II MARATONA PACODE

Julho de 2024

Caderno de problemas

Instituição: *CI - UFPB*

Instruções

Esta prova contém um total de **11** problemas. As páginas estão numeradas de 1 a 12, contando com essa página de rosto.

A) Sobre os nomes dos programas

- Se sua solução é implementada em C/C++ ou em Python, o nome do arquivo não é significativo, pode ser qualquer nome.
- Se sua solução for implementada em java, é obrigatório que o nome do arquivo seja no formato *Letra.java*, onde *Letra* corresponde à letra que representa o problema. Lembre que o nome da classe principal em java deve ter o mesmo nome do arquivo.

B) Sobre a entrada

- A entrada do seu programa deve ser lida da *entrada padrão*.
- A menos que a própria questão informe a quantidade ou as possíveis quantidades de casos de teste, assuma que esse valor é sempre 1.
- Quando a entrada possuir múltiplos valores, cada par de valores consecutivos numa mesma linha estará separado por exatamente um espaço em branco.

C) Sobre a saída

- A saída de seu programa deve ser escrita na *saída padrão*.
- Quando a saída possuir vários valores, esses valores devem estar separados por não mais que 1 espaço em branco.

A. Alberto está Insatisfeito

Costumam dizer que Alberto é um cara um pouco irritante, já que ele sempre reclama de tudo. Mas como também é muito engraçado, as pessoas tendem a relevar esse lado mais chato dele.

Acontece que Alberto é totalmente contrário a este rótulo. Ele afirma que suas reclamações são sempre acompanhadas de ação, que ele sempre procura mudar a realidade de forma a não precisar mais reclamar das coisas que o incomodam. Ele, por exemplo, tem falado muito dos buracos nas ruas da cidade (quem não se incomoda com isso?), ressaltando sua insatisfação com o prefeito. Tem um buraco na frente da casa dele que está lá há mais de trinta anos!

Cansado de ter que encarar essa realidade, ele convocou todos os habitantes insatisfeitos da cidade para taparem os buracos eles mesmos. Ao fim da reunião, ele conseguiu dividir os habitantes em N grupos de ação.

Sabendo que cada grupo consegue tapar exatamente um buraco um buraco por dia, que nenhum grupo ficará um dia sem trabalhar enquanto ainda houver buracos que não estão sendo tapados e que grupos diferentes nunca trabalham num mesmo buraco, quantos dias os habitantes levarão para tapar todos os buracos da cidade?

Entrada

A entrada consiste de dois valores inteiros. O primeiro valor é a quantidade N de grupos que Alberto conseguiu dividir os habitantes. O segundo valor é a quantidade K de buracos na cidade.

Perceba que pode acontecer de, no último dia, alguns grupos não trabalharem, como ocorre nos *Exemplos 02* e *03*.

Saída

Imprima a quantidade de dias necessários para que os habitantes consigam tapar todos os buracos.

Restrições

$1 \leq N \leq 100$
$1 \leq K \leq 10^5$

Exemplo de entrada 01:	**Exemplo de saída 01:**
2 10	5

Exemplo de entrada 02:	**Exemplo de saída 02:**
10 2	1

Exemplo de entrada 03:	**Exemplo de saída 03:**
3 20	7

B. Bit Strings

Alice sempre foi uma grande fã de strings, até mesmo antes de saber como programar. Na verdade, ela sempre gostou de ler, escrever e estudar a respeito das mais variadas formas de texto.

Recentemente, Alice descobriu uma nova forma de expressão poética, romântica, linda e quase sobrenatural: as strings binárias. Ela descobriu que pode representar cada caractere possível como uma sequência de zeros e uns, praticamente criando um novo alfabeto.

Ela, porém, ainda não entende muito das propriedades das strings binárias. Recentemente, disseram que a quantidade de combinações necessárias para poder escrever o que quisesse seria tão grande que tornaria a escrita completamente inviável.

Sendo assim, para ajudá-la a entender isso, sua tarefa é calcular o número de strings binárias de tamanho N.

Por exemplo, se $N = 3$, a resposta correta é 8, porque as possíveis strings binárias são 000, 001, 010, 011, 100, 101, 110 e 111.

Entrada

A entrada consiste apenas do valor inteiro N.

Saída

Imprima o número de strings binárias módulo $10^9 + 7$.

Restrições

$1 \leq N \leq 10^6$

Exemplo de entrada 01:	**Exemplo de saída 01:**
3	8

Exemplo de entrada 02:	**Exemplo de saída 02:**
1000000	235042059

C. Checksum

No mundo das redes de computadores, um problema crucial é garantir que os dados enviados de um computador para outro não sejam modificados no caminho. Para resolver isso, utiliza-se o *checksum*, uma técnica para verificar a integridade dos dados recebidos.

O *checksum* envolve a soma de dois valores incluídos na mensagem, que deve resultar em um valor binário onde todos os bits são 1. Por exemplo, se os números são 132 e 123, seus valores binários são 10000100 e 01111011, respectivamente. A soma desses valores binários é 11111111, confirmando a integridade dos dados.

Embora não seja infalível, essa técnica é útil e amplamente usada em datagramas UDP, um protocolo de rede comum.

O chefe de Bob pediu que ele escrevesse um programa que, dados dois números binários, verificasse se formam um *checksum* válido. Como Bob mentiu no currículo e não sabe programar, ele pediu sua ajuda para escrever o programa.

Para maior clareza, segue um caso de *checksum* **inválido**:

- representação binária do valor 130: 10000010
- representação binária do valor 124: 01111100
- 130 + 124 = 100000010 + 011111100 = 11111110 (inválido)

Entrada

A entrada consiste de dois números inteiros x_1, x_2: as representações binárias de 8 bits dos valores que fazem parte do *checksum*.

Saída

Imprima "SIM" (sem as aspas) se os valores formarem um *checksum* válido. Imprima "NAO" (sem as aspas) caso contrário.

Restrições

x_1, x_2 são valores **binários** no intervalo fechado $[0, 11111111]$

Exemplo de entrada 01:	**Exemplo de saída 01:**
10110010 01001101	SIM

Exemplo de entrada 02:	**Exemplo de saída 02:**
11111101 00000001	NAO

D. Divisores

Reza a lenda que estudantes de computação não são lá grandes fãs de cálculo, ou geometria, ou física, ou química, ou de acordar, nem de várias outras coisas.

Porém, é possível tirar muito proveito do estudo e da reflexão no campo da matemática, pois treina nosso cérebro para analisar de maneira lógica e coerente as situações, de forma a abstraí-las para um problema mais fácil de ser resolvido (ou simplesmente mais compreensível). E quem diz isso não sou eu, mas Bob, o grande Bob!

Para mostrar que a matemática, além de útil, também pode ser divertida, ele costuma propor problemas para seus alunos que abranjam apenas conteúdos que eles já conhecem e são capazes de resolver, se lhes for dado tempo suficiente. Dessa vez, para resolver o problema eles precisam escrever um programa, e você foi um dos desafiados.

Dado N inteiros, sua tarefa é calcular, para cada inteiro, sua quantidade de divisores. Por exemplo, se $x = 18$, a resposta correta é 6, porque seus divisores são 1, 2, 3, 6, 9 e 18.

Entrada

A primeira linha de entrada consiste de um valor inteiro N: o número de inteiros.

As N linhas seguintes contém, cada uma, um valor inteiro.

Saída

Imprima, para cada valor, a quantidade de seus divisores.

Restrições

$1 \leq N \leq 10^5$
$1 \leq x \leq 10^6$

Exemplo de entrada 01:	**Exemplo de saída 01:**
3 16 17 18	5 2 6

E. Espera Mínima

Bob agora é gerente de uma agência bancária, mas ele tem uma dor de cabeça enorme todos os dias: as pessoas sempre reclamam da demora para ser atendidas.

Bob odeia deixar seus clientes insatisfeitos, e definiu como meta da sua vida não descansar até fazer com que a insatisfação dos clientes seja a mínima possível. Mas como medir a insatisfação dos clientes? Outra dor de cabeça para Bob. Ele chamou os maiores profissionais, os mais competentes, os mais experientes e os mais estranhos também.

Mas as reuniões com esses profissionais não se mostraram muito promissoras, já que Bob não entendia absolutamente nada do que eles falavam. Eram sempre coisas tipo "Como assim você usa um sistema de filas não-preemptivo?", "Nunca ouviu falar de filas de prioridade?" e "FCFS é fácil e serve!". Cansado de ouvir tanta coisa sem sentido, ele demitiu todo mundo e decidiu arbitrariamente como resolver o problema: a insatisfação dos clientes seria a mínima se o somatório do tempo de espera de cada cliente fosse o mínimo possível. Portanto, se uma pessoa esperou 5min, outra 10min e ainda outra 15min, o somatório seria $5 + 10 + 15 = 30$.

A agência de Bob possui apenas um atendente para todos os clientes. Sabendo das suas capacidades duvidosas, mas ainda assim acreditando em você, Bob lhe contratou para escrever um programa que, dado o número de clientes e quanto tempo cada um leva sendo atendido, calcule o somatório dos tempos de espera caso o atendimento seja feito de maneira ótima - ou seja, caso o atendimento leve ao menor somatório possível dos tempos de espera.

Considere irrelevante o tempo de deslocamento/troca dos clientes em atendimento. Considere também que o primeiro cliente atendido espera 0 (zero) minutos e que **todos** os clientes estão na agência no início dos atendimentos.

Entrada

A primeira linha de entrada contém o inteiro N, a quantidade clientes. A segunda linha contém N números inteiros $a_1, a_2, \ldots, a_n$ separados por espaço, representado o tempo que cada cliente leva sendo atendido.

Saída

Imprima o somatório dos tempos de espera caso o atendimento seja realizado de maneira ótima, ou seja, de maneira que o somatório dos tempos de espera final seja mínimo.

Restrições

$1 \leq N \leq 10^5$
$1 \leq a_1, a_2, \ldots, a_n \leq 100$

Exemplo de entrada 01:	**Exemplo de saída 01:**
5 1 2 3 3 1	14

Exemplo de entrada 02:	**Exemplo de saída 02:**
2 80 3	3

No *Exemplo 01*, há duas pessoas que demoram 1min sendo atendidas, duas pessoas que demoram 3min, e uma pessoa que demora 2min. Atendendo-as de maneira ótima a soma dos tempos de espera é de 14min, e uma das formas de fazê-lo é a sequência $0 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow 3$, sendo cada valor correspondente ao índice do cliente na lista de entrada.

F. Fila do Sorvete

Galadriel é o proprietário de uma sorveteria e adora seu trabalho. Atender os clientes com excelência e proporcionar-lhes o sorvete mais delicioso de suas vidas é um grande prazer para ele. No entanto, sendo o único funcionário da sorveteria, ele nem sempre consegue atender a todos os clientes, o que o deixa bastante triste. Galadriel se pergunta se há alguma forma de melhorar o atendimento.

Para avaliar a eficiência de seu trabalho, ele pediu sua ajuda. Galadriel conhece bem seus fiéis clientes: sabe a hora em que cada um vai chegar, quanto tempo demora para ser atendido e quanto tempo está disposto a esperar na fila antes de desistir. Se um cliente não for atendido dentro desse período de espera, ele vai embora.

Com base na quantidade N de clientes que irão à sorveteria em um determinado dia e, para cada um deles, a hora de chegada H, o tempo de atendimento T e o tempo máximo de espera E, informe a Galadriel quantos clientes ele deixará de atender, ou seja, quantos irão embora antes de serem atendidos. Lembrando que, se há mais de um cliente na fila, o que chegou antes sempre será atendido primeiro.

Entrada

A primeira linha de entrada consiste no valor N, a quantidade de clientes que Galadriel receberá naquele dia. Cada uma das próximas N linhas contém três valores inteiros H, T e E sobre cada cliente: a hora de chegada, o tempo que demora para ser atendido e o tempo máximo que está disposto a esperar na fila.

É **garantido** que as informações dos clientes serão fornecidas em **ordem cronológica**. Observe que, como demonstrado no *Exemplo 01*, um cliente pode começar a ser atendido exatamente na hora em que outro terminou. No exemplo citado, o primeiro cliente começa a ser atendido no tempo 0 (zero) e termina no tempo 1 (um); o segundo cliente chega exatamente no tempo 1 (um) e não está disposto a esperar ($E = 0$), mas ainda assim é atendido. Portanto, a resposta é 0 (zero), pois Galadriel **não deixou** de atender nenhum cliente.

No *Exemplo 02*, Galadriel não consegue atender o terceiro cliente.

Saída

Informe a quantidade de clientes que Galadriel **deixará** de atender.

Restrições

$1 \leq N \leq 10^3$
$0 \leq H, T, E \leq 10^3$

Exemplo de entrada 01:	**Exemplo de saída 01:**
2 0 1 1 1 1 0	0

Exemplo de entrada 02:	**Exemplo de saída 02:**
5 1 2 3 2 1 1 3 4 0 5 6 9 11 1 8	1

G. Geometria de Inteiros

Há uma reta que passa pelos pontos $p_1 = (x_1, y_1)$ e $p_2 = (x_2, y_2)$. Há também um ponto $p_3 = (x_3, y_3)$.

Sua tarefa é determinar se p_3 está localizado à esquerda, à direita ou se toca na reta definida pelos pontos p_1 e p_2.

Entrada

A primeira linha de entrada contém um inteiro t: o número de testes.
Após isso, há t linhas descrevendo os testes. Cada linha tem seis inteiros: x_1, y_1, x_2, y_2, x_3 e y_3.

Saída

Para cada caso de teste, imprima "ESQUERDA" (sem as aspas) caso p_3 esteja à esquerda da reta, "DIREITA" (sem as aspas) caso p_3 esteja à direita da reta, ou "TOCANDO" caso p_3 faça parte da reta.

Restrições

$1 \leq t \leq 10^5$
$-10^9 \leq x_1, y_1, x_2, y_2, x_3, y_3 \leq 10^9$
$x_1 \neq x_2$ ou $y_1 \neq y_2$

Exemplo de entrada 01:	**Exemplo de saída 01:**
3 1 1 5 3 2 3 1 1 5 3 4 1 1 1 5 3 3 2	ESQUERDA DIREITA TOCANDO

H. Haja Troca

Lhe será dado um array $a[0 \ldots N-1]$ de tamanho N consistindo de inteiros não-negativos.

Um array é considerado *bom* se a paridade de cada índice coincide com o elemento do índice. Mais formalmente, um array é dito *bom* se para todo i a igualdade $i \ mod \ 2 = a[i] \ mod \ 2$ se mantém verdadeira, onde $x \ mod \ 2$ é o resto da divisão de x por 2.

Os arrays $[0, 5, 2, 1]$ e $[0, 17, 0, 3]$, por exemplo, são *bons*, enquanto o array $[2, 4, 6, 7]$ são ruins, porque para $i = 1$, as paridades de i e $a[i]$ são diferentes: $1 \ mod \ 2 = 1$, mas $4 \ mod \ 2 = 0$.

Em um movimento, você pode escolher **quaisquer** dois elementos do array e trocá-los de posição (os elementos **não precisam** ser adjacentes).

Encontre o número mínimo de movimentos necessários para tornar o array **bom**, ou informe que isso é impossível.

Lembre-se de que os índices de um array começam em 0 (zero), não em 1 (um).

Entrada

A primeira linha de entrada contém o valor N: o tamanho do array.

A segunda linha de entrada contém N elementos $a_0, a_1, \ldots, a_{N-1}$: o estado inicial do array.

Saída

Imprima o número mínimo de movimentos - possivelmente 0 (zero) - para tornar o array *bom*, ou -1 caso isso não seja possível.

Restrições

$1 \leq N \leq 1000$
$1 \leq a_i \leq 1000$

Exemplo de entrada 01:	**Exemplo de saída 01:**
4 3 2 7 6	2

Exemplo de entrada 02:	**Exemplo de saída 02:**
3 3 2 6	1

Exemplo de entrada 03:	**Exemplo de saída 03:**
1 7	-1

I. Ímpares e Pares

Bob está inconformado com o estado atual das coisas, mas especialmente com a ordem dos números inteiros maiores que zero. Ele está determinado a reordená-los, mas há muitos números naturais, então ele decidiu começar com os primeiros N. Assim, ele escreve sua própria sequência de números: primeiramente todos os inteiros ímpares de 1 a N (em ordem crescente), e depois todos os pares (também em ordem crescente).

Ajude o nosso herói a descobrir qual número estará na posição K.

Entrada

A entrada consiste dos valores N e K, respectivamente.

Saída

Imprima o número que estará na posição K depois do rearranjo de Bob.

Restrições

$1 \leq K \leq N \leq 10^{12}$

Exemplo de entrada 01:	**Exemplo de saída 01:**
10 3	5

Exemplo de entrada 02:	**Exemplo de saída 02:**
7 7	6

J. Jogando Jogos

Você está jogando o seguinte Jogo de Nim com seu amigo:

- Inicialmente, há uma pilha de pedras na mesa;
- Você e seu amigo jogam em turnos, e **você começa**;
- A cada turno, a pessoa da vez retira entre 1 e 3 pedras da pilha;
- Aquele que remover a última pedra é o vencedor.

Dado N, a quantidade de pedras na pilha, informe se você consegue vencer o jogo ou não, **assumindo que você e seu amigo jogam de maneira ótima**.

Possíveis definições do que seria "jogar de maneira ótima":

- "Jogar de maneira ótima" significa que, se há uma maneira de jogar que **garantidamente** o levará à vitória, você (ou o outro jogador) jogará desta maneira.
- Em teoria dos jogos, "jogar de maneira ótima" significa adotar estratégias que maximizam os ganhos ou minimizam as perdas, considerando as possíveis ações dos outros jogadores.

Sendo assim, se o jogo começa com 5 pedras, por exemplo, então você deveria retirar apenas 1 (uma) pedra, pois isso deixa o jogo com uma configuração que **garante** a sua vitória, pois independente do número de pedras que seu amigo retirar, você conseguirá retirar todas as outras na sua próxima rodada. Igualmente, se o jogo começasse com 4 pedras, você **garantidamente** perderia (lembre-se que seu amigo também está jogando de forma ótima).

Entrada

A entrada consiste de um único número inteiro N: a quantidade de pedras na pilha no início do jogo.

Saída

Imprima "VITORIA" (sem as aspas) caso você consiga vencer se jogar de maneira ótima. Imprima "DERROTA" (sem as aspas) caso contrário.

Restrições

$1 \leq N \leq 2 \times 10^9$

Exemplo de entrada 01:	**Exemplo de saída 01:**
4	DERROTA

Exemplo de entrada 01:	**Exemplo de saída 01:**
1	VITORIA

Exemplo de entrada 01:	**Exemplo de saída 01:**
5	VITORIA

K. Kuadrados

Kuadrado começou a estudar programação recentemente e está muito animado com todas as novas coisas que consegue fazer a cada conteúdo novo aprendido. Você conhece bem esta sensação, não é?

Ele está estudando matrizes e de vez em quando faz alguns desafios a si mesmo para ver o quanto realmente sabe. Ele normalmente consegue superá-los, mas agora chegou em um no qual não consegue resolver, e por isso pediu a sua ajuda.

O desafio consiste em, dada uma matriz de N linhas e M colunas, encontrar todas as submatrizes de tamanho 2×2 nas quais **todos** os valores são iguais a 1.

Entrada

A primeira linha de entrada consiste em dois valores N e M, a quantidade de linhas e de colunas da matriz, respectivamente.

As próximas N linhas contém M valores inteiros cada uma: a matriz. A matriz possui apenas os valores 0 e 1.

Perceba que pode ocorrer de duas matrizes diferentes possuírem intercessão entre si, mas devem ser consideradas como diferentes. O *Exemplo de entrada 01* deixa claro isso: se não fosse permitido intercessão entre as submatrizes, a resposta correta seria 1 (um), mas, como é permitido, há 2 (duas) submatrizes de tamanho 2×2 que possuem apenas o valor 1.

Saída

Imprima quantas submatrizes que atendem aos requisitos existem na matriz fornecida na entrada.

Restrições

$1 \leq N, M \leq 100$
$a_{n,m} = 0$ ou $a_{n,m} = 1$

Exemplo de entrada 01:	**Exemplo de saída 01:**
2 3 1 1 1 1 1 1	2

Exemplo de entrada 02:	**Exemplo de saída 02:**
3 3 1 1 0 1 1 1 0 1 1	2

Exemplo de entrada 03:	**Exemplo de saída 03:**
3 4 1 1 1 0 1 1 1 0 1 1 0 0	3