

MARATONA DE PROGRAMAÇÃO

# InterFatecs

XI Maratona InterFatecs  
Santana de Parnaíba 2022

Fase 2 - 24/09/2022

## Caderno de Problemas


Organização e Patrocínio

**Fatec**  
Santana de  
Parnaíba

**CPS**  
Centro  
Paula Souza

**SÃO PAULO**  
GOVERNO DO ESTADO  
| Secretaria de Desenvolvimento Econômico

  
**Go Ahead**


 **baymetrics**

**RCV**  
Rádio Clube de  
Votorantim


**DELFI**

 **PREFEITURA DE  
SANTANA DE PARNAÍBA**

**Natrielli**  
PRODUTOS DE CONFIANÇA

 **cordier**

 **devprime**  
UNILABOR. SOFTWARE. INOVAÇÃO.

 **SEMEDS**  
Secretaria de Inovação,  
Desenvolvimento Econômico,  
Ciência, Tecnologia, Inovação.

 **btgpactual**  
Banco BTG Pactual

 **SICOOB**  
Meridional

[interfatecs.com.br](http://interfatecs.com.br)

## 1 Instruções

Este caderno contém 11 problemas – identificados por letras de A até K, com páginas numeradas de 3 até 26. 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) É permitido somente acesso a conteúdo impresso em papel (cadernos, apostilas, livros);
- (d) Não é permitida a comunicação com o técnico ou qualquer outra pessoa que não seja a equipe para tirar dúvidas sobre a maratona
- (e) 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;
- (f) NÃO é permitido o uso de notebooks ou outro tipo de computador ou assistente pessoal;
- (g) 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 5.4.0 (lembre-se que não existem bibliotecas do Windows)
- (b) g++ version 5.4.0 (lembre-se que não existem bibliotecas do Windows)
- (c) javac 1.8.0-272 (lembre-se que as classes devem estar fora de pacotes)
- (d) Python 2.7.12 e 3.9.1

## Problema A

### Movies or Series

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

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

Hellen, 26, and Ian, 29, are a couple who love streaming services and keep a list of titles they want to watch. This list mixes movies and series.

When they have the opportunity to watch something, it is common for them to spend a lot of time deciding whether to watch a movie or a series. To not waste all this time, they created an objective criterion to help them, adding the age of each one with the day and month number to get a result. If the result is odd, they watch a movie, and if it is even, they watch a series.

Now, programmer, they need your help!

They gave you a file containing many days they probably will have spare time and asked you to write a software to tell them what to watch each day! The list of days contains the day and month numbers in consecutive lines and ascending order. The task is to read two consecutive lines, getting the day and month numbers, compute its sum together with Hellen's and Ian's age and verify its parity to decide about a Movie or a Series.

Be careful with one detail: as the criterion involves the age of the two, they were careful to record their birthdays there! When date of Helen's birthday is reached they recorded a 99 into the file and 98 for Ian's. Consider this in your software.

#### Entrada

The input is composed of many lines with one integer each and ending with 0 (zero). You must consider them in pairs representing a day/month, except when you find 98 (Ian's birthday) or 99 (Hellen's birthday).

#### Saída

Print **WATCH MOVIE** if the sum (day + month + Ian's age + Hellen's age) is odd, **WATCH SERIES** if it is even. When you reach Hellen's birthday print **HELLEN BIRTHDAY, GO OUT FOR DANCING** and in case of Ian's print **IAN BIRTHDAY, GO OUT FOR DINNER**

Don't forget the end of line in your output.



9/2 → Ian  
3/10 → Hellen

Example of Input 1

-7 2  
 2  
 -7 5  
 2  
 -7 9  
 2  
 -7 16  
 2  
 -7 25  
 2  
 -7 98  
 1  
 3  
 8  
 3  
 10  
 3  
 19  
 3  
 24  
 3  
 26  
 3  
 2  
 4  
 10  
 4  
 15  
 4  
 -7 99  
 18  
 4  
 23  
 4  
 0

Example of Output 1

WATCH MOVIE  
 WATCH SERIES  
 WATCH SERIES  
 WATCH MOVIE  
 WATCH SERIES  
 IAN BIRTHDAY, GO OUT FOR DINNER  
 WATCH SERIES  
 WATCH MOVIE  
 WATCH MOVIE  
 WATCH SERIES  
 WATCH MOVIE  
 WATCH MOVIE  
 WATCH SERIES  
 WATCH SERIES  
 WATCH MOVIE  
 HELLEN BIRTHDAY, GO OUT FOR DANCING  
 WATCH MOVIE  
 WATCH SERIES

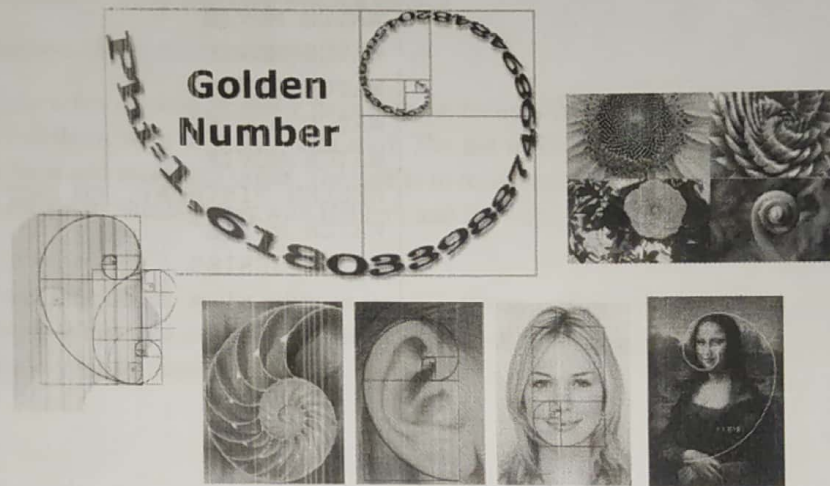
## Problema B

# Golden Number

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

Autor: Prof. Dr. Alex Marino (Fatec Ourinhos)

The golden number is the mathematical representative of perfection in nature. It has been studied since antiquity, and many Greek buildings and artistic works have this number as a base. The golden number is represented by the Greek letter phi and is obtained by the proportion  $\phi = \frac{1+\sqrt{5}}{2} = 1.61803399....$  But why is this number so significant? Why does he represent perfection, the beauty of nature? The answer is simple: it appears almost everywhere in nature and in the things we consider most beautiful.



## Input

Each test case entry consists of an integer  $N \leq 100$ .

## Output

For each test case your program must output a real number accurate to within 15 places, followed by a line break.

### Example of Input 1

1
---

### Example of Output 1

2.000000000000000

### Example of Input 2

2
---

### Example of Output 2

1.500000000000000

## Problema C

# Pera Ai... Tá Lidando Errado!

Arquivo fonte: `peraitalidandoerrado.{ c | cpp | java | py }`

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

Tibúrcio Lopes Espinosa é um grande amigo seu que está precisando de ajuda. Sabendo que você é bom com computadores ele pede que você escreva um programa para calcular alguns valores que está precisando com urgência. O caso dele é sério e o tempo é vital - ele precisa de tudo muito rápido...

O problema ele te explicou direitinho informando que se  $N$  é um número natural, é sabido que, caso  $N$  tenha raiz quadrada exata (também número natural), então ele pode ser escrito como a somatória dos termos de uma Progressão Aritmética iniciando em 1, com razão 2.

O Tibúrcio te passou um exemplo:

sendo  $N = 49$ , sua raiz quadrada é exata:  $\sqrt{49} = 7$

A Progressão Aritmética correspondente contém os termos:

1	3	5	7	9	11	13
---	---	---	---	---	----	----

...cuja Somatória produz o valor de  $N$ , ou seja:  $1 + 3 + 5 + 7 + 9 + 11 + 13 = 49$

Você deve escrever um programa que leia um número inteiro que será o valor de  $N$  e **imprima o maior elemento** da Progressão Aritmética que produz esse  $N$  de entrada.

Mas cuidado! Isso só dá certo quando a raiz quadrada de  $N$  é um Número Natural, Ok?

E seja ligeiro, o Tibúrcio tem pressa... tic, tac, tic, tac...

## Entrada

A entrada consiste de várias linhas, cada uma contendo um único número inteiro  $N$  e finalizando com zero. Os limites de  $N$  são:  $1 \leq N \leq 2^{64} - 1$

## Saída

A saída consiste em imprimir o maior elemento da Progressão Aritmética capaz de gerar  $N$ , conforme descrito acima. E saiba que erros podem acontecer: se a raiz quadrada do  $N$  de entrada não for um Número Natural o programa deverá identificar tal situação e exibir a mensagem:

**PERA AI... TA LIDANDO ERRADO!**

em letras maiúsculas sem acentos. Não esqueça de imprimir o salto de linha no final.



**Exemplo de Entrada 1**

```
49
169
256
230
25
440
14364100
16
4611686018427387904
214130
0
```

**Exemplo de Saída 1**

```
13
25
31
PERA AI... TA LIDANDO ERRADO!
9
PERA AI... TA LIDANDO ERRADO!
7579
7
4294967295
PERA AI... TA LIDANDO ERRADO!
```



## Problema D

# Radioatividade

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

Autor: Prof. Dr. Alex Marino (Fatec Ourinhos)

Nosso amigo Joãozinho tornou-se um grande filantropo e cidadão respeitado na Esbórnica. Toda comunidade esborniana admira sua história de vida. Joãozinho enternecido com o apreço que lhe dedicam, não se esvai de nenhuma demanda que lhe submetem. Desta vez o Museu Nacional de Mercuria (capital esborniana) e pesquisadores da Faculdade Nacional de Arqueologia desafiaram Joãozinho. O desafio consiste em determinar a idade de artefatos arqueológicos encontrados pelos pesquisadores. Para tal, nosso intrépido amigo sabe que não se trata de uma tarefa difícil para quem domina a "**Lei fundamental da desintegração radioativa**", e Joãozinho é um destes caras :-P.

A atividade de uma fonte radioativa representa a sua taxa de desintegração, obtida do decaimento radioativo, para um radionuclídeo isolado, em que o decréscimo no número de átomos do elemento radioativo por unidade de tempo  $N'(t)$  é proporcional ao número de átomos ainda não desintegrado  $N(t)$ , ou seja:

$$N = N_0 e^{-\lambda t}$$

onde temos que:

$N$  - número de núcleos radioativos remanescentes após um tempo  $t$

$N_0$  - número de núcleos radioativos na amostra num tempo  $t = 0$

$e$  - número de neper

$\lambda$  - Constante de desintegração

Para determinação da constante de desintegração ( $\lambda$ ) temos:

$$\lambda = \frac{\ln(2)}{t_{\frac{1}{2}}}$$

Joãozinho nos apresenta um exemplo prático onde a atividade de um certo fóssil diminui de 1530 desintegrações por segundo para 190 desintegrações por segundo, já com a correção da radiação de fundo, durante o processo de fossilização. Sendo a meia-vida do isótopo radioativo do  $C_{14}$  (carbono 14) de 5730 anos. Para determinar a idade do fóssil temos:

1. Determinar  $\lambda$ :

$$\lambda = \frac{\ln(2)}{t_{\frac{1}{2}}} \implies \frac{\ln(2)}{5730} = 0.00012096809$$

2. Determinar idade do fóssil:

$$N = N_0 e^{-\lambda t} \implies 190 = 1530 e^{-\lambda t} \implies t = 17244.20769 \cong 17244 \text{ anos}$$

Como já sabemos que Joãozinho não é bom programador, ele pede novamente sua ajuda para elaborar um programa que, dados o tempo de meia vida, números de núcleos radioativos em  $t_0$  e após um tempo  $t$ , responda a idade do fóssil.

## Input

Cada caso de teste consiste de três números inteiros,  $0 < N_0 \leq 5 \times 10^4$ ,  $0 < N \leq 5 \times 10^4$  e  $0 < t_1 \leq 5 \times 10^4$ .

## Output

A sua resposta deve conter apenas a parte inteira do resultado obtido.

### Exemplo de Entrada 1

1530  
190  
5730

### Exemplo de Saída 1

17244

### Exemplo de Entrada 2

2000  
537  
23647

### Exemplo de Saída 2

44858

## Problema E

# Campeonato

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

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

Seu amigo Pereirinha mora em uma cidade pequena, é um ex-jogador de futebol profissional e educador físico. Entre as várias atividades que exerce está a organização de campeonatos esportivos de várias modalidades e formatos de competição.

O formato que ele mais gosta é o de pontos corridos semelhante ao campeonato brasileiro de futebol masculino – o Brasileirão. Nesse formato todos os times jogam entre si – todos contra todos em várias rodadas – e o campeão será aquele que obtiver os melhores resultados segundo os critérios pré-estabelecidos na competição.

No caso do Brasileirão há dois jogos entre cada time alternando o mando de campo. Porém, nas competições organizadas pelo Pereirinha não há jogos de ida e volta, ou seja, todos os times jogam contra todos os demais, mas apenas uma vez já que os jogos acontecem no ginásio municipal e não há mando de campo, embora haja torcidas bem animadas. O critério de classificação que o Pereirinha usa segue os seguintes itens, em ordem de importância:

1. Maior número de pontos
2. Maior número de vitórias
3. Maior saldo de gols
4. Maior número de gols a favor

Caso haja empate em Número de Pontos (item 1) passa-se para o Número de Vitórias (item 2), se houver empate nos dois primeiros, passa-se para o Saldo de Gols (item 3) e assim por diante. Veja o exemplo:

Exemplo com 4 times

1 APUCARANA  
2 PARANA  
3 SERRANO  
4 VILAREAL

Apuração

	Pt	V	E	D	GP	GC	SG
1 APUCARANA	0	0	0	3	3	10	-7
2 PARANA	4	1	1	1	6	8	-2
3 SERRANO	6	2	0	1	9	3	6
4 VILAREAL	7	2	1	0	5	2	3

São 6 jogos

(1) APUCARANA 0 x 2 (4) VILAREAL  
(2) PARANA 1 x 4 (3) SERRANO  
(1) APUCARANA 0 x 4 (3) SERRANO  
(4) VILAREAL 1 x 1 (2) PARANA  
(1) APUCARANA 3 x 4 (2) PARANA  
(3) SERRANO 1 x 2 (4) VILAREAL

Classificação Final

	Pt	V	E	D	GP	GC	SG
4 VILAREAL	7	2	1	0	5	2	3
3 SERRANO	6	2	0	1	9	3	6
2 PARANA	4	1	1	1	6	8	-2
1 APUCARANA	0	0	0	3	3	10	-7

Apurar isso manualmente dá um trabalho enorme e é muito sujeito a erros. Como os campeonatos dele fazem sucesso, ele é muito requisitado, mas não tem condições de atender os chamados por falta de uma ferramenta que facilite essa apuração.

À essa altura você já entendeu porque está aqui, né? O Pereirinha precisa de você que vai escrever um software no qual ele lance os resultados dos jogos e a classificação seja produzida de maneira rápida e correta.

## Entrada

A entrada consiste em um único caso de teste. Na primeira linha há um número inteiro par  $QtTimes$  ( $4 \leq QtTimes \leq 20$ ) que é a quantidade de times no campeonato e na sequência haverá  $QtTimes$  linhas contendo duas informações em cada uma: um inteiro que é o identificador do time e uma cadeia de caracteres que é o nome dele com um máximo de 20 caracteres e sem espaços em branco.

Após isso haverá uma linha com um número inteiro  $QtJogos$  ( $QtJogos = (QtTimes^2 - QtTimes)/2$ ) que é a quantidade de jogos do campeonato e na sequência haverá  $QtJogos$  com quatro inteiros cada uma, na seguinte ordem: número identificador do time 1, número identificador do time 2, quantidade de gols do time 1 e quantidade de gols do time 2.

## Saída

A partir dos resultados dos jogos você deve apurar a classificação do campeonato e imprimir na saída três linhas com: o nome do time vencedor seguido da palavra CAMPEAO em letras maiúsculas sem acento; o nome do segundo colocado seguido da expressão SEGUNDO LUGAR em letras maiúsculas; o nome do terceiro colocado seguido da expressão TERCEIRO LUGAR em letras maiúsculas

### Exemplo de Entrada 1

```

4
1 APUCARANA
2 PARANA
3 SERRANO
4 VILAREAL
6
1 4 0 2
2 3 1 4
1 3 0 4
4 2 1 1
1 2 3 4
3 4 1 2
  
```

### Exemplo de Saída 1

```

VILAREAL CAMPEAO
SERRANO SEGUNDO LUGAR
PARANA TERCEIRO LUGAR
  
```

$r = \text{input}$   
 $Jogos = (r^2 - r) / 2$



### Exemplo de Entrada 2

6  
1 ALVORADA  
2 BRAVOS  
3 CEUTA  
4 DRACENA  
5 ELDORADO  
6 FARRAPOS  
15

1 6 4 1  
2 5 1 1  
3 4 1 4  
1 5 1 1  
6 4 4 4  
2 3 2 1  
1 4 2 0  
5 3 3 3  
6 2 3 2  
1 3 4 2  
4 2 0 3  
5 6 3 2  
1 2 1 3  
3 6 2 1  
4 5 0 2

### Exemplo de Saída 2

ALVORADA CAMPEAO  
BRAVOS SEGUNDO LUGAR  
ELDORADO TERCEIRO LUGAR

## Problema F Interestelar

Arquivo fonte: `interestelar.{ c | cpp | java | py }`

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

O enredo é conhecido: não temos muito tempo nesse planeta e precisamos encontrar algum outro que seja semelhante a esse que possa ser colonizado por nós e pelos nossos descendentes. Sendo assim, na busca por meio do radio-telescópio HAL captamos dados fracos de um possível planeta, denominado de Eniootat fora do nosso sistema solar, mas ali na esquina da Via-Láctea, aproximadamente 332 anos-luz da Terra. O HAL tem captado dados do planeta, mas os dados estão chegando embaralhados, sendo assim, temos que analisar tais dados encontrando uma sequência limpa, ou seja, com a menor quantidade de dados inválidos.

### Entrada

A entrada se dá por meio de casos de teste contendo as seguintes informações, expressas como números inteiros:

- $N$  tamanho da sequência de dados ( $0 < N < 10^4$ )
- $L, U$  intervalo que determina quais dados são válidos  $[L, U]$
- Sequência de dados em si, dispostos em  $N$  linhas.
- Sequência de  $X, Y$  que determinam quais intervalos de dados devem ser avaliados.

Atenção o *EOF* determina o fim da entrada de dados.

### Saída

A saída é composta pelas quantidades de dados válidos em cada um dos intervalos.

Exemplo de Entrada 1

10  
 100 150  
 49  
 108  
 122  
 78  
 109  
 115  
 142  
 3  
 129  
 112  
 3 8  
 9 9  
 0 3

Exemplo de Saída 1

4  
 1  
 2

## Problema G

### Entrada para a Mega

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

Autor: Adaptado TimusOJ

Um semáforo na entrada para o Mega Shopping Center a partir da rodovia Novomoskovskiy funciona de forma que  $K$  veículos são capazes de cruzá-lo em um minuto. Nos finais de semana todos os habitantes da cidade dirigem-se até o shopping para fazer compras, o que resulta em um enorme congestionamento na entrada. A administração do shopping instalou câmeras na ponte próxima, o que possibilitou calcular a quantidade de veículos chegando à entrada do shopping vindos do centro. A observação desse fluxo de veículos iniciou-se  $N$  minutos atrás. Você deve usar os dados da câmera para determinar a quantidade de carros atualmente esperando no congestionamento.

#### Entrada

A entrada é composta por diversos casos de teste, cada um contendo duas linhas. A primeira linha contém inteiros  $K$  e  $N$  ( $1 \leq K, N \leq 100$ ) que indicam a quantidade de veículos que chegam à entrada da Mega em um minuto e a quantidade de minutos decorridos desde o início da observação. A segunda linha contém inteiros  $A$  separados por espaço em branco,  $A_1 \dots A_N$  ( $0 \leq A_i \leq 100$ ), onde  $A_i$  é a quantidade de veículos que chegam à entrada durante o  $i$ -ésimo minuto. A observação iniciou-se pela manhã, quando não havia veículo algum esperando na entrada. O final da entrada é determinado por uma linha contendo  $K = N = 0$ , que não deverá ser processada.

#### Saída

Para cada caso de teste imprima a quantidade de veículos atualmente esperando no congestionamento.

##### Exemplo de Entrada 1

5 3	0
6 7 2	5
5 3	
20 0 0	
0 0	

##### Exemplo de Saída 1



## Problema H

# Os cigarros de Pedro

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

Autor: Adaptado UVa OJ

Pedro tem  $N$  cigarros de palha. Ele os fuma um a um, guardando todas as bitucas em seu bolso. A partir de  $K > 1$  bitucas ele consegue produzir um novo cigarro de palha para ser fumado. Quantos cigarros Peter pode fumar?

### Entrada

A entrada é uma sequência de linhas. Cada linha contém dois números inteiros determinando os valores de  $N$  e de  $K$ . A entrada é terminada por uma linha contendo  $N = K = 0$ .

### Saída

Para cada linha da entrada, imprima um número inteiro em uma linha separada dando a quantidade máxima de cigarros de palha que Pedro pode fumar.

#### Exemplo de Entrada 1

4 3	5
10 3	14
100 5	124
0 0	

#### Exemplo de Saída 1

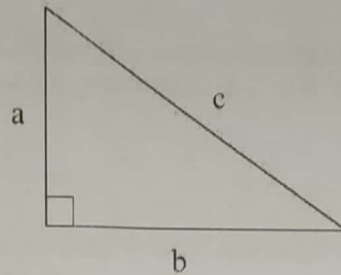
## Problema I

# Simplificando a geometria

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

Autor: Adaptado ZOJ

Matemática pode ser fácil quando você tem um computador. Considere o seguinte exemplo: você provavelmente sabe que em um triângulo corretamente angulado, o comprimento dos três lados  $a$ ,  $b$ ,  $c$  (onde  $c$  é o lado mais comprido, chamado de hipotenusa) satisfaz a relação  $a^2 + b^2 = c^2$ , chamada de Lei de Pitágoras. Aqui nós vamos tratar do problema de calcular o comprimento do terceiro lado, se os outros dois lados forem conhecidos.



### Entrada

A entrada contém a descrição de vários triângulos. Cada descrição consiste de uma linha contendo três inteiros  $a$ ,  $b$  e  $c$ , informando os comprimentos dos respectivos lados de um triângulo corretamente angulado. Exatamente um dos três números é igual a -1 (o lado 'desconhecido'), os outros são positivos (os lados 'conhecidos'). Uma descrição contendo  $a = b = c = 0$  encerra a entrada.

### Saída

Para cada descrição de triângulo na entrada, primeiro imprima o número do triângulo, como mostrado no exemplo de saída. Então imprima "Impossible." se não há um triângulo corretamente angulado que tenha os lados 'conhecidos'. Em caso contrário imprima o comprimento do lado 'desconhecido' no formato " $s = x$ ", onde  $s$  é o nome do lado desconhecido ( $a$ ,  $b$  ou  $c$ ), e  $x$  é o seu comprimento.  $x$  precisa ser impresso com exatamente três dígitos após o ponto decimal.

#### Exemplo de Entrada 1

```
3 4 -1
-1 2 7
5 -1 3
0 0 0
```

#### Exemplo de Saída 1

```
Triangle #1
c = 5.000
Triangle #2
a = 6.708
Triangle #3
Impossible.
```

## Problema J Chocolate

Arquivo fonte: chocolate.{ c | cpp | java | py }  
Autor: Adaptado TimusOJ

Maria está furiosa com João, que comeu todos os doces da casa e, não contente, atacou até os doces dos vizinhos. Agora os pais de Maria deixaram no armário da cozinha uma barra de chocolate e, com certeza, João vai querer comê-la também. Desta vez, entretanto, Maria está decidida que nenhum pedacinho daquele chocolate será comido pelo guloso. Ela pretende usar o vício de João por jogos de azar e vai sugerir a ele a seguinte disputa: uma barra de chocolate pode ser considerada como um retângulo de unidades “quadradas” dispostas em  $m$  linhas e  $n$  colunas, separadas por “linhas”. Dois jogadores jogam alternadamente. Em sua vez, um jogador precisa pegar um pedaço de chocolate e dividi-lo em dois em uma das “linhas” existentes. Se um jogador não puder fazer um movimento legal (o que acontecerá quando todas as peças de chocolate consistirem de apenas uma unidade “quadrada”), ele perde, e o ganhador fica com todo o chocolate. Mas João é bem inteligente e imediatamente percebeu quem deve fazer o primeiro movimento para que ele próprio ganhe, assumindo que os jogadores farão os melhores movimentos possíveis. Você saberia como?

### Entrada

A entrada é composta por vários casos de teste. Inicialmente é informado um inteiro  $C$ ,  $1 \leq C \leq 500$ , que indica a quantidade de casos a serem processados. Seguem-se  $C$  linhas compostas, cada uma, por dois inteiros  $m$  e  $n$ ,  $1 \leq m, n \leq 50$ , que representam respectivamente as quantidades de linhas e de colunas da barra de chocolate em disputa.

### Saída

Para cada caso de teste imprima uma linha contendo ‘[:=[primeiro]’ (sem as aspas) se João deve fazer o primeiro movimento para vencer e ‘[segundo]=:]’ (sem as aspas) em caso contrário.

#### Exemplo de Entrada 1

```
2
2 4
1 3
```

#### Exemplo de Saída 1

```
[:=[primeiro]
[segundo]=:]
```

## Problema K

### Faça o untwist

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

Autor: Adaptado UVA OJ

Criptografia trata de métodos de comunicação secreta que transformam uma mensagem (o texto base) em uma forma disfarçada (o texto cifrado), de maneira que ninguém que veja o texto cifrado será capaz de compreender o texto base, exceto o destinatário desejado. Transformar o texto base no texto cifrado é criptografar; transformar o texto cifrado no texto base é descriptografar. Twisting é um método simples de criptografia que requer que tanto o emissor como o destinatário concordem com uma chave secreta  $k$ , que é um número inteiro positivo.

O método usa quatro arrays: texto-base e texto-cifrado são arrays de caracteres e código-base e código-cifrado são arrays de inteiros. Todos os arrays têm  $n$  elementos, onde  $n$  é o comprimento da mensagem a ser criptografada. Arrays têm seus índices com base zero, então os elementos são numerados de 0 a  $n - 1$ . Para este problema todas as mensagens conterão apenas letras minúsculas, o ponto e o caracter hífen '-' (para representar o espaço em branco).

A mensagem a ser criptografada é armazenada no array texto-base. Dada uma chave  $k$ , o método de criptografia funciona como descrito a seguir. Primeiro converta as letras de texto-base para códigos inteiros no array código-base conforme a regra: '-' = 0, 'a' = 1, 'b' = 2, ... '.' = 27. Em seguida converta cada código de código-base aplicando a seguinte fórmula: para todo  $i$  de 0 até  $n - 1$ ,

$$\text{código-cifrado}[i] = (\text{código-base}[ki \bmod n] - i) \bmod 28$$

(Aqui  $x \bmod y$  é o resto positivo quando  $x$  é dividido por  $y$ . Por exemplo,  $3 \bmod 7 = 3$ ,  $22 \bmod 8 = 6$  e  $-1 \bmod 28 = 27$ ).

Finalmente, converta os códigos de código-cifrado para caracteres em texto-cifrado de acordo com a regra listada anteriormente. A mensagem final convertida (twisted) estará em texto-cifrado. Criptografando, por exemplo, a mensagem 'cat' por meio da chave 5 teremos o seguinte:

Array	0	1	2
texto-base	'c'	'a'	't'
código-base	3	1	20
código-cifrado	3	19	27
texto-cifrado	'c'	's'	'.'

$$27 = (5) \cdot 5 + 2$$

Sua tarefa será escrever um programa capaz de fazer o untwist da mensagem, ou seja, a sua descriptografia, convertendo o texto cifrado de volta para o texto base dada a chave  $k$ . Por exemplo, dada a chave 5 e o texto cifrado 'cs.' o seu programa precisará imprimir o texto base 'cat'.

### Entrada

O arquivo de entrada contém um ou mais casos de teste, seguidos por uma linha contendo o número zero, que sinaliza o final das entradas. Cada caso de teste está em uma linha própria e consiste de uma chave  $k$ , um espaço em branco e a mensagem cifrada que contém no mínimo 1 e no máximo 70 caracteres. A chave  $k$  será um inteiro positivo não maior que 300.



## Saída

Para cada caso de teste imprima uma linha contendo a mensagem descriptografada.

### Exemplo de Entrada 1

```
5 cs.  
101 thqqxw.lui.qswer  
3 b-ylxmhzyjsys.virpbkr  
0
```

### Exemplo de Saída 1

```
cat  
this-is-a-secret  
beware.-dogs-barking
```