acm International Collegiate Programming Contest 2014 IBM event sponsor

Maratona de Programação da SBC 2014

Sub-Regional Brasil do ACM ICPC

13 de Setembro de 2014

Caderno de Problemas

Informações Gerais

Este caderno contém 11 problemas; as páginas estão numeradas de 1 a 15, não contando esta página de rosto. Verifique se o caderno está completo.

A) Sobre a entrada

1) A entrada de seu programa deve ser lida da *entrada padrão*.
2) A entrada é composta de um único caso de teste, descrito em um número de linhas que depende do problema.
3) Quando uma linha da entrada contém vários valores, estes são separados por um único espaço em branco; a entrada não contém nenhum outro espaço em branco.
4) Cada linha, incluindo a última, contém exatamente um caractere final-de-linha.
5) O final da entrada coincide com o final do arquivo.

B) Sobre a saída

1) A saída de seu programa deve ser escrita na *saída padrão*.
2) Quando uma linha da saída contém vários valores, estes devem ser separados por um único espaço em branco; a saída não deve conter nenhum outro espaço em branco.
3) Cada linha, incluindo a última, deve conter exatamente um caractere final-de-linha.

Promoção:

Sociedade Brasileira de Computação

Patrocínio:

Fundação Carlos Chagas

v1.0

Problema A

Volta

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

No automobilismo é bastante comum que o líder de uma prova, em determinado momento, ultrapasse o último colocado. O líder, neste momento, está uma volta à frente do último colocado, que se torna, assim, um retardatário. Neste problema, dados os tempos que o piloto mais rápido e o piloto mais lento levam para completar uma volta, você deve determinar em que volta o último colocado se tornará um retardatário, ou seja, será ultrapassado pelo líder. Você deve considerar que, inicialmente, eles estão lado a lado, na linha de partida do circuito, ambos no início da volta de número 1 (a primeira volta da corrida); e que uma nova volta se inicia sempre depois que o líder cruza a linha de partida.

Entrada

A única linha da entrada contém dois números inteiros X e Y ($1 \leq X < Y \leq 10000$), os tempos, em segundos, que o piloto mais rápido e o piloto mais lento levam para completar uma volta, respectivamente.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro: a volta em que o piloto mais lento se tornará um retardatário.

Exemplos

Entrada	**Saída**
`1 10`	`2`

Entrada	**Saída**
`4 8`	`2`

Entrada	**Saída**
`5 7`	`4`

Entrada	**Saída**
`6875 7109`	`31`

Problema B

Baralho Embaralhado

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

Um baralho contém um número par $2n$ de cartas $a_1, a_2, \ldots, a_{2n}$, todas distintas ($a_1 < a_2 < \cdots < a_{2n}$). O baralho encontra-se perfeitamente ordenado, ou seja, a primeira carta é a_1, a segunda carta é a_2, e assim por diante, até a última carta, que é a_{2n}.

Um croupier então executa repetidamente um procedimento de embaralhar, que consiste de dois passos:

1. o baralho é dividido ao meio;
2. as cartas das duas metades são então intercaladas, de maneira que se a sequência de cartas do baralho no início do passo 1 é $x_1, x_2, \ldots, x_{2n}$, então ao final do passo 2 a sequência de cartas se torna $x_{n+1}, x_1, x_{n+2}, x_2, \ldots, x_{2n}, x_n$.

Dado o número de cartas do baralho, escreva um programa que determine quantas vezes o procedimento de embaralhar descrito acima deve ser repetido de forma que o baralho volte a ficar ordenado.

Entrada

A única linha da entrada contém um inteiro par P ($2 \leq P \leq 2 \times 10^5$), indicando o número de cartas do baralho (note que o valor P corresponde ao valor $2n$ na descrição acima).

Saída

Seu programa deve produzir uma única linha contendo um único inteiro, o número mínimo de vezes que o processo de embaralhamento deve ser repetido para que o baralho fique novamente ordenado.

Exemplos

Entrada	**Saída**
4	4

Entrada	**Saída**
6	3

Entrada	**Saída**
2	2

Entrada	**Saída**
100002	100002

Problema C

Confederação

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

A Confederação Galática resolveu fazer uma reforma administrativa, para melhor distribuir os recursos de sua frota. Para isso, ela dividiu todo o espaço em regiões. Para definir as regiões, inicialmente um conjunto de planos infinitos foi especificado, e as regiões foram definidas pelos cortes desses planos. Note que algumas regiões são ilimitadas, mas que também podem existir regiões limitadas. O conjunto de planos foi escolhido de tal maneira que nenhum dos planos intercepta a órbita de um planeta, e portanto cada planeta transita por apenas uma região durante sua órbita (ou seja, um planeta dentro de uma região nunca cruzará um plano para outra região).

Sua tarefa consiste em determinar, dadas as equações dos planos e as posições dos planetas, quantos planetas existem na região com o maior número de planetas (em outras palavras, qual o número máximo de planetas dentro de uma região).

Entrada

A primeira linha da entrada contém dois inteiros M ($1 \leq M \leq 500$) e N ($1 \leq N \leq 10000$), indicando respectivamente o número de planos e número de planetas. As M linhas seguintes contêm cada uma quatro inteiros A, B, C e D ($-10000 \leq A, B, C, D \leq 10000$), os coeficientes e o termo livre da equação $Ax + By + Cz = D$ que define cada um dos planos. A seguir, cada uma das N linhas seguintes contém três inteiros X, Y e Z ($-10000 \leq X, Y, Z \leq 10000$), indicando a posição (X, Y, Z) de um planeta.

Saída

Seu programa deve produzir uma única linha contendo apenas um número inteiro, o número de planetas na região que contém o maior número de planetas.

Exemplos

Entrada	**Saída**
2 5 1 0 0 1 2 0 0 8 0 1 0 2 2 2 3 3 3 5 5 5 2 18 4	3

Entrada	**Saída**
4 8 0 0 1 1 1 0 1 2 -1 1 1 3 -1 -1 1 3 0 0 5 0 0 4 0 0 -2 1 0 5 40 19 104 13 26 84 89 -45 18 3 1 0	5

Problema D

Dona Minhoca

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

Dona Minhoca fica furiosa quando ouve as pessoas dizerem que minhocas são bichos palíndromes, nos quais não é possível distinguir a cabeça do rabo. Que infâmia!

Dona Minhoca vive em uma linda caverna, composta de salões e túneis. Cada túnel liga dois salões distintos e pode ser usado nas duas direções. Um "ciclo" na caverna é uma sequência de salões $s_1, s_2, \ldots, s_n, s_{n+1} = s_1$, tais que $s_i \neq s_{i+1}$ e (s_i, s_{i+1}) é um túnel, para $1 \leq i \leq n$. A caverna de Dona Minhoca pode conter ciclos, mas cada salão faz parte de no máximo um ciclo da caverna. Os túneis e salões são estreitos, de forma que se uma parte do corpo de Dona Minhoca ocupa um túnel ou salão, não há espaço para Dona Minhoca entrar novamente por esse túnel ou salão.

Alguns salões da caverna têm acesso a partir da superfície. Dona Minhoca tem um mapa que descreve a caverna, informando para cada túnel o seu comprimento e quais dois salões o túnel liga. Dona Minhoca também é vaidosa e conhece o seu próprio comprimento.

Dona Minhoca quer saber, para os salões que têm acesso à superfície, se é possível entrar na caverna pelo salão, percorrer a menor distância possível dentro da caverna, e sair novamente pelo mesmo salão que entrou, sempre andando para a frente, sem nunca dar marcha-a-ré. Você pode ajudá-la?

Entrada

A primeira linha contém dois inteiros S ($2 \leq S \leq 10^4$) e T ($1 \leq T \leq 2S$) representando respectivamente o número de salões e o número de túneis da caverna. Os salões são identificados por inteiros de 1 a S. Cada uma das T linhas seguintes descreve um túnel e contém três inteiros A, B e C ($1 \leq A < B \leq S; 1 \leq C \leq 100$), onde A e B representam os salões ligados pelo túnel, e C representa o comprimento do túnel. Um salão é ligado por túneis a no máximo outros 100 salões e cada dois salões são ligados por no máximo um túnel. A próxima linha contém um inteiro Q ($1 \leq Q \leq 100$), que indica o número de consultas. Cada uma das Q linhas seguintes descreve uma consulta, e contém dois inteiros X ($1 \leq X \leq S$) e M ($1 \leq M \leq 10^5$), que indicam respectivamente o salão pelo qual Dona Minhoca quer entrar e o comprimento de Dona Minhoca.

Saída

Para cada consulta da entrada seu programa deve produzir apenas uma linha, contendo apenas um número inteiro, o comprimento do percurso mínimo que Dona Minhoca deve percorrer dentro da caverna para entrar e sair pelo salão indicado na consulta, sem dar marcha-a-ré. Se não for possível para Dona Minhoca entrar e sair sem dar marcha-a-ré, a linha deve conter o valor -1.

Exemplos

Entrada	**Saída**
4 4 1 2 12 2 3 10 3 4 8 2 4 5 3 1 23 4 10 1 24	47 23 -1

Entrada	**Saída**
8 9 1 2 1 2 3 1 3 4 1 2 5 10 5 6 25 2 6 20 3 7 9 7 8 3 3 8 4 4 1 10 4 60 8 5 7 55	20 -1 16 71

Problema E

Ecologia

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

O reino da Poliminogônia passou recentemente uma lei ecológica que obriga todas as fazendas a preservar o máximo de árvores possível em uma porcentagem fixa da área da fazenda. Além disso, para que os animais silvestres possam se movimentar livremente, a área preservada deve ser conexa.

As fazendas na Poliminogônia são sempre um reticulado de $N \times N$ quadrados de um hectare cada. A figura ao lado ilustra uma fazenda com $N = 5$. A área preservada deve cobrir exatamente M quadrados. No exemplo da figura, $M = 6$. Ela deve ser conexa ortogonalmente; quer dizer, tem que ser possível se movimentar entre quaisquer dois quadrados preservados apenas com movimentos ortogonais entre quadrados preservados. A área não preservada, entretanto, pode ser desconexa.

31	12	7	1	14
23	98	3	87	1
5	31	8	2	99
12	3	42	17	88
120	2	7	5	7

Os fazendeiros sabem o número de árvores que há dentro de cada quadrado e você deve escrever um programa que calcule o número máximo possível de árvores que podem ser preservadas com uma área de M quadrados. No exemplo, é possível preservar 377 árvores!

Entrada

A primeira linha da entrada contém dois inteiros N e M ($2 \leq N \leq 50$, $1 \leq M \leq 10$). As N linhas seguintes contêm, cada uma, N inteiros de valor entre 1 e 1000, representando o número de árvores dentro de cada quadrado da fazenda.

Saída

Seu programa deve imprimir uma linha contendo um número inteiro, o número máximo de árvores que podem ser preservadas, com as restrições dadas.

Exemplos

Entrada	**Saída**
5 6 31 12 7 1 14 23 98 3 87 1 5 31 8 2 99 12 3 42 17 88 120 2 7 5 7	377

Entrada	**Saída**
4 8 1 1 1 1 9 9 9 1 9 1 9 1 9 9 9 1	72

Problema F

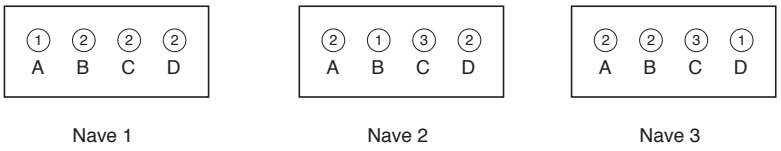
Teletransporte

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

A Confederação Galática instalou um novo sistema de teletransporte em suas naves espaciais. Cada nave recebeu uma cabine de teletransporte, na qual há um painel com quatro botões. Cada botão é rotulado com uma letra diferente A, B, C ou D e com um número que indica a nave destino para a qual o usuário será transportado, instantaneamente, se o respectivo botão for pressionado (como todos sabem, as naves da Confederação são identificadas por inteiros de 1 a N).

Para usar o sistema, o usuário deve adquirir um bilhete para cada viagem que deseja realizar (uma viagem corresponde a pressionar um botão). Note que como o número botões no painel é pequeno comparado com o número de naves da Confederação, pode ser necessário que o usuário tenha que comprar um bilhete múltiplo de L viagens para ir de uma dada nave S para uma outra nave T.

Por exemplo, para as naves da figura abaixo, se o usuário está na cabine de teletransporte da nave 3 e pressiona o botão B ele é transportado para a nave 2. Se ele tem um bilhete múltiplo e pressiona novamente o botão B ele é então transportado para a nave 1.



Sua tarefa neste problema é, dados a nave de partida S, a nave de chegada T e o número de viagens L do bilhete, determinar quantas sequências distintas de L botões levam o usuário da nave S para a nave T. Por exemplo, para as naves da figura acima, existem quatro sequências distintas de $L = 2$ botões que levam um usuário da nave $S = 3$ para a nave $T = 1$: CD, DA, AB, e BB.

Entrada

A primeira linha da entrada contém dois inteiros N ($1 \leq N \leq 100$) e L ($0 \leq L < 2^{30}$), indicando respectivamente o número de naves e o número de viagens do bilhete. A segunda linha da entrada contém dois inteiros S e T ($1 \leq S, T \leq N$), indicando respectivamente a nave de partida e a nave de chegada. Cada uma das N linhas seguintes descreve o painel da cabine de teletransporte de uma nave. A i-ésima dessas linhas, $1 \leq i \leq N$, contém quatro inteiros A, B, C e D ($1 \leq A, B, C, D \leq N$), que representam os números escritos nos quatro botões da cabine de teletransporte da nave de número i.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro, que deve ser igual a r módulo 10^4, onde r é o número de sequências distintas de L botões que levam o usuário da nave S para a nave T.

Exemplos

Entrada	**Saída**
2 20 1 1 2 2 2 2 1 1 1 1	7776

Entrada	**Saída**
2 29 1 1 2 2 2 2 1 1 1 1	0

Entrada	**Saída**
2 0 1 1 2 2 2 2 1 1 1 1	1

Entrada	**Saída**
2 0 1 2 2 2 2 2 1 1 1 1	0

Entrada	**Saída**
3 2 3 1 1 2 2 2 2 1 3 2 2 2 3 1	4

Problema G

Letras

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

Os parques na Cidade da Lógica são reticulados de $N \times N$ quadrados ($2 \leq N \leq 100$), onde cada quadrado contém uma das 10 primeiras letras ASCII, `abcdefghijABCDEFGHIJ`, em caixa minúscula ou maiúscula. As pessoas na Cidade da Lógica têm orgulho de seguir apenas caminhos consistentes quando cruzam os parques. Por exemplo, se eles passam por um `c` minúsculo, eles não vão se permitir, mais adiante, passar por um `C` maiúsculo. Para definir isso mais precisamente, um caminho consistente é uma sequência de quadrados satisfazendo: quadrados consecutivos na sequência são adjacentes ortogonalmente; nenhuma letra ocorre na sequência tanto minúscula quanto maiúscula. Quer dizer, ou a letra não está na sequência, ou ela ocorre apenas em caixa minúscula, ou somente em caixa maiúscula.

```
DdaAaA      D.....
CBAcca      C.....
eEaeeE      e.....
bBbabB      b.bab.
DbDdDc      DbD.D.
fFaAaC      ....aC
```

Você deve escrever um programa para ajudar as pessoas da Cidade da Lógica a computar o comprimento do menor caminho consistente entre o quadrado de coordenadas $(1, 1)$, no canto superior esquerdo, e o quadrado de coordenadas (N, N), no canto inferior direito. Por exemplo, para o parque acima, o menor caminho consistente tem comprimento 13.

Entrada

A primeira linha da entrada contém um inteiro N ($2 \leq N \leq 100$), o tamanho do parque. As N linhas seguintes contêm, cada uma, uma sequência de N letras, definindo o parque.

Saída

Seu programa deve imprimir uma linha contendo um inteiro, o comprimento de um caminho consistente mínimo. Se não houver um caminho consistente, imprima `-1`.

Exemplos

Entrada	**Saída**
6 DdaAaA CBAcca eEaeeE bBbabB DbDdDc fFaAaC	13

Entrada	**Saída**
7 aAaaaaa aAaaaAa aAaaaAA aaAaAaa AaAaaAa aaAAaAa aaaaaAa	-1

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 I
RSA

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

O algoritmo RSA é um dos algoritmos de criptografia mais utilizados e é considerado uma das alternativas mais seguras existentes. Seu funcionamento básico é descrito a seguir.

Dois números primos ímpares p e q são escolhidos e calcula-se $n = pq$. A seguir é calculada a função totiente $\phi(n) = (p-1)(q-1)$ e um inteiro e satisfazendo $1 < e < \phi(n)$ é escolhido de forma que $\text{mdc}(\phi(n), e) = 1$. Finalmente é calculado o inteiro d, o inverso multiplicativo de e módulo $\phi(n)$, ou seja, o inteiro d satisfazendo $de = 1 \pmod{\phi(n)}$.

Assim obtemos a chave pública, formada pelo par de inteiros n e e, e a chave secreta, formada pelos inteiros n e d.

Para criptografar uma mensagem m, com $0 < m < n$, calcula-se $c = m^e \pmod{n}$, e c é a mensagem criptografada. Para descriptografá-la, ou seja, para recuperar a mensagem original, basta calcular $m = c^d \pmod{n}$. Note que, para isso, a chave secreta deve ser conhecida, não sendo suficiente o conhecimento da chave pública. Note ainda que a expressão $x = 1 \pmod{y}$ usada acima equivale a dizer que y é o menor natural tal que o resto da divisão de x por y é 1.

Neste problema você deve escrever um programa para quebrar a criptografia RSA.

Entrada

A única linha da entrada contém três inteiros N, E, e C, onde $15 \leq N \leq 10^9$, $1 \leq E < N$ e $1 \leq C < N$, de forma que N e E constituem a chave pública do algoritmo RSA descrita acima e C é uma mensagem criptografada com essa chave pública.

Saída

Seu programa deve produzir uma única linha, contendo um único inteiro M, $1 \leq M < N$, a mensagem original.

Exemplos

Entrada	**Saída**
`1073 71 436`	`726`

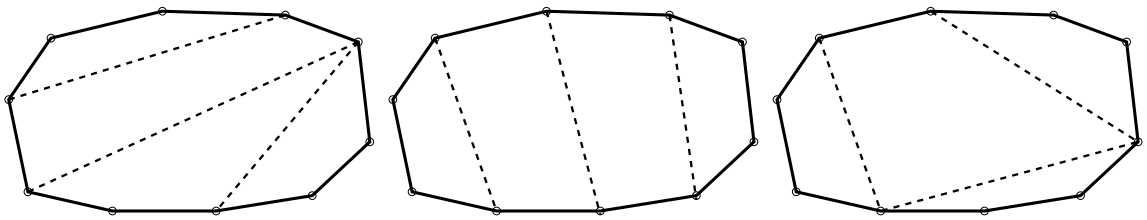
Entrada	**Saída**
`91 43 19`	`33`

Problema J

Corte

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

Todo polígono convexo, com $2N$ vértices, pode ser decomposto em $N-1$ quadriláteros, fazendo-se $N-2$ cortes em linha reta entre certos pares de vértices. A figura abaixo ilustra três diferentes decomposições do mesmo polígono com $N = 5$. O *peso* da decomposição é a soma dos comprimentos de seus $N-2$ cortes. Seu programa deve computar o peso de uma decomposição de peso mínimo!



Entrada

A primeira linha da entrada contém um inteiro N ($2 \leq N \leq 100$). As $2N$ linhas seguintes contém cada uma dois números reais X e Y ($0 \leq X, Y \leq 10000$), com precisão de 4 casas decimais: as coordenadas dos $2N$ pontos, em sentido anti-horário, do polígono convexo.

Saída

Seu programa deve imprimir uma linha contendo um número real, com precisão de 4 casas decimais. O número deve ser o peso de uma decomposição de peso mínimo do polígono dado.

Exemplos

Entrada	**Saída**
4 5715.7584 3278.6962 3870.5535 4086.7950 3823.2104 4080.7543 3574.4323 170.2905 4521.4796 144.9156 4984.6486 306.2896 5063.1061 347.1661 6099.9959 2095.9358	4519.6176

Entrada	**Saída**
2 6044.4737 2567.9978 5752.5635 3226.5140 5148.8242 3802.9292 4598.8042 4036.8000	0.0000

Problema K

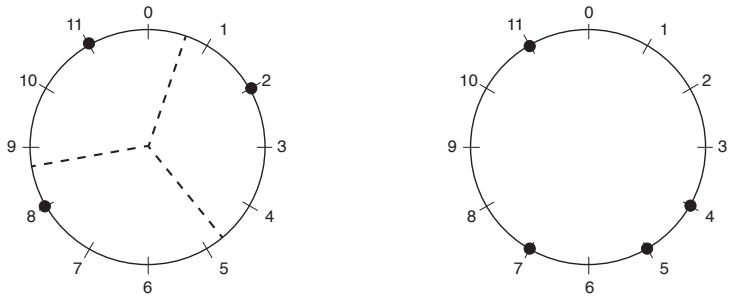
Pizza do Vô Pepe

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

Vovô Pepe é famoso por suas pizzas. Elas são deliciosas, e têm o formato de um círculo perfeito. Vovô preparou uma pizza especial para o jantar de hoje à noite, e colocou um certo número de azeitonas distribuídas aleatoriamente, mas colocadas exatamente na borda da pizza.

Sua tarefa é determinar, conhecendo a circunferência da pizza, a quantidade de azeitonas e a posição de cada azeitona, se é possível dividir a pizza em setores circulares de mesmo tamanho, de tal forma que cada pedaço de pizza contenha exatamente uma azeitona.

A figura abaixo mostra (a) uma pizza de circunferência 12 com 3 azeitonas e uma possível divisão em pedaços iguais; e (b) uma pizza de circunferência 12 com 4 azeitonas que não pode ser dividida em pedaços iguais como descrito acima. Apesar de deliciosas, as azeitonas são muito pequenas, e suas dimensões podem ser desconsideradas no cálculo da divisão.



(a) (b)

Entrada

A primeira linha contém dois inteiros C ($3 \leq C \leq 10^5$) e N ($3 \leq N \leq 10^4, N \leq C$) representando respectivamente a circunferência da pizza e o número de azeitonas. O inteiro C é múltiplo de N. A segunda linha contém N inteiros distintos X_i ($0 \leq X_1 < X_2 < \ldots < X_N < C$), em ordem crescente, descrevendo as posições das azeitonas, dadas pelo comprimento do arco circular no sentido horário, a partir de um ponto fixo da circunferência.

Saída

Seu programa deve produzir apenas uma linha, com apenas uma letra, que deve ser `S` se é possível dividir a pizza como descrito acima, ou `N` caso contrário.

Exemplos

Entrada	**Saída**
`12 3` `2 8 11`	`S`

Entrada	**Saída**
`12 4` `4 5 7 11`	`N`