1 Laboratório 16

1. **Fechem as Portas!**
2. **Café**
3. **Varetas**
4. **Subprime**
5. **Trabalho prático**

2 Laboratório 17

1. **Handebol**
2. **Circuito Bioquímico Digital**
3. **Elevador**
4. **Loop Musical**
5. **Tomadas**
6. **Jogo**
7. **Andando no Tempo**

3 Extra

1. **Revisão de Contrato**
2. **Laser**
3. **Plágio**

Problema A
Fechem as portas!

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

Madame Beauvoir possui uma mansão onde ela recebe todos os seus descendentes (netos e bisnetos) durante as férias. Sua mansão possui exatamente N quartos (cada quarto é numerado de 1 a N), onde N é também a quantidade de netos e bisnetos (cada descendente é também numerado de 1 a N).

Como toda criança, os descendentes de Mme. Beauvoir são bastante travessos. Todo dia é a mesma confusão: eles acordam de manhã cedo antes dela e se encontram no grande jardim. Cada descendente, um de cada vez, entra na mansão e troca o estado das portas dos quartos cujos números são múltiplos do seu identificador. Trocar o estado de uma porta significa fechar uma porta que estava aberta ou abrir uma porta que estava fechada. Por exemplo, o descendente cujo identificador é igual a 15 vai trocar o estado das portas 15, 30, 45, etc.

Considerando que todas as portas estão inicialmente fechadas (todos os descendentes fecham as portas antes de descer para o jardim) e que cada descendente entra exatamente uma vez na mansão (a confusão é tão grande que não sabemos em que ordem), quais portas estarão abertas após a entrada de todos os descendentes na mansão?

Entrada

A entrada contém vários casos de teste. Cada caso de teste consiste em uma linha que contém um inteiro N ($0 \leq N \leq 25000000$), indicando o número de portas e descendentes. O final da entrada é indicado por $N = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma linha na saída, contendo a seqüência crescente de números correspondente aos identificadores dos quartos cujas portas estarão abertas. Ao imprimir a seqüência, deixe um espaço em branco entre dois elementos consecutivos.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**
1 2 3 4 0	1 1 1 1 4

Problema M

Máquina de café

O novo prédio da Sociedade Brasileira de Computação (SBC) possui 3 andares. Em determinadas épocas do ano, os funcionários da SBC bebem muito café. Por conta disso, a presidência da SBC decidiu presentear os funcionários com uma nova máquina de expresso. Esta máquina deve ser instalada em um dos 3 andares, mas a instalação deve ser feita de forma que as pessoas não percam muito tempo subindo e descendo escadas.

Cada funcionário da SBC bebe 1 café expresso por dia. Ele precisa ir do andar onde trabalha até o andar onde está a máquina e voltar para seu posto de trabalho. Todo funcionário leva 1 minuto para subir ou descer um andar. Como a SBC se importa muito com a eficiência, ela quer posicionar a máquina de forma a minimizar o tempo total gasto subindo e descendo escadas.

Sua tarefa é ajudar a diretoria a posicionar a máquina de forma a minimizar o tempo total gasto pelos funcionários subindo e descendo escadas.

Entrada

A entrada consiste em 3 números, A_1, A_2, A_3 ($0 \leq A_1, A_2, A_3 \leq 1000$), um por linha, onde A_i representa o número de pessoas que trabalham no i-ésimo andar.

Saída

Seu programa deve imprimir uma única linha, contendo o número total de minutos a serem gastos com o melhor posicionamento possível da máquina.

Exemplo de entrada 1	**Exemplo de saída 1**
10 20 30	80

Exemplo de entrada 2	**Exemplo de saída 2**
10 30 20	60

Exemplo de entrada 3	**Exemplo de saída 3**
30 10 20	100

Problema H
Jogo de Varetas

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

Há muitos jogos divertidos que usam pequenas varetas coloridas. A variante usada neste problema envolve a construção de retângulos. O jogo consiste em, dado um conjunto de varetas de comprimentos variados, desenhar retângulos no chão, utilizando as varetas como lados dos retângulos, sendo que cada vareta pode ser utilizada em apenas um retângulo, e cada lado de um retângulo é formado por uma única vareta. Nesse jogo, duas crianças recebem dois conjuntos iguais de varetas. Ganha o jogo a criança que desenhar o maior número de retângulos com o conjunto de varetas.

Dado um conjunto de varetas de comprimentos inteiros, você deve escrever um programa para determinar o maior número de retângulos que é possível desenhar.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro N que indica o número de diferentes comprimentos de varetas ($1 \leq N \leq 1.000$) no conjunto. Cada uma das N linhas seguintes contém dois números inteiros C_i e V_i, representando respectivamente um comprimento ($1 \leq C_i \leq 10.000$) e o número de varetas com esse comprimento ($1 \leq V_i \leq 1.000$). Cada comprimento de vareta aparece no máximo uma vez em um conjunto de teste (ou seja, os valores C_i são distintos). O final da entrada é indicado por $N = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma única linha na saída, contendo um número inteiro, indicando o número máximo de retângulos que podem ser formados com o conjunto de varetas dado.

A saída deve ser escrita na saída padrão.

Exemplo de entrada	**Saída para o exemplo de entrada**
1 10 7 4 50 2 40 2 30 4 60 4 5 15 3 6 3 12 3 70 5 71 1 0	1 3 2

Problema D
Sub-prime

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

A mais recente crise econômica foi em parte causada pela forma como os bancos faziam empréstimos para pessoas que não tinham capacidade de honrá-los e revendiam tais empréstimos para outros bancos (debêntures). Obviamente, quando as pessoas pararam de pagar os empréstimos, o sistema inteiro entrou em colapso.

A crise foi tão profunda que acabou atingindo países do mundo inteiro, inclusive a Nlogônia, onde o honrado primeiro ministro Man Dashuva ordenou que o presidente do Banco Central procurasse uma solução para o problema. Esse, por sua vez, teve uma ideia brilhante: se cada banco fosse capaz de liquidar seus empréstimos somente com suas reservas monetárias, todos os bancos sobreviveriam e a crise seria evitada.

Entretanto, com o elevado número de debêntures e bancos envolvidos, essa tarefa é extremamente complicada, e portanto ele pediu a sua ajuda para escrever um programa que, dados os bancos e as debêntures emitidas, determine se é possível que todos os bancos paguem suas dívidas, utilizando suas reservas monetárias e seus créditos.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém dois inteiros B e N, indicando respectivamente o número de bancos ($1 \leq B \leq 20$) e o número de debêntures emitidas pelos bancos ($1 \leq N \leq 20$). Os bancos são identificados por inteiros entre 1 e B. A segunda linha contém B inteiros R_i separados por espaços, indicando as reservas monetárias de cada um dos B bancos ($0 \leq R_i \leq 10^4$, para $1 \leq i \leq B$). As N linhas seguintes contêm cada uma três inteiros separados por espaços: um inteiro D, indicando o banco devedor ($1 \leq D \leq B$), um inteiro C, indicando o banco credor ($1 \leq C \leq B$ e $D \neq C$), e um inteiro V, indicando o valor da debênture ($1 \leq V \leq 10^4$).

O final da entrada é indicado por uma linha que contém apenas dois zeros, separados por um espaço em branco.

Os dados devem ser lidos da entrada padrão.

Saída

Para caso de teste, seu programa deve imprimir uma única linha, contendo um único caractere: '`S`', se for possível liquidar todos as debêntures sem intervenção do Banco Central da Nlogônia, e '`N`', se algum banco precisar de empréstimos do governo para liquidar suas debêntures.

O resultado de seu programa deve ser escrito na saída padrão.

Exemplo de entrada	**Exemplo de saída**
3 3 1 1 1 1 2 1 2 3 2 3 1 3 3 3 1 1 1 1 2 1 2 3 2 3 1 4 3 3 1 1 1 1 2 2 2 3 2 3 1 2 0 0	S N S

Problema H

Handebol

Arquivo: `handebol.[c|cpp|java]`

Frustrado e desanimado com os resultados de sua equipe de futebol, o Super Brasileiro Clube (SBC) resolveu investir na equipe de handebol. Para melhor avaliar os atletas, os técnicos identificaram que seria útil analisar a regularidade dos jogadores. Especificamente, eles estão interessados em saber quantos jogadores fizeram gols em todas as partidas.

Como o volume de dados é muito grande, eles gostariam de ter um programa de computador para realizar essa contagem.

Entrada

A primeira linha da entrada contém dois inteiros N e M ($1 \leq N \leq 100$ e $1 \leq M \leq 100$), indicando respectivamente o número de jogadores e o número de partidas. Cada uma das N linhas seguintes descreve o desempenho de um jogador: a i-ésima linha contém M inteiros X_j ($0 \leq X_j \leq 100$, para $1 \leq j \leq M$), informando o número de gols do i-ésimo jogador em cada partida.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, o número de jogadores que fizeram gols em todas as partidas.

Exemplos

Entrada	**Saída**
5 3 0 0 0 1 0 5 0 0 0 0 1 2 1 1 0	0

Entrada	**Saída**
12 5 4 4 2 3 7 0 0 0 1 0 7 4 7 0 6 1 2 3 3 2 0 0 0 0 0 4 0 9 10 10 0 1 0 0 0 1 2 0 2 3 10 10 10 1 0 0 3 3 3 4 10 10 0 10 10 1 1 2 0 9	2

Problema A
Circuito Bioquímico Digital

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

Um circuito bioquímico digital (CBD) é um artefato composto de um conjunto de *pontos de processamento*. Cada ponto de processamento é constituído por um minúsculo receptáculo para reagentes bioquímicos, feito de um substrato biológico que se comporta como um micro-circuito eletrônico digital. Dependendo do estado da reação no receptáculo, o substrato gera dois níveis de voltagem. Um leitor acoplado ao CBD é capaz de realizar a leitura de todos os pontos de processamento de um CDB num dado instante, interpretando os dois níveis de voltagem como 0 ou 1.

Um experimento com o CBD é realizado da seguinte maneira. Os pontos de processamento são carregados com as substâncias de interesse e reagentes apropriados e, a cada intervalo fixo de tempo (tipicamente alguns milisegundos), os pontos de processamento são lidos. Assim, o experimento resulta em uma seqüência de conjuntos (vetores) de bits, cada vetor correspondendo a uma medição do CBD.

Uma seqüência ininterrupta de bits 1 de um mesmo ponto de processamento ao longo do tempo é denominada de *palito*. O *comprimento* de um palito é o número de bits 1 que o compõe (note que o comprimento dos palitos de um experimento pode variar entre um e o número de medições efetuadas). Uma característica importante de um experimento com o CBD é a quantidade e o comprimento dos palitos gerados. A figura abaixo mostra o resultado de um experimento realizado com um CBD de seis pontos de processamento, em que foram efetuadas quatro medições, contendo três palitos de comprimento um, um palito de comprimento dois e um palito de comprimento quatro.

```
0 1 0 1 1 0
0 0 0 1 0 0
0 1 0 1 0 1
0 1 0 1 0 0
```

Você foi contratado para escrever um programa que determine, dado o resultado de um experimento, quantos palitos de comprimento igual ou maior do que um certo valor foram gerados.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém três inteiros P, N e C que indicam respectivamente o número de pontos de processamento ($1 \leq P \leq 1000$), o número de medições efetuadas ($1 \leq N \leq 1000$) e o comprimento mínimo de palitos de interesse ($1 \leq C \leq N$). Cada uma das próximas N linhas contém seqüências de P dígitos $\{0, 1\}$, separados por um espaço em branco. O final da entrada é indicado por $P = N = C = 0$.

A entrada deve ser lida da entrada padrão.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma única linha da saída, contendo o número de palitos de comprimento maior ou igual a C produzidos pelo experimento.

A saída deve ser escrita na saída padrão.

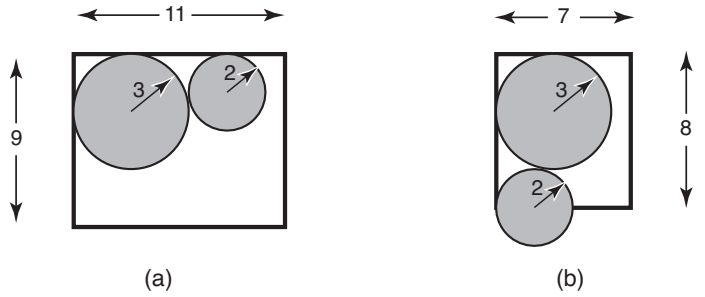
Exemplo de entrada	**Saída para o exemplo de entrada**
2 2 2 1 1 1 1 4 5 3 0 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0	2 2

Problema E
Elevador

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

A FCC (Fábrica de Cilindros de Carbono) fabrica de vários tipos de cilindros de carbono. A FCC está instalada no décimo andar de um prédio, e utiliza os vários elevadores do prédio para transportar os cilindros. Por questão de segurança, os cilindros devem ser transportados na posição vertical; como são pesados, no máximo dois cilindros podem ser transportados em uma única viagem de elevador. Os elevadores têm formato de paralelepípedo e sempre têm altura maior que a altura dos cilindros.

Para minimizar o número de viagens de elevador para transportar os cilindros, a FCC quer, sempre que possível, colocar dois cilindros no elevador. A figura abaixo ilustra, esquematicamente (vista superior), um caso em que é isto possível (a), e um caso em que isto não é possível (b):



Como existe uma quantidade muito grande de elevadores e de tipos de cilindros, a FCC quer que você escreva um programa que, dadas as dimensões do elevador e dos dois cilindros, determine se é possível colocar os dois cilindros no elevador.

Entrada

A entrada contém vários casos de teste. A primeira e única linha de cada caso de teste contém quatro números inteiros L, C, R_1 e R_2, separados por espaços em branco, indicando respectivamente a largura do elevador ($1 \leq L \leq 100$), o comprimento do elevador ($1 \leq C \leq 100$), e os raios dos cilindros ($1 \leq R_1, R_2 \leq 100$).

O último caso de teste é seguido por uma linha que contém quatro zeros separados por espaços em branco.

Saída

Para cada caso de teste, o seu programa deve imprimir uma única linha com um único caractere: '`S`' se for possível colocar os dois cilindros no elevador e '`N`' caso contrário.

Exemplo de entrada	**Exemplo de saída**

`11 9 2 3`
`7 8 3 2`
`10 15 3 7`
`8 9 3 2`
`0 0 0 0`
	`S`
	`N`
	`N`
	`S`

Problema F
Loop Musical

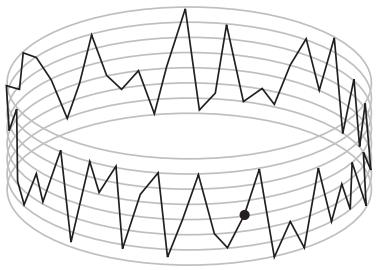
Nome do arquivo fonte: `loop.c`, `loop.cpp` *ou* `loop.java`

Um loop musical é um trecho de música que foi composto para repetir continuamente (ou seja, o trecho inicia novamente toda vez que chega ao final), sem que se note descontinuidade. Loops são muito usados na sonorização de jogos, especialmente jogos casuais pela internet.

Loops podem ser digitalizados por exemplo utilizando PCM. PCM, do inglês *Pulse Code Modulation*, é uma técnica para representação de sinais analógicos, muito utilizada em áudio digital. Nessa técnica, a magnitude do sinal é amostrada a intervalos regulares de tempo, e os valores amostrados são armazenados em sequência. Para reproduzir a forma de onda amostrada, o processo é invertido (demodulação).

Fernandinha trabalha para uma empresa que desenvolve jogos e compôs um bonito loop musical, codificando-o em PCM. Analisando a forma de onda do seu loop em um software de edição de áudio, Fernandinha ficou curiosa ao notar a quantidade de "picos" existentes. Um *pico* em uma forma de onda é um valor de uma amostra que representa um máximo ou mínimo local, ou seja, um ponto de inflexão da forma de onda. A figura abaixo ilustra (a) um exemplo de forma de onda e (b) o loop formado com essa forma de onda, contendo 48 picos.

(a) Uma forma de onda



(b) Mesma forma de onda em loop

Fernandinha é uma amiga muito querida e pediu sua ajuda para determinar quantos picos existem no seu loop musical.

Entrada

A entrada contém vários casos de teste. A primeira linha de um caso de teste contém um inteiro N, representando o número de amostras no loop musical de Fernandinha ($2 \leq N \leq 10^4$). A segunda linha contém N inteiros H_i, separados por espaços, representando a sequência de magnitudes das amostras ($-10^4 \leq H_i \leq 10^4$ para $1 \leq i \leq N$, $H_1 \neq H_N$ e $H_i \neq H_{i+1}$ para $1 \leq i < N$). Note que H_1 segue H_N quando o loop é reproduzido.

O final da entrada é indicado por uma linha que contém apenas o número zero.

Os dados devem ser lidos da entrada padrão.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha, contendo apenas um inteiro, o número de picos existentes no loop musical de Fernandinha.

O resultado de seu programa deve ser escrito na saída padrão.

Exemplo de entrada	**Exemplo de saída**
2 1 -3 6 40 0 -41 0 41 42 4 300 450 449 450 0	2 2 4

Problema A
Revisão de Contrato

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

Durante anos, todos os contratos da Associação de Contratos da Modernolândia (ACM) foram datilografados em uma velha máquina de datilografia.

Recentemente Sr. Miranda, um dos contadores da ACM, percebeu que a máquina apresentava falha em um, e apenas um, dos dígitos numéricos. Mais especificamente, o dígito falho, quando datilografado, não é impresso na folha, como se a tecla correspondente não tivesse sido pressionada. Ele percebeu que isso poderia ter alterado os valores numéricos representados nos contratos e, preocupado com a contabilidade, quer saber, a partir dos valores originais negociados nos contratos, que ele mantinha em anotações manuscritas, quais os valores de fato representados nos contratos. Por exemplo, se a máquina apresenta falha no dígito 5, o valor 1500 seria datilografado no contrato como 100, pois o 5 não seria impresso. Note que o Sr. Miranda quer saber o *valor numérico* representado no contrato, ou seja, nessa mesma máquina, o número 5000 corresponde ao valor numérico 0, e não 000 (como ele de fato aparece impresso).

Entrada

A entrada consiste de diversos casos de teste, cada um em uma linha. Cada linha contém dois inteiros D e N ($1 \leq D \leq 9$, $1 \leq N < 10^{100}$), representando, respectivamente, o dígito que está apresentando problema na máquina e o número que foi negociado originalmente no contrato (que podem ser grande, pois Modernolândia tem sido acometida por hiperinflação nas últimas décadas). O último caso de teste é seguido por uma linha que contém apenas dois zeros separados por espaços em branco.

Saída

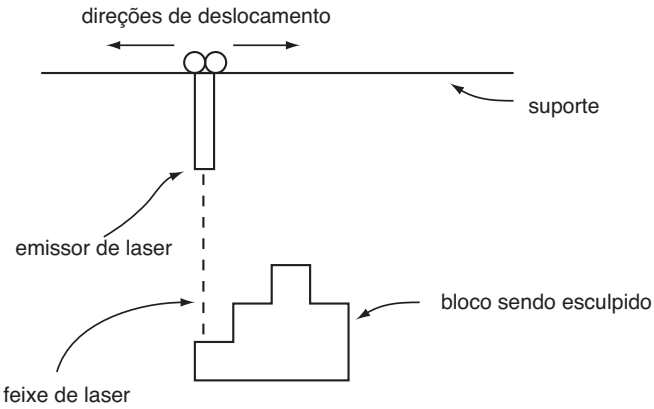
Para cada caso de teste da entrada o seu programa deve imprimir uma linha contendo um único inteiro V, o valor numérico representado de fato no contrato.

Exemplo de entrada	**Exemplo de saída**
5 5000000 3 123456 9 23454324543423 9 99999999991999999 7 777 0 0	0 12456 23454324543423 1 0

Problema H
Escultura a Laser

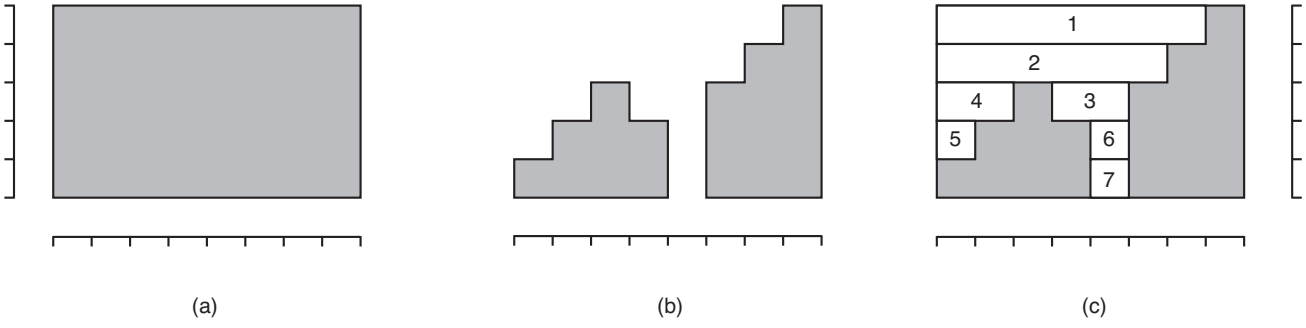
Nome do arquivo fonte: `laser.c`, `laser.cpp` *ou* `laser.java`

Desde a sua invenção, em 1958, os raios laser têm sido utilizados em uma imensa variedade de aplicações, como equipamentos eletrônicos, instrumentos cirúrgicos, armamentos, e muito mais.



A figura acima mostra um diagrama esquemático de um equipamento para esculpir, com laser, um bloco de material maciço. Na figura vemos um emissor laser que se desloca horizontalmente para a direita e para a esquerda com velocidade constante. Quando o emissor é ligado durante o deslocamento, uma camada de espessura constante é removida do bloco, sendo vaporizada pelo laser.

A figura abaixo ilustra o processo de escultura a laser, mostrando um exemplo de (a) um bloco, com 5 mm de altura por 8 mm de comprimento, no início do processo, (b) o formato que se deseja que o bloco esculpido tenha, e (c) a sequência de remoção das camadas do bloco durante o processo, considerando que a cada varredura uma camada de espessura de 1 mm é removida. Na primeira varredura, o pedaço numerado como 1 é removido; na segunda varredura, o pedaço numerado como 2 é removido, e assim por diante. Durante o processo de remoção, o laser foi ligado um total de 7 vezes, uma vez para cada pedaço de bloco removido.



Escreva um programa que, dados a altura do bloco, o comprimento do bloco, e a forma final que o bloco deve ter, determine o número total vezes de que o laser deve ser ligado para esculpir o bloco.

Entrada

A entrada contém vários casos de teste. Cada caso de teste é composto por duas linhas. A primeira linha de um caso de teste contém dois números inteiros A e C, separados por um espaço em branco, indicando respectivamente a altura ($1 \leq A \leq 10^4$) e o comprimento ($1 \leq C \leq 10^4$) do bloco a ser esculpido, em milímetros. A segunda linha contém C números inteiros X_i, cada um indicando a altura final, em milímetros, do bloco entre as posições i e $i + 1$ ao longo do comprimento ($0 \leq X_i \leq A$, para $0 \leq i \leq C - 1$). Considere que a cada varredura uma camada de espessura 1 milímetro é removida do bloco ao longo dos pontos onde o laser está ligado.

O final da entrada é indicado por uma linha que contém apenas dois zeros, separados por um espaço em branco.

Os dados devem ser lidos da entrada padrão.

Saída

Para cada caso de teste da entrada seu programa deve imprimir uma única linha, contendo um numero inteiro, indicando o número de vezes que o laser deve ser ligado para esculpir o bloco na forma indicada.

O resultado de seu programa deve ser escrito na saída padrão.

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

Problema H
Plágio Musical

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

As notas musicais são unidades básicas da música ocidental tradicional. Cada nota está associada a uma frequência. Duas notas musicais cujas frequências fundamentais tenham uma relação de potência de 2 (uma metade da outra, uma duas vezes a outra, etc.) são percebidas como muito similar. Por isso, todas as notas com esse tipo de relação recebem o mesmo nome, como descrito a seguir.

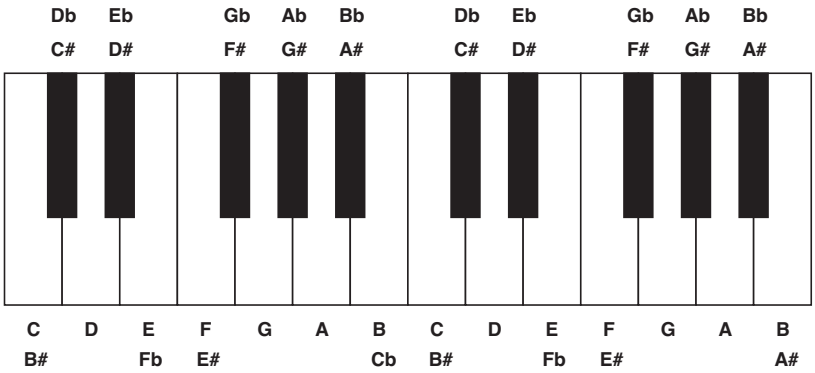
Há doze notas básicas, em uma sequência crescente de frequências, cada nota separada da anterior por uma mesma distância na escala musical (essa distância é chamada de *meio-tom*). Sete dessas doze notas são representadas por letras do alfabeto (A, B, C, D, E, F e G). A tabela abaixo mostra a distância, em meio-tons, entre essas notas.

Notas	A-B	B-C	C-D	D-E	E-F	F-G	G-A
Número de meios-tons	2	1	2	2	1	2	2

Note que há cinco notas que não são representadas pelas letras do alfabeto: as que estão entre A e B, entre C e D, entre D e E, entre F e G e entre G e A.

As notas podem ser modificadas por duas *alterações cromáticas*: *sustenido* e *bemol*, representadas respectivamente pelos símbolos '`#`' e '`b`'. Sustenido altera a nota em meio tom para cima, e bemol altera a nota em meio tom para baixo. Uma nota com alteração cromática é denotada pelo nome da nota seguida pelo símbolo da alteração. Note que com esse esquema conseguimos representar todas as doze notas.

A figura abaixo ilustra o nome das notas, segundo o esquema descrito acima, em um trecho de teclado de piano.



Uma melodia pode ser representada por uma sequência de *notas musicais*. Por exemplo,

```
A A D C# C# D E E E F# A D G# A
```

é uma melodia muito conhecida. Note no entanto que, como as distâncias entre os meios-tons são sempre iguais, a mesma melodia pode ser escrita iniciando em outra nota (dizemos que a melodia está em outro tom):

```
B B E D# D# E Gb Gb Gb G# B E A# B
```

Sua vizinha é uma famosa compositora que suspeita que tenham plagiado uma de suas músicas. Ela pediu a sua ajuda para escrever um programa que, dada a sequência de notas da melodia de sua música, e a sequência de notas de um trecho de melodia suspeito, verifique se o trecho supeito ocorre, em algum tom, na música dada.

Entrada

A entrada é composta por vários casos de teste. A primeira linha de um caso de teste contém dois inteiros M e T ($1 \leq M \leq 100000$, $1 \leq T \leq 10000$, $T \leq M$), indicando respectivamente o número de notas da música e do trecho suspeito de ter sido plagiado. As duas linhas seguintes contém M e T notas, respectivamente, indicando as notas da música e do trecho suspeito.

As notas em cada linha são separadas por espaço; cada nota é uma dentre '`A`', '`B`', '`C`', '`D`', '`E`', '`F`' ou '`G`', possivelmente seguida de um modificador: '`#`' para um sustenido ou '`b`' para um bemol.

O último caso de teste é seguido por uma linha que contém apenas dois números zero separados por um espaço em branco.

Saída

Para cada caso de teste, imprima uma única linha contendo um caractere: '`S`' caso o trecho realmente tenha sido plagiado pela música ou '`N`' caso contrário.

Exemplo de entrada	**Exemplo de saída**
16 4 D G A B C D G G G C D E F# G C C G G C D 12 2 C C# D D# E F F# G G# A A# B C D 12 2 C Db D Eb E F Gb G Ab A Bb B C D 4 3 C E G Bb D F# A 0 0	S N N S

Problema C

Tomadas

Finalmente, o time da Universidade conseguiu a classificação para a Final Nacional da Maratona de Programação da SBC. Os três membros do time e o técnico estão ansiosos para bem representar a Universidade, e além de treinar muito, preparam com todos os detalhes a sua viagem a São Paulo, onde será realizada a Final Nacional.

Eles planejam levar na viagem todos os seus vários equipamentos eletrônicos: celular, tablet, notebook, ponto de acesso wifi, câmeras, etc, e sabem que necessitarão de várias tomadas de energia para conectar todos esses equipamentos. Eles foram informados de que ficarão os quatro no mesmo quarto de hotel, mas já foram alertados de que em cada quarto há apenas uma tomada de energia disponível.

Precavidos, os três membros do time e o técnico compraram cada um uma régua de tomadas, permitindo assim ligar vários aparelhos na única tomada do quarto de hotel; eles também podem ligar uma régua em outra para aumentar ainda mais o número de tomadas disponíveis. No entanto, como as réguas têm muitas tomadas, eles pediram para você escrever um programa que, dado o número de tomadas em cada régua, determine o número máximo de aparelhos que podem ser conectados à energia num mesmo instante.

Entrada

A entrada consiste de uma linha com quatro números inteiros T_1, T_2, T_3, T_4, indicando o número de tomadas de cada uma das quatro réguas ($2 \leq T_i \leq 6$).

Saída

Seu programa deve produzir uma única linha contendo um único número inteiro, indicando o número máximo de aparelhos que podem ser conectados à energia num mesmo instante.

Exemplos

Entrada	**Saída**
2 4 3 2	8

Entrada	**Saída**
6 6 6 6	21

Entrada	**Saída**
2 2 2 2	5

Problema J

Jogos olímpicos

Um grupo de investidores está pensando em investir pesado em atletas da delegação brasileira após as olimpíadas do Rio. Para isso, eles vêm observando N atletas e perceberam que alguns estão em decadência e outros em ascensão. Em especial, o grupo está de olho em dois fatores sobre cada atleta: seu cansaço e sua habilidade. Eles anotaram os valores de habilidade e cansaço de cada atleta logo ao final das olimpíadas de 2016. Em seguida, o grupo estimou a taxa com a qual cada atleta perde ou ganha habilidade e a taxa com a qual cada atleta se cansa ao longo do tempo, e percebeu que essas taxas são constantes para os dois atributos.

Os investidores perceberam que esses dados lhes permitem definir o que resolveram chamar de atleta de ouro: um atleta que, em um determinado período de tempo, é o atleta menos cansado e o mais habilidoso. Ficou decidido que investimentos serão feitos apenas em atletas de ouro. Descubra quantos jogadores, entre os observados inicialmente, receberão algum investimento. Considere que o tempo $t = 0$ é o tempo das olimpíadas do Rio: nenhum atleta que foi de ouro antes desse tempo pode receber investimento. Considere também que qualquer tempo após as olimpíadas do Rio deve ser considerado, por maior que seja. Um atleta que é de ouro exatamente no tempo $t = 0$ deve ser contado.

Entrada

A primeira linha da entrada contém um inteiro, N ($1 \leq N \leq 10^5$), o número de atletas. Seguem N linhas, cada uma com quatro números inteiros: H_i, H_i^t, C_i, C_i^t ($-10^6 < H_i, H_i^t, C_i, C_i^t \leq 10^6$, $H_i^t, C_i^t \neq 0$), representando, respectivamente, a habilidade ao final das olimpíadas, a taxa de variação da habilidade, o cansaço ao final das olimpíadas e a taxa de variação do cansaço do i-ésimo atleta.

Saída

Seu programa deve produzir uma única linha com um inteiro O, representando o número de atletas que receberão algum investimento do grupo.

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

Exemplo de entrada 2	**Exemplo de saída 2**
6 1 10 5 8 8 7 12 -5 10 -2 -3 8 -3 -5 -8 -12 0 1 10 2 8 3 9 -3	0

Problema A

Andando no tempo

Imagine que você tenha uma máquina do tempo que pode ser usada no máximo três vezes, e a cada uso da máquina você pode escolher voltar para o passado ou ir para o futuro. A máquina possui três créditos fixos; cada crédito representa uma certa quantidade de anos, e pode ser usado para ir essa quantidade de anos para o passado ou para o futuro. Você pode fazer uma, duas ou três viagens, e cada um desses três créditos pode ser usado uma vez apenas. Por exemplo, se os créditos forem $5, 12$ e 9, você poderia decidir fazer duas viagens: ir 5 anos para o futuro e, depois, voltar 9 anos para o passado. Dessa forma, você terminaria quatro anos no passado, em 2012. Também poderia fazer três viagens, todas indo para o futuro, usando os créditos em qualquer ordem, terminando em 2042.

Neste problema, dados os valores dos três créditos da máquina, seu programa deve dizer se é ou não possível viajar no tempo e voltar para o presente, fazendo pelo menos uma viagem e, no máximo, três viagens; sempre usando cada um dos três créditos no máximo uma vez.

Entrada

A entrada consiste de uma linha contendo os valores dos três créditos A, B e C ($1 \leq A, B, C \leq 1000$).

Saída

Seu programa deve imprimir uma linha contendo o caracter "`S`" se é possível viajar e voltar para o presente, ou "`N`" caso contrário.

Exemplo de entrada 1	**Exemplo de saída 1**
`22 5 22`	`S`

Exemplo de entrada 2	**Exemplo de saída 2**
`31 110 79`	`S`

Exemplo de entrada 3	**Exemplo de saída 3**
`45 8 7`	`N`