigo. Ele vai começar, mas, logo antes, ele vai passar numa pastelaria para comer um pastel. Você, obviamente, vai roubar. Você vai fazer até um movimento em cada tabuleiro antes dele voltar (seu amigo é esperto, se você fizer mais de um movimento em um tabuleiro ele vai notar).



De quantas formas você pode fazer até um movimento em cada tabuleiro, de tal forma que, quando seu amigo voltar, ao jogar o jogo (ele começa), se ambos jogarem de forma ótima, você ganha?

Entrada

A entrada consiste em três tabuleiros. Cada tabuleiro é representado por uma linha com as dimensões N e M do tabuleiro $1 \leq N \times M \leq 10^5$. Em seguida há N linhas, cada uma com M caracteres: $.$ se for uma posição vazia e x se for um obstáculo. É garantido que nunca haverá um obstáculo na posição superior esquerda de um tabuleiro.

Saída

Um único inteiro: o número de formas de se fazer até um movimento em cada um dos três tabuleiros de forma que você ganha O Jogo.

Restrições

- $1 \leq N \times M \leq 10^5$

Entrada	Saída
2 2	
..	
..	
2 2	
..	
.x	
2 2	
.x	
xx	

Entrada	Saída
3 4	
....	
.x..	
....	
4 2	
..	
.x	
..	
x.	
2 2	
..	
..	

Explicação do Exemplo 1: Podemos fazer um dos dois movimentos possíveis no segundo tabuleiro; dessa forma, apenas o primeiro tabuleiro tem jogadas possíveis, e podemos ver que o primeiro jogador ganha. Outra possibilidade é fazer um dos dois movimentos no primeiro tabuleiro. Pode ser provado que no arranjo inicial o segundo jogador ganha.