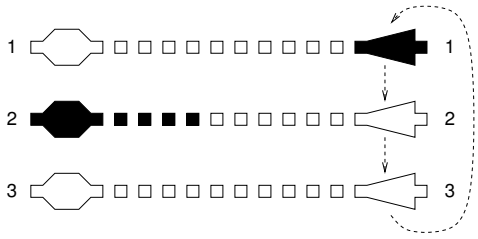
Game-10

Nome do arquivo: `game10.c`, `game10.cpp`, `game10.pas`, `game10.java`, `game10.js` *ou* `game10.py`

No princípio dos anos 1980 surgiram nos colégios os primeiros relógios de pulso digitais com joguinhos. Era uma febre entre os alunos e quem tinha um era muito popular na hora do recreio. Os joguinhos eram bem simples, mas muito legais. Um dos primeiros era o Game-10, no qual você controlava um avião que aparecia na parte direita do visor. Na parte esquerda aparecia um disco voador em qualquer uma de três posições, aleatoriamente, e lançava um míssil. O objetivo do jogador era movimentar o avião verticalmente para que ficasse na frente do disco voador (na mesma linha horizontal, do lado direito) e atirar para interceptar o míssil antes que esse atingisse o avião.



Como o movimento do avião era feito com apenas um botão, só dava para movimentar em um sentido: ao apertar o botão sucessivas vezes, o avião se movia na sequência de posições $\cdots 1 \to 2 \to 3 \to 1 \to 2 \to 3 \to 1 \cdots$ Veja que, na situação da figura, o jogador deveria apertar o botão apenas uma vez, para ir da posição 1 para a posição 2, e conseguir atirar e interceptar o míssil.

Neste problema vamos considerar que existem N posições e não apenas três. Dado o número de posições N, a posição D na qual o disco voador aparece, e a posição A onde está o avião, seu programa deve computar o número mínimo de vezes que o jogador precisa apertar o botão para movimentar o avião até a mesma posição do disco voador e poder atirar!

Entrada

A primeira linha da entrada contém um inteiro N, o número de posições. A segunda linha contém um inteiro D, a posição do disco voador. A terceira linha contém um inteiro A, a posição do avião.

Saída

Seu programa deve imprimir uma linha contendo um inteiro, o número mínimo de vezes que o jogador deve apertar o botão para poder atirar.

Restrições

- $3 \leq N \leq 100$
- $1 \leq D, A \leq N$

Exemplos

Entrada	**Saída**
3 2 1	1

Entrada	**Saída**
20 8 13	15

Drone de Entrega

Nome do arquivo: `drone.c`, `drone.cpp`, `drone.pas`, `drone.java`, `drone.js` *ou* `drone.py`

A loja do Pará, especializada em vendas pela internet, está desenvolvendo *drones* para entrega de caixas com as compras dos clientes. Cada caixa tem a forma de um paralelepípedo reto retângulo (ou seja, no formato de um tijolo).

O *drone* entregará uma caixa de cada vez, e colocará a caixa diretamente dentro da casa do cliente, através de uma janela. Todas as janelas dos clientes têm o formato retangular e estão sempre totalmente abertas. O *drone* tem um aplicativo de visão computacional que calcula exatamente as dimensões H e L da janela. O *drone* consegue colocar a caixa através da janela somente quando uma das faces da caixa está paralela à janela, mas consegue virar e rotacionar a caixa antes de passá-la pela janela.

O aplicativo de controle do *drone* está quase pronto, mas falta um pequeno detalhe: um programa que, dadas as dimensões da maior janela do cliente e as dimensões da caixa que deve ser entregue, determine se o *drone* vai ser capaz de entregar a compra (pela janela) ou se a compra terá que ser entregue por meios normais.

Entrada

A entrada é composta por cinco linhas, cada uma contendo um número inteiro. A três primeiras linhas contêm os valores A, B, C, indicando as três dimensões da caixa, em centímetros. As duas últimas linhas contêm os valores H e L, indicando a altura e a largura da janela, em centímetros.

Saída

Seu programa deve escrever uma única linha, contendo apenas a letra `S` se a caixa passa pela janela e apenas a letra `N` em caso contrário.

Restrições

- $1 \leq A, B, C \leq 80$
- $1 \leq H, L \leq 100$

Exemplos

Entrada	**Saída**
30 50 80 80 60	S

Entrada	**Saída**
75 100 50 100 30	N

Língua do P

Nome do arquivo fonte: `lingua.c`, `lingua.cpp`, `lingua.pas`, `lingua.java`, *ou* `lingua.py`

Uma brincadeira que crianças adoram é se comunicar na *língua do P*, acrescentando *pê* antes de cada sílaba, como uma forma de código para dificultar que outras pessoas entendam a conversa (pê-va pê-mos pê-no pê-ci pê-ne pê-ma?).

Jacy e Kátia adaptaram a língua do P para mensagens eletrônicas, acrescentando a letra P minúscula '`p`' antes de cada letra das palavras de uma mensagem. Um exemplo de mensagem codificada e a respectiva mensagem decodificada é mostrada na figura abaixo.

Mensagem codificada	Mensagem decodificada
`pVpapmpops papo pcpipnpepmpa`	`Vamos ao cinema`

Sua tarefa é escrever um programa que decodifique uma mensagem escrita na língua do P eletrônica de Jacy e Kátia.

Entrada

A entrada consiste de uma única linha, contendo uma mensagem escrita na língua do P eletrônica de Jacy e Kátia.

Saída

Seu programa deve produzir uma única linha, contendo a mensagem decodificada.

Restrições

- A mensagem contém apenas letras maiúsculas e minúsculas e espaços em branco.
- A mensagem tem entre 1 e 1000 caracteres.
- Não há dois espaços em branco consecutivos na mensagem.

Exemplos

Entrada	**Saída**
`pUpm pfpiplpmpe plpepgpapl`	`Um filme legal`

Entrada	**Saída**
`pA pppapppa pdpo pPpapppa`	`A papa do Papa`

Cobra coral

Nome do arquivo: `coral.c`, `coral.cpp`, `coral.pas`, `coral.java`, `coral.js` *ou* `coral.py`

O professor Rui está desenvolvendo um sistema automático para identificar se uma cobra é uma coral verdadeira ou uma falsa coral. A cobra coral verdadeira é venenosa e os anéis coloridos no seu corpo seguem o padrão `...BVBPBVBPBVBP...`, onde `B`,`V` e `P` representam as cores branco, vermelho e preto, respectivamente. Já a falsa coral não é venenosa e os anéis seguem o padrão `...BVPBVPBVPBVP....`

O problema é que os sensores do sistema do professor Rui produzem apenas uma sequência de quatro números representando um pedaço do padrão de cores. Só que ele não sabe qual número representa qual cor. Mas, por exemplo, se a sequência for `5 3 9 3`, podemos dizer com certeza que é uma coral verdadeira, mesmo sem saber qual número representa qual cor! Você deve ajudar o professor Rui e escrever um programa que diga se a coral é verdadeira ou falsa.

Entrada

A entrada consiste de apenas uma linha, contendo quatro números inteiros.

Saída

Seu programa deve imprimir na saída uma linha com a letra "`V`" se a coral for verdadeira ou com a letra "`F`", caso seja falsa.

Restrições

- Os quatro números têm valores entre 1 e 9, inclusive, e a sequência sempre representa uma coral verdadeira, ou uma coral falsa.

Exemplos

Entrada	**Saída**
`5 3 9 3`	`V`

Entrada	**Saída**
`7 1 4 7`	`F`

Entrada	**Saída**
`6 2 6 8`	`V`

Fechadura

Nome do arquivo fonte: `fechadura.c`, `fechadura.cpp`, `fechadura.pas`, `fechadura.java`, *ou* `fechadura.py`

Joãozinho estava um dia chegando em casa quando percebeu que havia perdido a chave da porta. Desesperado, ele resolveu pedir ajuda a seu amigo Roberto, que em poucos segundos conseguiu abrir a porta usando suas ferramentas.

Admirado com a velocidade em que seu amigo conseguiu abrir a porta de sua casa sem a chave, ele decidiu perguntar como ele tinha conseguido aquilo. Roberto explicou que a fechadura da casa de Joãozinho é baseada em um sistema de pinos de tamanhos diferentes que, uma vez alinhados na mesma altura M, possibilitam a abertura da porta.

Uma fechadura é um conjunto de N pinos dispostos horizontalmente que podem ser movimentados para cima ou para baixo com o auxílio de uma chave de metal que, ao ser inserida dentro da fechadura, pode aumentar ou diminuir em 1mm, simultaneamente, a altura de quaisquer dois pinos consecutivos.

Joãozinho como um exemplar perfeccionista decidiu desbloquear sua fechadura na menor quantidade de movimentos, onde cada movimento consiste em escolher dois pinos consecutivos da fechadura e aumentar ou diminuir a altura dos dois pinos em 1mm. Após todos os pinos possuírem altura exatamente igual a M, a fechadura é desbloqueada.

Entrada

A primeira linha da entrada contém dois inteiros N e M representando, respectivamente, a quantidade de pinos da fechadura e a altura em que eles devem ficar para a fechadura ser desbloqueada.

A segunda linha da entrada contém N inteiros, representando as alturas dos pinos da fechadura.

Saída

Seu programa deve imprimir uma linha contendo um inteiro representando a quantidade mínima de movimentos para desbloquear a fechadura.

Restrições

- $1 \leq N \leq 1000$
- $1 \leq M \leq 100$
- Cada altura dos pinos está entre 1 e 100.
- É garantido que os casos de testes sempre possuem uma solução.

Exemplos

Entrada	**Saída**
`4 50` `45 45 55 55`	`10`

Entrada	**Saída**
`5 84` `84 39 17 72 94`	`77`

Sanduíche

Nome do arquivo: `sanduiche.c`, `sanduiche.cpp`, `sanduiche.pas`, `sanduiche.java`, `sanduiche.js`, `sanduiche.py2` *ou* `sanduiche.py3`

Você está na Seletiva para a IOI e depois de um dia cansativo de provas, chegou a hora do jantar. Hoje, trouxeram um sanduíche muito longo cortado em N pedaços de diversos tamanhos diferentes. Você gostaria de comer uma quantidade total de sanduíche de comprimento D, porém há uma regra: para evitar bagunça, você só pode ou pegar uma sequência contínua de pedaços, ou pegar pedaços das extremidades. Você sabe a sequência $C_1, C_2, ... C_N$ dos comprimentos dos pedaços na ordem em que estão posicionados no sanduíche. Agora, para otimizar o seu jantar, quer fazer um programa que com esses dados responda de quantas formas você pode escolher os pedaços do sanduíche que vai comer. Em outras palavras, deve contar quantos pares (i, j), $1 \leq i \leq j \leq N$, existem tais que o somatório $C_i + C_{i+1} + ... + C_j$ seja igual a D e quantos pares (i, j), $1 \leq i < j \leq N$, existem tais que o somatório $C_1 + C_2 + ... + C_i + C_j + C_{j+1} + ... + C_N$ seja igual a D.

Entrada

A primeira linha contém dois inteiros N e D, representando respectivamente o número de pedaços e a quantidade de sanduíche que você quer comer. A segunda linha contém N inteiros $C_1, C_2, ... C_N$, onde C_i é o tamanho do i-ésimo pedaço.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número de maneiras de comer pedaços de sanduíche com soma D.

Restrições

- $2 \leq N \leq 10^6$.
- $1 \leq D \leq 10^9$.
- $1 \leq C_i \leq 10^3$.

Informações sobre a pontuação

- Em um conjunto de casos de teste equivalente a 20 pontos, $N \leq 200$.
- Em um conjunto de casos de teste equivalente a 40 pontos, $N \leq 1000$.

Exemplos

Entrada	**Saída**
`5 10` `1 2 3 4 3`	`3`

Entrada	**Saída**
`5 5` `1 1 1 1 1`	`5`

Entrada	**Saída**
9 618 665 658 248 282 428 562 741 290 457 5	0