XII Maratona InterFatecs

Praia Grande 2023

Fase 1 - 03/06/2023

Caderno de Problemas

Organização e Patrocínio

interfatecs.com.br

1 Instruções

Este caderno contém 11 problemas – identificados por letras de A até K, com páginas numeradas de 3 até 25. Verifique se seu caderno está completo.

Informações gerais

1. Sobre a competição
   (a) A competição possui duração de 5 horas (início as 13:00h término as 18:00h);
   (b) NÃO é permitido acesso a conteúdo da Internet ou qualquer outro meio eletrônico digital;
   (c) NÃO é permitido o uso de ferramentas de auxílio à codificação, como GitHub Copilot, Tabnine, Amazon CodeWhisperer ou similar;
   (d) É permitido somente acesso a conteúdo impresso em papel (cadernos, apostilas, livros);
   (e) Não é permitida a comunicação com o técnico ou qualquer outra pessoa que não seja a equipe para tirar dúvidas sobre a maratona
   (f) Cada equipe terá acesso a 1 computador dotado do ambiente de submissão de programas (BOCA), dos compiladores, link-editores e IDEs requeridos pelas linguagens de programação permitidas;
   (g) NÃO é permitido o uso de notebooks, smartphones, ou outro tipo de computador ou assistente pessoal;
   (h) Todos os problemas têm o mesmo valor na correção.
2. Sobre o arquivo de solução e submissão:
   (a) O arquivo de solução (o programa fonte) DEVE TER o mesmo nome que o especificado no enunciado (logo após o título do problema);
   (b) confirme se você escolheu a linguagem correta e está com o nome de arquivo correto antes de submeter a sua solução;
   (c) NÃO insira acentos ou outros caracteres especiais no arquivo-fonte.
3. Sobre a entrada
   (a) A entrada de seu programa deve ser lida da entrada padrão (não use interface gráfica);
   (b) Seu programa será testado em vários casos de teste válidos além daqueles apresentados nos exemplos. Considere que seu programa será executado uma vez para cada caso de teste.
4. Sobre a saída
   (a) A saída do seu programa deve ser escrita na saída padrão;
   (b) Não exiba qualquer outra mensagem além do especificado no enunciado.
5. Versões das linguagens
   (a) gcc version 11.3.0 (lembre-se que não existem bibliotecas do Windows)
   (b) Python 3.10.6
   (c) Java 17.0.7+7-Ubuntu-0ubuntu122.04.2 (lembre-se que as classes devem estar fora de pacotes)

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problem A

Blaittland

Source file: blaittland.{ c | cpp | java | py }
Author: Prof. Dr. Leandro Luque (Fatec Mogi das Cruzes)

In the calm Blaittland University, there exists an old library full of rarities, including a bookshelf brimming with programming language books. Every morning, these books are meticulously organized in alphabetical order from left to right. None of the books have titles that start with the same letter. However, students, always rushing towards assignment deadlines, pick books to read and place them back exactly one position to the left of its original spot, maintaining a unique semblance of disorder in the bookshelf.

Interestingly, Systems Analysis and Development students have adopted a peculiar habit: if a book has been read by more than five people during a day, they don't read it again. This unspoken rule, however, is not always respected by the Database Systems students. Another quirk that has been noticed is that no one ever picks up the book located furthest to the left to read.

At the end of the day, Jeff, the librarian, not only needs to calculate the number of times books were picked to read, but he also wants to determine if it is possible to be certain that any Database Systems student has read a book.

Input

The first line contains a single integer N ($1 \leq N \leq 26$), representing the number of books on the shelf. The second line contains a string of N uppercase letters (from A to Z), representing the final order of the books on the shelf.

Output

If it's possible to affirm (beyond a shadow of a doubt) that a Database Systems student read a book, print "A Database Systems student read a book."Otherwise, print a single integer, representing the number of times any book was picked to read. Remember to include a newline at the end of your output.

Example of Input 1	Example of Output 1
7 GCBADEF	A Database Systems student read a book.

Example of Input 2	Example of Output 2
12 DFGNMHRQSYVZ	5

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problem B

Fifteenlove

Source file: fifteenlove.{ c | cpp | java | py }
Author: Prof. Dr. Leandro Luque (Fatec Mogi das Cruzes)

A very popular sport worldwide is tennis, known for its peculiar scoring system. To beginners, this scoring might seem strange, but as you get used to the game, you start understanding the logic behind it.

A tennis match consists of sets. For instance, matches at Roland Garros, the most charming tennis tournament, may consist of up to 5 sets. The player who wins three sets first is declared the match winner.

Each set consists of multiple games. During each game, one of the players serves. The scoring for a game follows this sequence: 0 (love), 15, 30, 40, and GAME. If the game is tied at 40 (also known as "deuce"), the next player to win a point gains the advantage. If that player wins the subsequent point, they win the game. If not, the score reverts to deuce.

A set is won by the first player to reach 6 games, but they must win by a margin of 2 games. Therefore, if the score reaches 6-6, a "tie-break"is played. In the tie-break, the first player to score 7 points wins, provided they have a 2-point lead. If not, the tie-break continues until a player has a 2-point lead. Unlike points in games, points in a tie-break follow the simple numerical order of 1, 2, 3, 4, and so on. The player who served first in the tiebreak should be the one who received the serve in the previous game. Thereafter, the serve switches to the opponent and alternates every two points. For instance, let's suppose that Munarinho was serving at 6-6 and the game went into a tie-break. The tie-break will start with Munarinho's opponent serving. Then, there will be two serves by Munarinho, followed by two serves from his opponent, and so on. Additionally, the player who receive the first serve in a tie-break is the one who will serve the game after it.

That said, here's your challenge: given a record of each point in a tennis match, you should determine the current score of the game. You should consider that player 1 always starts serving and that the match follows the rules of Roland Garros.

Input

The input consists of a single test case. The first line contains the number of points $N(1 \leq N \leq 300)$. The next line contains N characters $B(B \in \{W, L\})$, in uppercase, indicating whether the player serving Won or Lost the point.

Output

The output should reflect the game score after the points specified in the input have been played. The output format is "$W(X)[U] - Y(Z)[V]$", without quotes. Here, W and Y represent the number of sets won by player 1 and 2, respectively $(0 \leq W, Y \leq 3)$. X and Z denote the number of games won by player 1 and 2 in the current set, respectively $(0 \leq X, Z \leq 7)$. The values of U and V depend on whether the game is in a tie-break. If so, the values are integers starting at 0 $(U, V \geq 0)$. If not, the possible values are $(U, V \in \{0, 15, 30, 40, ADV, GAME\})$. $GAME$ should be used only for the last game in the match. For others, after finishing, the score will be restarted to $0, 0$.

If the input match concludes, your output should display the number of games in the final set and the number of points in the final game. For example, if the score is $1(2)[15] - 2(5)[40]$ and player 2 scores, the result

should be $1(2)[15] - 3(6)[GAME]$.

Example of Input 1	Example of Output 1
4 WWLL	0(0)[30]-0(0)[30]

Example of Input 2	Example of Output 2
21 WWWWLLLLWWWWLLLLWWWWW	0(5)[0]-0(0)[15]

Example of Input 3	Example of Output 3
11 WWWLLLWLLWW	0(0)[ADV]-0(0)[40]

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema C

Pirâmide Alfabética

Arquivo fonte: piramide.{ c | cpp | java | py }

Autor: Prof. Me. Lucio Nunes de Lira (Fatec Ferraz de Vasconcelos)

Um famoso personagem, de um igualmente conhecido filme, disse a seguinte frase: "*Palavras são, na minha nada humilde opinião, nossa inesgotável fonte de magia [...]*". É claro que, para formar palavras, precisamos que exista um alfabeto com símbolos que permitam construí-las. Um exemplo é o nosso alfabeto latino.

O alfabeto latino contém vinte e seis letras, iniciando com o caractere 'A' e encerrando em 'Z', se desconsiderarmos as acentuações e as diferenças entre letras maiúsculas e minúsculas.

Hermione, uma garota muita estudiosa, percebeu que é possível desenhar usando letras do alfabeto latino. Em uma folha quadriculada de vinte e seis colunas, Hermione escreveu na 1ª linha e 26ª coluna o primeiro caractere do alfabeto. Na 2ª linha escreveu o primeiro e o segundo do alfabeto, ocupando a 25ª e a 26ª colunas, respectivamente. Na 3ª linha escreveu do 1º ao 3º caractere, preenchendo da 24ª à 26ª coluna. Com este procedimento, foi possível preencher a 26ª linha com todos os caracteres do alfabeto, em que 'A' ocupou a 1ª coluna e 'Z' ocupou a 26ª. Assim, formou-se uma "pirâmide alfabética", semelhante a um triângulo retângulo, como pode ser visualizado na Figura 1, supondo que o desenho parasse na 8ª linha.

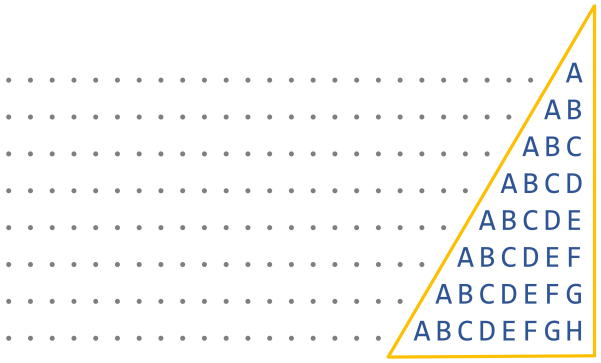


Figura C.1: Exemplo de pirâmide alfabética com oito linhas.

Hermione está ocupada estudando para uma prova de programação (que ela também considera como um tipo de magia!) e pediu sua ajuda para automatizar os desenhos das "pirâmides alfabéticas".

Entrada

Um número natural N ($1 \leqslant N \leqslant 26$) e uma *string* P ($P \in \{maiusculas, minusculas\}$) (sem acentuação e em minúsculas), indicando se a pirâmide será composta só de letras maiúsculas ou minúsculas, respectivamente.

Saída

Uma pirâmide alfabética com exatas N linhas e com letras maiúsculas ou minúsculas, conforme P, seguindo a mesma estratégia descrita no texto e ilustrada na figura e casos de teste de exemplo. Note que: (a) não há espaços entre os caracteres; (b) qualquer uma das vinte e seis colunas não ocupada por uma letra, será

preenchida com um ponto '.' (sem apóstrofos) e; (c) toda linha é encerrada com uma quebra de linha.

Exemplo de Entrada 1	**Exemplo de Saída 1**
8 maiusculas	.......A ......AB .....ABC ....ABCD ...ABCDE ..ABCDEF .ABCDEFG ABCDEFGH

Exemplo de Entrada 2	**Exemplo de Saída 2**
26 minusculas	.........a ....ab ....abc ......abcd .....abcde ....abcdef ...abcdefg ..abcdefgh .abcdefghi abcdefghij .abcdefghijk ..abcdefghijkl ...abcdefghijklm ....abcdefghijklmn .....abcdefghijklmno ......abcdefghijklmnop ....abcdefghijklmnopq ....abcdefghijklmnopqr ....abcdefghijklmnopqrs ....abcdefghijklmnopqrst ....abcdefghijklmnopqrstu ....abcdefghijklmnopqrstuv ....abcdefghijklmnopqrstuvw ....abcdefghijklmnopqrstuvwx ....abcdefghijklmnopqrstuvwxy abcdefghijklmnopqrstuvwxyz

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema D

Argasd

Arquivo fonte: argasd.{ c | cpp | java | py }

Autor: Prof. Me. Sérgio Luiz Banin (Fatec São Paulo e Fatec São Caetano do Sul)

No Reino de Argasd existe uma prática comum que consiste em caminhões circularem pelas cidades para vender botijões de gás de cozinha. Os botijões vazios são retornados para que a empresa os reutilize. Os botijões são marcados com o nome e logotipo da empresa que é sua proprietária. No passado, uma empresa só aceitava o botijão vazio de sua propriedade, porém, com o tempo, isso foi considerado abusivo pelos governantes de Argasd e ocorreu a aprovação de uma lei a respeito do assunto. Segundo essa lei, todas as empresas, ao realizar uma venda, são obrigadas a aceitar botijões vazios de qualquer empresa concorrente. A mesma lei proíbe que uma empresa envase um botijão pertencente a outra.

Surgiu então a necessidade de realizar a troca de botijões vazios e, para isso, foi formado um consórcio com representantes de todas as empresas que estudou o assunto e desenvolveu uma rotina diária.

No final do dia cada empresa fará a contagem dos botijões recebidos separando-os por proprietário e lançando em um sistema ao qual todas terão acesso. O consórcio contratou uma transportadora que ficará responsável pelo leva e traz de botijões vazios.

Para racionalizar os custos dessa operação, ficou estabelecido que haverá uma quantidade padrão a ser transportada, tanto na ida como na volta. Por exemplo, suponha que a quantidade padrão seja de 50 botijões, considere os seguintes casos:

1. Empresa A tem 30 botijões da empresa B e a empresa B tem 21 botijões da empresa A – neste caso não haverá transporte;

2. Empresa A tem 53 botijões da empresa B e a empresa B tem 21 botijões da empresa A – neste caso não haverá transporte;

3. Empresa A tem 53 botijões da empresa B e a empresa B tem 61 botijões da empresa A – neste caso haverá o transporte que se dará da seguinte forma: Na ida 50 botijões são enviados de A para B e na volta 50 botijões retornam de B para A. O saldo ficará na empresa de origem (3 botijões para A e 11 botijões para B) aguardando uma nova oportunidade;

4. Empresa A tem 110 botijões da empresa B e a empresa B tem 61 botijões da empresa A – neste caso haverá o transporte que se dará da seguinte forma: Na ida A para B serão feitas duas viagens de 50 botijões cada uma; e no retorno de B para A será feita uma viagem de 50 botijões;

5. Como regra geral, foi estabelecido pelo consórcio que havendo botijões para a ida e também para a volta o transporte será feito. Se um sentido do transporte exigir múltiplas viagens, todas serão feitas, mesmo que no retorno haja apenas uma viagem.

Agora começa o seu trabalho. O Consórcio selecionou você para desenvolver o software responsável pela apuração dos totais diários a serem transportados, então mãos à obra.

Entrada

Cada entrada tem um caso de teste. Na primeira linha serão lidos dois números inteiros $NEmpr (3 \leq NEmpr \leq 10)$ e $QPadrao (QPadrao \geq 10)$, respectivamente o número de empresas envolvidas e a quan-

tidade padrão usada no transporte. Na segunda linha há um número inteiro contendo $NDias(NDias > 0)$, representando a quantidade de dias que há no período de apuração contido no caso de teste. Em seguida, haverá $NDias$ blocos de linhas contendo uma linha de cabeçalho, que deve ser descartada, mais $NRegs(NRegs = NEmpr^2 - NEmpr)$ linhas contendo três inteiros cada. Os dois primeiros são a empresa de origem $Empr1$, a empresa de destino $Empr2(1 \leq Empr1, Empr2 \leq NEmpr)$ e o terceiro é a quantidade de botijões em posse da empresa de origem e que pertence à empresa de destino.

Saída

A saída deve conter um bloco de linhas para cada dia do período contido na entrada. O bloco deve ser iniciado com um cabeçalho contendo o texto "Final dia #", sem aspas, onde o caractere # deve ser substituído pelo número do dia.

Nas linhas subsequentes deve constar o texto " Sem Trocas", sem aspas e com dois espaços em branco à esquerda, caso não haja trocas.

Se houver trocas, o texto deve ser (repare nos dois espaços em branco à esquerda) " Trocas entre #1(#2v) e #3(#4v)" onde #1 e #3 são os números das empresas envolvidas, sendo que o primeiro número (#1) deve ser menor que o segundo número (#3). Já os valores #2 e #4 são os números de viagens necessárias, sendo:

• #2 é o número de viagens de #1 para #3 e

• #4 é a quantidade de viagens de #3 para #1

Assim, a título de exemplo, o texto " Trocas entre 3(1v) e 4(2v)" significa que haverá trocas entre as empresas 3 e 4, sendo que haverá uma viagem com origem na empresa 3 e destino na 4, e haverá duas viagens com origem na empresa 4 e destino na 3.

Como pode haver mais de uma linha desse tipo, elas devem estar ordenadas pelo número #1 e, em caso de repetição deste, deve-se ordenar pelo número #3.

Exemplo de Entrada 1	Exemplo de Saída 1

3 20
4
Dia 1
1 2 15
1 3 8
2 1 6
2 3 11
3 1 3
3 2 9
Dia 2
1 2 4
1 3 7
2 1 8
2 3 12
3 1 8
3 2 5
Dia 3
1 2 9
1 3 0
2 1 8
2 3 11
3 1 2
3 2 17
Dia 4
1 2 8
1 3 7
2 1 2
2 3 7
3 1 5
3 2 11
	Final dia 1
	  Sem Trocas
	Final dia 2
	  Sem Trocas
	Final dia 3
	  Trocas entre 1(1v) e 2(1v)
	  Trocas entre 2(1v) e 3(1v)
	Final dia 4
	  Trocas entre 2(1v) e 3(1v)

Exemplo de Entrada 2

```
3 20
5
Dia 1
1 2 30
1 3 30
2 1 15
2 3 18
3 1 15
3 2 15
Dia 2
1 2 30
1 3 30
2 1 15
2 3 18
3 1 15
3 2 15
Dia 3
1 2 30
1 3 30
2 1 15
2 3 18
3 1 15
3 2 15
Dia 4
1 2 30
1 3 30
2 1 15
2 3 18
3 1 15
3 2 15
Dia 5
1 2 30
1 3 30
2 1 15
2 3 18
3 1 15
3 2 15
```

Exemplo de Saída 2

```
Final dia 1
  Sem Trocas
Final dia 2
  Trocas entre 1(3v) e 2(1v)
  Trocas entre 1(3v) e 3(1v)
  Trocas entre 2(1v) e 3(1v)
Final dia 3
  Trocas entre 1(1v) e 2(1v)
  Trocas entre 1(1v) e 3(1v)
  Trocas entre 2(1v) e 3(1v)
Final dia 4
  Trocas entre 1(2v) e 2(1v)
  Trocas entre 1(2v) e 3(1v)
  Trocas entre 2(1v) e 3(1v)
Final dia 5
  Sem Trocas
```

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema E

Crausio

Arquivo fonte: crausio.{ c | cpp | java | py }
Autor: Prof. Dr. Leandro Luque (Fatec Mogi das Cruzes)

Nos últimos anos, o uso de robôs limpadores tem aumentado nas residências brasileiras. Munarinho, um estudante de engenharia, aderiu recentemente a essa onda, mas, para economizar, decidiu criar o seu próprio robô: o Cráusio.

Como a grana para o projeto estava curta, ele criou um robô para o qual o usuário tem que configurar manualmente a rotina de movimentos. Para essa configuração, o local de limpeza é representado por uma matriz de $L \times C$ células. Cada célula representa um lugar onde Cráusio pode estar. A configuração inicial do Cráusio inclui a célula de partida (X, Y) e uma sequência de comandos: cima (C), baixo (B), esquerda (E) e direita (D).

No entanto, logo na primeira limpeza, Munarinho percebeu que subdimensionou a bateria de Cráusio e ele não consegue seguir muitos dos comandos especificados na sua rotina. Se Cráusio tem B de bateria, consegue executar B comandos. Outra coisa que Munarinho percebeu é que Cráusio não é muito resistente e, quando ele dá comandos errados e o Cráusio fica batendo nas paredes da casa, sua pintura se desgasta facilmente.

Ajude Munarinho a determinar qual é a última posição que Cráusio consegue alcançar em um dia, dada sua rotina, posição e bateria inicial. Ainda, determine quantas vezes Cráusio bateu nas paredes da casa, representadas pelos limites da matriz.

Entrada

A primeira linha contém três inteiros L, C e B ($1 \leq L, C \leq 100$; $0 \leq B \leq 10000$), representando o número de linhas e colunas da casa de Munarinho e a carga inicial de bateria do Cráusio, respectivamente.

A segunda linha contém dois inteiros X e Y ($1 \leq X \leq L$; $1 \leq Y \leq C$), representando a posição inicial do Cráusio na casa, sendo $(1, 1)$ o canto superior esquerdo.

A terceira linha contém uma string com R caracteres I ($1 \leq R \leq 10000$; $I \in \{C, B, E, D\}$), representando a rotina diária de Cráusio.

Saída

Imprima três inteiros, representando a última posição que o Cráusio conseguirá alcançar (sendo 1,1 o canto superior esquerdo) e a quantidade de vezes que ele bateu nas paredes da casa.

Exemplo de Entrada 1	Exemplo de Saída 1
5 5 10 3 3 CDCBEDBECB	3 3 0

Exemplo de Entrada 2	Exemplo de Saída 2
5 5 10 3 3 EEEEEEDDDDDD	3 5 4

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema F

Experimento com Spins

Arquivo fonte: spins.{ c | cpp | java | py }

Autor: Prof. Dr. Lucas Baggio Figueira (Fatec Ribeirão Preto)

Um cientista chamado Doc. Hugo Strange fez diversos experimentos com *spins*, elementos capazes de trocar o status de *quantum gates* quando passam por eles. Os *quantum gates* podem assumir os estados *OPEN* e/ou *CLOSED*. Nos experimentos, Doc. Hugo Strange percebeu que toda a vez que um *spin* é criado um *quantum gate* também é criado, e tal spin é responsável por alterar o seu estado. Entretanto, Doc. Hugo está com dificuldade em determinar quais *quantum gates* estarão abertos após a criação de N *quantum gates*, uma vez que ele percebeu que os *spins* só trocam o estado de *quantum gates* múltiplos da ordem em que foram criados. Por exemplo, o 3o. spin criado alterará o estado dos *quantum gates* 3, 6, 9, e assim por diante.

Considerando que, a cada experimento, N *spins* são criados e, consequentemente, N *quantum gates* serão criados com estado inicial *CLOSED*, Doc. Hugo Strange deseja saber quais ficarão abertos após a liberação dos *spins*, sendo que não é possível saber a ordem de liberação dos mesmos.

Entrada

A entrada contém vários casos de teste, onde cada caso tem um número inteiro N $(0 < N < 45 \times 10^6)$ indicando a quantidade de *spins* e *quantum gates*. O fim da entrada é dada por $N = 0$.

Saída

Para cada caso de teste, deve-se produzir uma linha de saída indicando quais *quantum gates*, em ordem crescente, estarão abertos após a liberação dos *spins*. Você deve separar a sequência de *quantum gates* por um espaço em branco.

Exemplo de Entrada 1	Exemplo de Saída 1
2 3 4 0	1 1 1 4

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema G

A Chave de Silas

Arquivo fonte: silas.{ c | cpp | java | py }

Autor: Prof. Me. Erico de Souza Veriscimo (Instituto Federal de São Miguel Paulista)

Vinicius e Alex desenvolveram um jogo cujo cenário pode ser representado por um vetor bidimensional. O protagonista chamado é o Silas, um jovem que está em uma aventura em busca da chave do tesouro. O cenário é composto não apenas por Silas e a chave, mas também por monstros de diversos níveis de força e bloqueios, além de obstáculos intransponíveis para Silas, como paredes, por exemplo.

Ao começar o jogo, é fornecida a força de Silas e sua posição inicial, assim como a localização da chave, dos monstros e dos bloqueios. O desafio consiste em determinar se é possível Silas alcançar a chave e, se for, qual o número mínimo de passos necessários para fazê-lo.

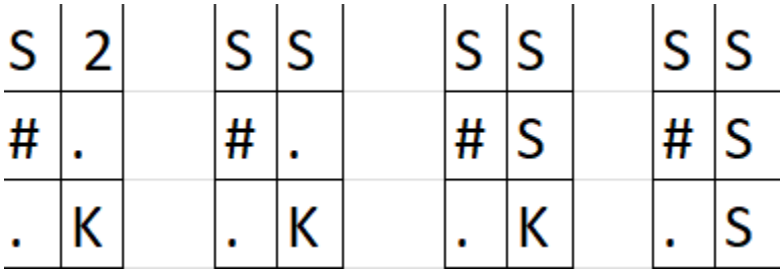
Silas tem a habilidade de derrotar monstros que possuem um nível de força igual ou inferior ao dele. Além disso, Silas pode se mover somente na horizontal e na vertical.

Exemplo:

Silas (S) inicia em (1,1) e possui força 10.

Existe um monstro com força 2 em (1,2).

Silas é capaz de derrotar o monstro e ocupar sua posição. Em seguida, pode avançar para a próxima posição vazia (representada por um ponto "." em (2,2)) e, finalmente, mover-se para a posição da chave, representada pela letra 'K' em (3,2). Nesse caso, foram necessários 3 movimentos. O símbolo # em (2,1) representa um bloqueio por onde Silas não pode passar.



Entrada

A primeira linha contém um valor inteiro $P(1 \leq P \leq 10000)$, que representa a força de Silas.

A segunda linha contém dois valores inteiros X e $Y(1 \leq X, Y \leq 80)$, correspondendo às dimensões do cenário (vetor bidimensional).

Finalmente, seguem-se X linhas, cada uma com Y valores separados por espaço, representando um elemento no cenário. Os possíveis elementos são: "S" (Silas), "K" (chave), "." (espaço livre), "$\#$" (bloqueio) e um valor inteiro que representa o nível de força do monstro presente naquela posição.

Saída

A saída consiste em uma única linha, contendo a letra N se Silas não conseguir alcançar a chave, ou um número inteiro indicando o número mínimo de passos entre Silas e a chave.

Exemplo de Entrada 1

```
10
4 7
S . . 2 . . .
. . 3 . # . .
. . . # . . .
. . 1 . 1 . K
```

Exemplo de Saída 1

```
9
```

Exemplo de Entrada 2

```
1
4 4
S . . 2
. . 3 .
. . . #
. . 2 K
```

Exemplo de Saída 2

```
N
```

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema H

Radar

Arquivo fonte: radar.{ c | cpp | java | py }
Autor: Prof. Antonio Cesar de Barros Munari (Fatec Sorocaba)

Uma das realidades da vida dos motoristas é o radar de velocidade. Tão onipresente quanto o custo do combustível, os pedágios, os buracos nas vias de trânsito, os radares semafóricos, os amarelinhos (ou marronzinhos e outros tonalidades), as mudanças nas regras de trânsito, os habilitados que não respeitam as mais elementares regras de trânsito e o famigerado IPVA, os dispositivos de controle de velocidade são onipresentes no dia a dia do condutor nas ruas desta Terra Brasilis. E essa situação desperta alguns instintos não muito nobres do ser humano, especificamente do ser humano que possui uma CNH. Zequinha é um motorista de aplicativo, seu amigo, que vive todos esses perrengues diariamente. Ele está absolutamente revoltado porque recebeu algumas multas por excesso de velocidade neste último mês.

Zequinha não é muito bom com números. Também não é particularmente imune a *fake news*. Ele acreditava que todo e qualquer radar de velocidade tinha uma tolerância de 10% (para mais, óbvio) em relação à velocidade efetivamente apurada pelo dispositivo. Pediu para você, seu amigo de infância inteligente, que "até faz Fatec", no curso mais disputado das Fatecs (em "Computação"), para calcular qual a velocidade que deveria ser considerada em cada uma das 95 multas que ele recebeu. Claro que você, inteligente como é, digno de um aluno de Fatec, foi consultar a informação exata sobre os critérios de margem de erro dos dispositivos de controle de velocidade, e descobriu que a realidade é um pouco diferente: se a velocidade medida for de até 107 km/h, será considerada uma margem de erro de 7 km/h; em caso contrário, a margem de erro será de 7% da velocidade medida. Isso quer dizer que para um dispositivo posicionado em um local cujo limite é 40 km/h, será multado apenas quem superar a velocidade de 47 km/h; em um radar configurado para 110 km/h, será autuado apenas quem superar 118 km/h. Obviamente, Zequinha está com uma ideia errada da situação. Seu trabalho é fazer um programa que, para uma velocidade limite informada para um determinado trecho, determina qual a maior velocidade que será efetivamente considerada para que a autuação não ocorra.

Entrada

A entrada possui apenas um caso de teste, consistindo de um inteiro V $(0 \leq \mathrm{V} \leq 300)$ que expressa a velocidade nominal para um radar.

Saída

Imprima a velocidade máxima, arredondada para um valor inteiro, que será permitida para que um veículo não seja autuado por excesso de velocidade.

Exemplo de Entrada 1	Exemplo de Saída 1
40	47

Exemplo de Entrada 2	Exemplo de Saída 2
110	118

Esta página foi propositadamente deixada em branco.

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema I

foraminis punchers

Arquivo fonte: foraminis.{ c | cpp | java | py }
Autor: Prof. Dr. Alex Marino (Fatec Ourinhos)

A guerra entre Sneakyland e Esbórnia persiste há muito tempo. As hostilidades entre os países escalaram a ponto de tornarem-se brutais. O exército de Sneakyland desenvolveu um sistema de mísseis teleguiados que têm causado destruição em alvos estratégicos esbornianos. O sistema, chamado ***Foraminis Punchers***, usa o sistema ***Phanthon Signum Scrambler*** para manter suas trajetórias invisíveis aos radares esbornianos.

A BOSSAD (serviço secreto esborniano), desesperada com a quase certa possibilidade de rendição, deslocou nosso amigo Joãozinho de sua ocupação no desenvolvimento do canhão de laser teletransportador (graças a ele, muitas vidas esbornianas foram salvas) e o incumbiu da missão de criar um sistema anti ***Foraminis Punchers***. Como bem conhecemos nosso amigo, missão dada é missão cumprida.

Nosso intrépido amigo descobriu, de maneira desconhecida (Joãozinho tornou-se um excelente espião), que a sequência de sinais digitais transmitidos pelo sistema ***Foraminis Punchers*** possui uma característica que possibilita identificar sua posição no espaço aéreo esborniano. Essa característica reside na faixa de frequência de ondas curtas, e é possível identificá-la se a energia na faixa baixa de frequência superar 50% da energia do sinal original.

Joãozinho e sua equipe de físicos desenvolveram um dispositivo de filtro de sinal que discretiza o sinal original, com cada sinal de entrada possuindo comprimento/suporte igual a oito, ou seja, suporte $M = 8$. Para identificar o míssil, ainda é preciso criar um filtro passa alta que segregue os componentes de baixa frequência e, por fim, medir sua energia. Para elaborar o filtro passa alta, Joãozinho optou por utilizar o algoritmo de *Mallat* e usará como convolucionador o par de filtros de *Haar*.

A explicação de Joãozinho para a solução do problema segue-se:
Dado um sinal S[.] = { $S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7$} e o par de filtros Haar, nos quais H={h_0, h_1} e G={g_0, g_1}, tem-se :

$$\underbrace{\begin{pmatrix} h_0 & h_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ g_0 & g_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_0 & g_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_0 & h_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_0 & g_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_0 & h_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & g_0 & g_1 \end{pmatrix}}_{\text{8x8 matrix } A[\cdot][\cdot]} \cdot \underbrace{\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{pmatrix}}_{\text{input } S[\cdot]} = \underbrace{\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7 \end{pmatrix}}_{\text{output } R[\cdot]}$$

Figura I.1: M[.][.]×S[.]=R[.]

O par de filtros de Haar é composto por: H={ $\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}$} e G={ $\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}$}.

De tal modo que o sinal resultante R[.]={ $R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7$} contém os componentes de alta

(HF) e baixa (LF) frequência conforme explícitos na equação abaixo:

$$\begin{array}{ll} i \in HF & \forall i\%2 == 1 \\ i \in LF & \forall i\%2 == 0 \end{array} \tag{1}$$

Um vez determinados os sinais HF e LF, podemos facilmente identificar a fragilidade do sistema ***foraminis punchers*** pela equação abaixo:

$$\begin{cases} \text{identificado}, & \text{se } \frac{E(LF[.])}{E(S[.])} > 0.5, \\ \text{não identificado}, & \text{caso contrário.} \end{cases} \tag{2}$$

Por fim, o cálculo da energia de um sinal digital dá-se por:

$$E(S[.]) = \sum_{i=0}^{M-1} S_i^2 \tag{3}$$

Em que M é o número de componentes do sinal de interesse ou também chamado de suporte do sinal.

Sabemos que Joãozinho não é bom programador e por isso ele te convocou para salvar a população esborniana dos ataques perversos dos endiabrados sneakys. Joãozinho e o povo esborniano contam com sua preciosa ajuda. Assim seu trabalho é implementar o algoritmo explicado.

Entrada

A primeira linha contém o número N de sinais coletados, tal que $N <= 500$. Seguida pelas N linhas subsequentes, cujos conteúdos representam os 8 componentes de cada sinal de interesse a ser avaliado.

Saída

Para cada sinal de interesse avaliado, seu programa deve responder INIMIGO caso identifique a fragilidade do sinal ou hífen "-" caso contrário.

Exemplo de Entrada 1	**Exemplo de Saída 1**
4 7 -5 4 -3 7 -1 -4 -10 -6 8 -10 -8 -7 10 -7 -7 8 -4 1 -10 6 0 2 -4 1 -8 -5 5 -3 5 6 -7	- INIMIGO - -

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema J

A Cifra do Palíndromo Perdido

Arquivo fonte: palindromo.{ c | cpp | java | py }
Autor: Rafael Pacheco (Fatec Sorocaba)

Era uma vez, em um reino distante, uma misteriosa e antiga profecia que falava sobre a existência de um Palíndromo Perdido, uma sequência de letras mágicas que detinha um poder extraordinário. Um palíndromo é uma palavra ou frase que permanece igual quando lida de trás para frente. Diziam que quem decifrasse o Palíndromo Perdido seria agraciado com uma imensa sabedoria e poderes inimagináveis. Nesse reino, vivia um jovem aprendiz de feiticeiro chamado Iranum. Obcecado pela magia das palavras, ele embarcou em uma jornada épica para desvendar a Cifra do Palíndromo Perdido. A lenda dizia que, ao encontrar a sequência de letras correta e pronunciar seu encantamento especial, Iranum teria acesso aos segredos mais profundos do universo. Contudo, a tarefa não seria fácil. A Cifra do Palíndromo Perdido era uma sequência de caracteres que poderia conter letras repetidas e até mesmo números. Iranum precisa criar um programa capaz de identificar se uma sequência de caracteres era, de fato, o tão procurado Palíndromo Perdido, porém, Iranum não entende muito sobre programação e pediu a sua ajuda nessa busca por conhecimento e poderes extraordinários! Sua missão é desenvolver um programa que, ao receber uma sequência de caracteres, seja capaz de decifrar se ela é o Palíndromo Perdido ou não. Observe nos exemplos de entrada as regras de um Palíndromo Perdido.

Entrada

A entrada contém vários casos de teste e termina com EOF. Cada caso de teste consiste em várias sequências de caracteres alfanuméricos, composta por letras minúsculas, maiúsculas, números e até mesmo símbolos. Cada sequência de caracteres pode ter no máximo 10.000 caracteres.

Saída

A saída consiste em uma resposta mágica que revelará se a sequência de caracteres é o Palíndromo Perdido ou não. Se a sequência for de fato um palíndromo, sua resposta deve ser "Parabéns, você encontrou o Palíndromo Perdido!". Caso contrário, seu programa deve responder com a mensagem "A busca continua, o Palíndromo Perdido ainda não foi encontrado.

Exemplo de Entrada 1

```
Subi no ônibus.
Anotaram a data da Maratona
A rã ama arara. A arara ama a rã
Queremos paçoca
```

Exemplo de Saída 1

```
Parabens, voce encontrou o Palindromo Perdido!
Parabens, voce encontrou o Palindromo Perdido!
A busca continua, o Palindromo Perdido ainda nao foi encontrado.
A busca continua, o Palindromo Perdido ainda nao foi encontrado.
```

Exemplo de Entrada 2	Exemplo de Saída 2
1----------1 1,0,1,0,1,0,1,0,1 4r4R4	Parabens, voce encontrou o Palindromo Perdido! Parabens, voce encontrou o Palindromo Perdido! Parabens, voce encontrou o Palindromo Perdido!

MARATONA DE PROGRAMAÇÃO
InterFatecs

Problema K

Grande Evento

Arquivo fonte: evento.{ c | cpp | java | py }
Autor: Guilherme Sernajoto (Fatec Sorocaba)

O prefeito da cidade de Praia Grande está planejando implementar melhorias na mobilidade do município devido à expectativa de receber muitos competidores no grande evento da fase final da Maratona Interfatecs. Para isso, ele solicitou ao Centro de Tráfego Avançado Futurista (CETAF) uma análise da situação atual e foi constatado que existem lugares na cidade onde não é possível viajar de um ponto V para um ponto W. Diante dessa constatação, o prefeito decretou que em todos os pontos da cidade deve ser possível viajar entre quaisquer dois pontos, ou seja, dado um par de pontos V e W, deve ser possível viajar de V para W e de W para V. Como estagiário do CETAF, foi designada a você a tarefa de desenvolver um programa capaz de determinar, com base no sistema de tráfego da cidade, se é possível viajar entre quaisquer dois pontos ou não.

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso de teste contém dois inteiros, N ($2 \leq N \leq 2000$), indicando a quantidade de pontos em uma cidade e M ($2 \leq M \leq N(N-1)/2$), representando o número de ruas da cidade. Segue-se então M linhas, onde cada linha possui três inteiros, V, W e D, onde V e W ($1 \leq V,W \leq N, V \neq W$) representam pontos distintos e D pode ser 1 ou 2, representando respectivamente rua de mão única entre V e W, ou então rua de mão dupla que liga V e W. A entrada se encerra quando N e M forem iguais a zero.

Saída

Para cada caso de teste seu programa deve imprimir "S" caso seja possível viajar entre quaisquer dois pontos da cidade, ou "N", caso contrário.

Exemplo de Entrada 1	Exemplo de Saída 1
4 2 1 2 2 3 4 2 4 5 1 2 1 1 3 2 2 4 1 3 4 1 4 1 2 0 0	N S