

14ª EDIÇÃO



MARATONA DE PROGRAMAÇÃO

# InterFatecs

2025

**FASE FINAL**

16 DE AGOSTO DE 2025

## CADERNO DE PROBLEMAS

**Fatec**  
Ferraz de  
Vasconcelos

**55** anos  
**CPs**  
Centro  
Paula Souza

**SÃO PAULO**  
GOVERNO DO ESTADO  
SÃO PAULO SÃO TODOS

**FUNDAÇÃO**  
**FAT**

**T2S**

**venturus**

**éboli**  
tecnologia

**Mestres**  
da **web**

**INPOWER**  
TECHSOLUTIONS

**FERRAZ**  
DE VASCONCELOS  
CIDADE DA UVA

**ASSOCIAÇÃO**  
**COMERCIAL**  
ACIFV FERRAZ DE VASCONCELOS

[WWW.INTERFATECS.COM.BR](http://WWW.INTERFATECS.COM.BR)

# 1 Instruções

Este caderno contém 14 problemas – identificados por letras de A até N, 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 9.4.0 (lembre-se que não existem bibliotecas do Windows)
- (b) Python 3.8.10
- (c) Java 11.0.22 (lembre-se que as classes devem estar fora de pacotes)

## Problema A

# The Last Player

*Arquivo fonte:* lastplayer.{ c | cpp | java | py }

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

In a playground game,  $n$  players stand in a circle, numbered from 1 to  $n$  in clockwise order. Starting from player 1, they begin counting clockwise. Every time the count reaches the  $k$ -th player, that player is removed from the circle. The counting resumes from the next player still in the circle. The process continues until only one player remains.

Your task is to determine the number of the last remaining player.

### Input

The input contains two integers:

`n k`

Where:

- $n$  — the number of players in the circle ( $1 \leq n \leq 100$ )
- $k$  — the step count for elimination ( $1 \leq k \leq 100$ )

### Output

Print a single integer:

`position`

Where `position` is the number of the last remaining player.

#### Exemplo de Entrada 1

5 2

#### Exemplo de Saída 1

3

#### Exemplo de Entrada 2

7 3

#### Exemplo de Saída 2

4

## Problema B

# Jokenpo

*Arquivo fonte:* jokenpo.{ c | cpp | java | py }  
*Autor:* Prof. Me. Sérgio Luiz Banin (Fatec São Paulo)

Beatriz and Artur are siblings, and they usually do a lot of things together. Sometimes, when they have different preferences, they play Rock, Paper, Scissors — known in Brazil as Jokenpo — to decide.

Since they are part of a school group focused on technology, they came up with an idea: to write a computer program to play the game for them. Typical laziness!

They started by creating a random generator to simulate the game turns. This generator uses an asterisk (\*) for Rock, a capital letter O for Paper, and a capital letter V for Scissors.

They're stuck. You're not. Time to step in, write the rest, and bring this game to life — before their laziness takes over!

### Input

The input consists of a single test case with multiple lines, each containing two characters separated by a blank space. The first character represents Beatriz's move, and the second represents Artur's. All characters are guaranteed to be valid: "\*"for Rock, "O"for Paper, and "V"for Scissors. The input ends with two hyphens separated by a blank space: - -.

### Output

The program must print either "BEATRIZ WIN"or "ARTUR WIN", depending on who has more points. Since the data is randomly generated, a tie is possible. In that case, the program must output the word: "TIE". Important: all output must be in capital letters, must not include quotation marks, and don't forget the line break at the end.

#### Exemplo de Entrada 1

```
V O
* *
O V
O *
- -
```

#### Exemplo de Saída 1

```
BEATRIZ WIN
```

#### Exemplo de Entrada 2

```
* V
V O
O *
* O
V *
O V
- -
```

#### Exemplo de Saída 2

```
TIE
```

## Problema C

### Cerco

*Arquivo fonte:* cerco.{ c | cpp | java | py }

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

Joãozinho vive em sua pequena propriedade rural na encosta das colinas esbornianas, onde costuma apreciar a bela paisagem marcada por imbondeiros centenários e palancas-negras pastando tranquilamente nos campos. Tudo corria bem... até que surgiram os Sneakys — pequenos, sorrateiros e incômodos invasores, conhecidos por se aproximarem sorrateiramente das casas e jardins.

Com receio de que esses visitantes indesejados possam perturbar a paz de sua propriedade, Joãozinho resolveu medir quantos Sneakys estão próximos o suficiente para representar uma ameaça imediata.

A casa de Joãozinho está localizada em uma coordenada fixa no plano cartesiano. Os Sneakys estão espalhados pelo mapa, cada um com suas próprias coordenadas. Joãozinho deseja saber quantos deles estão dentro ou exatamente na borda de seu raio de segurança.

#### Entrada

A entrada é composta por:

1. Uma linha com três inteiros:

onde:

- $x_j, y_j$  = posição da casa de Joãozinho ( $-10^4 \leq x_j, y_j \leq 10^4$ );
- $R$  = raio de segurança ( $1 \leq R \leq 10^4$ ).

2. Um inteiro  $N$  ( $1 \leq N \leq 10^5$ ) representando o número de Sneakys.

3.  $N$  linhas seguintes, cada uma contendo dois inteiros:

- $x_i y_i$

Representando as coordenadas de cada Sneaky ( $-10^4 \leq x_i, y_i \leq 10^4$ ).

#### Saída

Imprima apenas um número inteiro: A quantidade de Sneakys que estão a uma distância menor ou igual a  $R$  da casa de Joãozinho.

#### Exemplo de Entrada 1

```
0 0 5
5
1 2
4 0
6 0
-3 -4
0 5
```

#### Exemplo de Saída 1

```
4
```

## Problema D

# Estrada Perigosa

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

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

Após recuperar o carregamento valioso do cofre de **Cabeça de Ovo**, Joãozinho precisa transportá-lo com urgência até um porto seguro no extremo norte da **Esbórnica**. No entanto, a única rota viável até o destino obriga-o a atravessar a **Cypria** — um província fortemente monitorada e com **postos de vigilância infiltrados por espiões Sneakys**.

Embora Joãozinho possa cruzar algumas estradas vigiadas sem levantar suspeitas, existe um **limite máximo** de quantas dessas estradas perigosas ele pode percorrer em toda a sua jornada. Ele precisa encontrar **a rota mais rápida** que saia de sua cidade de origem, cruze a Cypria e chegue ao destino final respeitando essa restrição.

Dado um grafo não-direcionado ponderado representando as estradas da Esbórnica e da Cypria, determine o **menor tempo de viagem** para ir da cidade inicial  $S$  até a cidade final  $T$ , passando **obrigatoriamente** por pelo menos uma cidade da Cypria e usando **no máximo  $K$  estradas vigiadas pelos Sneakys**.

Se não houver rota possível que satisfaça as condições, imprima  $-1$ .

### Entrada

A entrada é composta por:

1. Uma linha com quatro inteiros:

$N$   $M$   $K$   $R$

Onde:

- $N$  = número total de cidades ( $2 \leq N \leq 10^5$ )
- $M$  = número de estradas ( $1 \leq M \leq 3 \times 10^5$ )
- $K$  = número máximo de estradas vigiadas permitidas ( $0 \leq K \leq 30$ )
- $R$  = número de cidades pertencentes à Cypria

2. Uma linha com  $R$  inteiros indicando os identificadores das cidades que pertencem à Cypria ( $1 \leq id \leq N$ ).

3. Uma linha com dois inteiros:

$S$   $T$

Onde:

- $S$  = cidade inicial ( $1 \leq S \leq N$ )
- $T$  = cidade destino ( $1 \leq T \leq N$ )

4. As próximas  $M$  linhas descrevem as estradas:

$u \ v \ w \ p$

Onde:

- $u, v$  = cidades conectadas ( $1 \leq u, v \leq N, u \neq v$ )
- $w$  = tempo de viagem ( $1 \leq w \leq 10^9$ )
- $p = 0$  se estrada é segura, 1 se estrada tem vigilância Sneaky

**Saída** Imprima um único número:

- O menor tempo possível para ir de  $S$  a  $T$ , passando por pelo menos uma cidade da Cypria, usando no máximo  $K$  estradas vigiadas.
- Se não houver rota possível, imprima  $-1$ .

**Exemplo de Entrada 1**

```
6 7 2 2
3 4
1 6
1 2 4 0
2 3 2 1
3 4 3 0
4 5 2 1
5 6 5 0
2 4 10 0
3 6 50 1
```

**Exemplo de Saída 1**

```
16
```



## Problema E

### Removido

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

Este problema foi removido.

[illegible]

Exemplo de Entrada 1	Exemplo de Saída 1
10 30	29



**Exemplo de Entrada 2**

1 20
------

**Exemplo de Saída 2**

19
----

**Exemplo de Entrada 3**

100 200
---------

**Exemplo de Saída 3**

199
-----

## Problema F

# Número de Homer

*Arquivo fonte:* numerohomer.{ c | cpp | java | py }

*Autor:* Prof. Me. Sérgio Luiz Banin (Fatec São Paulo)

Na cidade Springfield o Número de Homer, sigla  $HS$ , é um número real maior que zero muito usado para expressar o tempo necessário para que um homem adulto coma  $ND$  donuts (rosquinhas doces recheadas e com cobertura - muito populares em Springfield). A relação entre  $ND$  e  $HS$  é dada pela equação

$$ND = HS^{HS}$$

Por representar uma quantidade de rosquinhas, deve-se saber que, na prática,  $ND$  é um número inteiro maior que zero. Porém, neste problema faremos alguns cálculos como se esse número fosse um real. Você não deve se importar com isso, pois trata-se apenas de um método de cálculo, necessário à solução do problema.

A fórmula do Número de Homer foi estabelecida a partir de dados estatísticos disponíveis em Springfield referentes a um eminente cidadão adulto residente na cidade.

Devido à importância do Número de Homer para a economia de Springfield, o prefeito Joe Quimby contratou dois profissionais, você e eu, para desenvolver um método automatizado para calcular o tempo  $HS$  a partir de um número de rosquinhas fornecido  $ND$ .

Eu desenvolvi o método de cálculo, que explico a seguir. E você tem a incumbência de escrever um programa aplicando esse método. É importante ressaltar que o **cálculo computacional** de tudo que será apresentado a seguir deve ser feito usando números reais **em precisão dupla**.

### O método

É fácil saber que:

- se  $ND = 1$ , então  $HS = 1$  minuto, pois  $1^1 = 1$
- se  $ND = 4$ , então  $HS = 2$  minutos, pois  $2^2 = 4$
- se  $ND = 27$ , então  $HS = 3$  minutos, pois  $3^3 = 27$

Vejamos um caso de maior complexidade: se  $ND = 7$ , quanto vale  $HS$ ?

Como 7 está entre 4 e 27 sabemos que o tempo  $HS$  estará no intervalo entre 2 e 3 minutos. Agora o trabalho consiste em fazer uma série de aproximações partindo do intervalo (2, 3) estreitando esse intervalo até um limite considerado razoavelmente bom. Faremos o seguinte:

Dado o intervalo, calculamos seu termo médio e assumimos que seja  $HS$ , portanto:

$$HS = (2 + 3)/2 = 2.5 \text{ e usando esse valor calculamos:}$$

$$ND = 2.5^{2.5} = 9.88212 \text{ rosquinhas: uma diferença de } 2.88212$$

É aqui que surgem os números reais usados para representar rosquinhas, como mencionado antes. Não importa, use-os assim mesmo pois o objetivo é calcular o tempo.

Veja que neste tempo  $HS = 2.5$  podem ser comidas 2.88212 rosquinhas a mais do que o desejado. Como apenas 7 rosquinhas serão comidas, conclui-se que o tempo deve ser menor. Em seguida, atualizamos o intervalo para: (2, 2.5) minutos e repetimos o processo.

Com isso teremos:

$$HS = (2 + 2.5)/2 = 2.25 \text{ e usando esse valor calculamos:}$$

$$ND = 2.25^{2.25} = 6.20027 \text{ rosquinhas: uma diferença de } -0.79973$$

Desta vez o tempo ficou muito baixo, com uma diferença **a menor** de 0.79973. Então atualizamos o intervalo para (2.25, 2.5) e repetimos mais uma vez. O quadro a seguir demonstra o panorama geral desse cálculo:

```
intervalo (2, 3) -> metade = 2.5
rosquinhas comidas = 9.88212 diferença para 7 = 2.88212 (a maior)
intervalo (2, 2.5) -> metade = 2.25
rosquinhas comidas = 6.20027 diferença para 7 = 0.79973 (a menor)
intervalo (2.25, 2.5) -> metade = 2.375
rosquinhas comidas = 7.80191 diferença para 7 = 0.80191 (a maior)
intervalo (2.25, 2.375) -> metade = 2.3125
rosquinhas comidas = 6.94927 diferença para 7 = 0.05073 (a menor)
intervalo (2.3125, 2.375) -> metade = 2.34375
rosquinhas comidas = 7.36172 diferença para 7 = 0.36172 (a maior)
intervalo (2.3125, 2.34375) -> metade = 2.328125
rosquinhas comidas = 7.15215 diferença para 7 = 0.15215 (a maior)
intervalo (2.3125, 2.328125) -> metade = 2.3203125
rosquinhas comidas = 7.04989 diferença para 7 = 0.04989 (a maior)
intervalo (2.3125, 2.3203125) -> metade = 2.31640625
rosquinhas comidas = 6.99937 diferença para 7 = 0.00063 (a menor)
```

Figura F.1: Exemplo de cálculo do Número de Homer

Note no quadro que a magnitude (ignoramos o sinal) da diferença entre o  $ND$  desejado e o  $ND$  obtido vai caindo e pode-se provar (mas não vamos nos preocupar com isso agora) que esse método é convergente. Vamos estipular que quando essa diferença for menor que 0.001 (um milésimo) chegamos ao resultado desejado. Essa diferença menor que 0.001 é a precisão do nosso cálculo.

Neste exemplo o tempo  $HS = 2.31640625$  minutos é o resultado desejado. O programa deve apresentar na tela esse tempo no formato minuto:segundo:milésimo, ou seja, o formato será 0:00:000, no qual os segundos devem ter dois dígitos e os milésimos devem ter 3 dígitos.

Assim, para  $HS = 2.31640625$  deve ser apresentado  $HS = 2 : 18 : 984$ .

Outra possibilidade de solução para este problema é você usar logaritmos neperianos e a função W de Lambert. Porém, se essa for sua escolha você sabe bem o que está fazendo, então é com você... faz aí !!

### Entrada

A entrada contém diversas linhas, sendo que na última haverá o valor 0 (zero) indicando o final da entrada. Cada linha contém um inteiro  $ND$  que é a quantidade de rosquinhas a serem comidas.

### Saída

Para cada  $ND$  maior que zero da entrada deve ser calculado o tempo  $HS$ . Na saída esse tempo será mostrado no formato minuto:segundo(com dois dígitos):milésimo(com 3 dígitos), ou seja, formato 0:00:000. E não se esqueça de quebrar a linha na última saída.

**Exemplo de Entrada 1**

1  
2  
3  
4  
5  
6  
7  
15  
27  
30  
0

**Exemplo de Saída 1**

1:00:000  
1:33:574  
1:49:526  
2:00:000  
2:07:763  
2:13:908  
2:18:984  
2:42:788  
3:00:000  
3:03:001

## Problema G

### Pontes

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

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

O reino da Esbórnica é separado em duas regiões por um grande rio e, para conectar as regiões, o rei precisa construir pontes que possibilitem o escoamento de valiosos minerais da região de terras raras. Assim, o rei optou por construir dois tipos de pontes:

- Cada ponte **Estaiadas** custa  $A$  moedas de ouro;
- Cada ponte **Treliçadas** custa  $B$  moedas de ouro.

O rei tem um orçamento limitado de  $C$  moedas de ouro e deseja gastar todas elas na construção dessas pontes. Para cumprir esta missão, o rei não pode desconsiderar a preciosa ajuda de nosso amigo Joãozinho, que irá encontrar uma solução para gastar todo o orçamento, sem restar nada, com a menor quantidade de pontes possível, isto é, somando tanto as estaiadas quanto as treliçadas.

Caso existam duas soluções com a mesma quantidade de pontes, deve-se escolher aquela com a menor quantidade de pontes estaiadas. E caso não seja possível construir ponte alguma, exiba "IMPOSSIVEL", sem as aspas.

#### Entrada

A entrada consiste de uma única linha contendo três inteiros  $A$ ,  $B$  e  $C$  ( $1 \leq A, B \leq 10^6$ ;  $1 \leq C \leq 10^9$ )

#### Saída

Se for possível construir pelo menos uma ponte, exiba dois inteiros  $X$  e  $Y$ , separados por um único espaço, representando a quantidade de pontes estaiadas e treliçadas, respectivamente. Caso ponte alguma possa ser construídas, exiba "IMPOSSIVEL", em maiúsculo, sem acentuação e sem as aspas.

#### Exemplo de Entrada 1

4 7 23	4 1
--------	-----

#### Exemplo de Saída 1

#### Exemplo de Entrada 2

5 9 7	IMPOSSIVEL
-------	------------

#### Exemplo de Saída 2

## Problema H

# Mesada Diferente

*Arquivo fonte:* mesada.{ c | cpp | java | py }

*Autor:* Prof. Dr. Reinaldo Gen Ichiro Arakaki (Fatec São José dos Campos)

Um professor da Fatec decidiu organizar melhor a mesada dos seus filhos. Para isso, ele criou um sistema simples, mas com algumas regras curiosas.

Regras: Ele tem um valor total em reais que quer distribuir entre seus filhos.

Esse valor deve ser dividido igualmente entre eles, arredondando para baixo (divisão inteira).

O que sobrar vai para o último filho.

Cada filho deve gastar o valor recebido em categorias como: alimentação, roupas, cinema, etc.

Para cada categoria, ele só pode gastar R\$ 10, R\$ 20 ou R\$ 30 — nada além disso. E apenas estes valores podem ser usados e apenas uma única vez em cada categoria!

A ideia é sempre alocar o maior valor possível em cada categoria, seguindo a ordem definida (por exemplo: primeiro alimentação, depois roupas, depois cinema...).

Sua missão: Escreva um programa que:

Receba o valor total da mesada, o número de filhos e a lista de categorias (em ordem).

Divida o dinheiro conforme as regras acima. Para cada filho, distribua o valor recebido nas categorias, sempre tentando usar o maior valor possível permitido (10, 20 ou 30).

Mostre quanto cada filho recebeu e como esse valor foi dividido entre as categorias. Exemplo: Valor total da mesada R\$380,00, com 5 filhos, cada filho receberia R\$70,00, com exceção do último filho, que receberia R\$ 100,00 (acrescimo de R\$30,00 que sobraram do arredondamento). Considerando as 6 categorias de gastos teremos

Filho	Cat1	Cat2	Cat3	Cat4	Cat5	Cat6
1	30	30	10	0	0	0
2	30	30	10	0	0	0
3	30	30	10	0	0	0
4	30	30	10	0	0	0
5	30	30	30	10	0	0

### Entrada

Contém o número de filhos ( $2 \leq f \leq 100$ ), o montante da mesada  $m$  ( $100 \leq m \leq 2000$ , sem valor nas unidades) e categorias de gastos  $c$  ( $2 \leq c \leq 10$ ), separadas por espaço simples

### Saída

Imprimir uma linha para cada filho, contendo os valores para cada categoria de gasto, separadas por espaço simples

**Exemplo de Entrada 1**

5 380 6

**Exemplo de Saída 1**

30 30 10 0 0 0  
30 30 10 0 0 0  
30 30 10 0 0 0  
30 30 10 0 0 0  
30 30 30 10 0 0

**Exemplo de Entrada 2**

4 350 3

**Exemplo de Saída 2**

30 30 20  
30 30 20  
30 30 20  
30 30 30

**Exemplo de Entrada 3**

4 70 6

**Exemplo de Saída 3**

10 0 0 0 0 0  
10 0 0 0 0 0  
10 0 0 0 0 0  
30 10 0 0 0 0

## Problema I

# Cifra de César

*Arquivo fonte:* cifracesar.{ c | cpp | java | py }  
*Autor:* Prof. Me. Sérgio Luiz Banin (Fatec São Paulo)

César levava uma rotina bem agitada e precisava se comunicar rápido e com segurança. E calma aí, não é o Munarinho – dessa vez estamos falando do Imperador de Roma mesmo!

O Império era enorme e mensageiros a cavalo cruzavam as famosas estradas romanas levando mensagens pra todo lado. Mas havia sempre o risco de elas caírem nas mãos erradas... então, pra garantir o sigilo, entrou em cena um método bem esperto de codificação: a famosa Cifra de César.

A ideia era simples, mas genial: cada letra da mensagem original era trocada por outra, seguindo um número fixo de posições no alfabeto. Por exemplo, com um deslocamento de 2, A virava C, B virava D, C virava E... e por aí ia.

Assim, se alguém interceptasse a mensagem, só ia ver um monte de letras embaralhadas - a menos que soubesse qual era o número usado no deslocamento. Sem isso, decifrar era quase impossível na época.

Como exemplo, a seguir está uma frase e sua equivalente codificada com um deslocamento 2:

```
PROGRAMADOR, VOCE ESTA CONVOCADO PARA A FINAL DA MARATONA. AVE  
RTQITCOCFQT, XQEG GUV C EQPXQECFQ RCTC C HKPCN FC OCTCVQPC. CXG
```

Neste exemplo P virou R, R virou T, O virou Q e assim por diante. Espaços em branco e pontuação não contam e não são alterados. E antes que me pergunte, se houvesse letras Y e Z na mensagem, elas virariam A e B, respectivamente.

Para decodificar uma mensagem assim, o destinatário precisa conhecer o deslocamento usado e realizar o processo inverso. Neste exemplo R voltará para P, T voltará para R, Q voltará para O, B voltará para Z, A voltará para Y e assim por diante. Você já entendeu, certo?

Um segundo exemplo. Considere esta mensagem codificada: GYH MUAYGUPY

Investigando os caracteres finais com cuidado pode-se identificar que foi usado um deslocamento 20 para codificar AVE em UPY. Fazendo o processo inverso a mensagem decodificada fica: MESSAGESAVE

Em buscas arqueológicas recentes foram encontradas arcos com mensagens cifradas desse tempo. Os arqueólogos precisam de ajuda computacional para decodificá-las, pois não conhecem o valor do deslocamento usado em cada uma.

Sua tarefa é fazer um programa capaz de realizar a decodificação. Você também não sabe o deslocamento, mas existe uma pista: todas as mensagens de e para César terminam com a palavra AVE e todas as mensagens de e para o senado romano terminam com SPQR (Senatus Populusque Romanus). Não era muito esperto incluir essa saudação no final, mas os outros povos da época eram menos espertos que os romanos e agora isso vai ajudar pra caramba!!!

### Entrada

A entrada contém diversas linhas, cada uma com uma mensagem codificada. O conjunto de linhas termina com uma sequência de três asteriscos - \*\*\*. Garante-se que cada mensagem tem no máximo 200 caracteres. Garante-se também que a transcrição das mensagens do papel para o digital está correta, pois já foram



totalmente conferidas. Portanto, todas as mensagens presentes na entrada são passíveis de decodificação.

### Saída

Para cada mensagem codificada da entrada o programa deve exibir na saída a mensagem decodificada. E não se esqueça de quebrar a linha na última saída.

#### Exemplo de Entrada 1

```
BCDEFGBWF
BCDEFGTQRS
GYHMUAYGUPY
GYHMUAYGMJKL
YJSMFR FYJSHFT FAJ
YJSMFR FYJSHFT XUVW
***
```

#### Exemplo de Saída 1

```
ABCDEFAVE
ABCDEFSPQR
MENSAGEMAVE
MENSAGEMSPQR
TENHAM ATENCAO AVE
TENHAM ATENCAO SPQR
```

#### Exemplo de Entrada 2

```
RTQITCOCFQT, XQEG GUV C EQPXQECFQ RCTC C HKPCN FC OCTCVQPC. CXG
JOJDDJFN B QSPHSBNBDBP JNFEJBUBNFOUF.BWF
HJXFW IJYJWRNSF: SFT TZXR JWWFW JXYJ UWTLWFRF! FAJ
P TFOBEP DPOGJSNB, SFBMJABS FTUB QSPHSBNBDBP UFN HSBOEF VSHFODJB. TQRS
KPHQTOGO Q EQPUWN TQOCPQ UQDTG UGW RTQITGUUQ.URST
RES IWUYIGE Q HE UYIFVE HI PMRLE RS JMREP.WTUV
***
```

#### Exemplo de Saída 2

```
PROGRAMADOR, VOCE ESTA CONVOCADO PARA A FINAL DA MARATONA. AVE
INICIEM A PROGRAMACAO IMEDIATAMENTE.AVE
CESAR DETERMINA: NAO OUSEM ERAR ESTE PROGRAMA! AVE
O SENADO CONFIRMA, REALIZAR ESTA PROGRAMACAO TEM GRANDE URGENCIA. SPQR
INFORMEM O CONSUL ROMANO SOBRE SEU PROGRESSO.SPQR
NAO ESQUECAM DA QUEBRA DE LINHA NO FINAL.SPQR
```

## Problema J

### Sorvete

*Arquivo fonte:* sorvete.{ c | cpp | java | py }

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

Durante um verão escaldante, uma sorveteria decidiu oferecer uma degustação especial. Há uma fila de clientes esperando para experimentar diferentes sabores de sorvete. Cada cliente tem uma **temperatura ideal** para o sorvete, representada por um número inteiro.

Para garantir a melhor experiência possível, o atendente precisa formar grupos de clientes que serão atendidos ao mesmo tempo. No entanto, como os sorvetes de um grupo são preparados juntos, todos os clientes do mesmo grupo devem aceitar uma diferença de temperatura de no máximo  $D$  graus entre o mais quente e o mais frio.

Seu objetivo é ajudar o atendente a descobrir o **menor número de grupos** necessários para que todos os clientes sejam atendidos, respeitando a restrição de variação de temperatura.

**Entrada** A primeira linha da entrada contém dois inteiros  $N$  e  $D$  ( $1 \leq N \leq 10^5$ ,  $0 \leq D \leq 10^9$ ), onde:

- $N$  é o número de clientes;
- $D$  é a diferença máxima de temperatura que pode existir entre os sorvetes de um mesmo grupo.

As próximas  $N$  linhas contêm um inteiro  $t_i$  ( $0 \leq t_i \leq 10^9$ ), representando a temperatura ideal do  $i$ -ésimo cliente.

#### Saída

Imprima um único número inteiro representando o **menor número de grupos** necessários para que todos os clientes recebam seu sorvete com uma variação de temperatura de no máximo  $D$  dentro do grupo.

#### Exemplo de Entrada 1

6 2 3 1 4 1 5 9	3
-----------------------------------	---

#### Exemplo de Saída 1

#### Exemplo de Entrada 2

6 0 3 1 4 1 5 9	5
-----------------------------------	---

#### Exemplo de Saída 2

## Problema K

# Telesbórnica

*Arquivo fonte:* telesbornia.{ c | cpp | java | py }

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

A principal operadora de telefonia da Esbórnica (Telesbórnica) permite cadastrar padrões numéricos para facilitar a memorização de longas sequências. Cada número discado pode ser convertido em uma sequência de padrões pré-definidos.

Joãozinho, o nosso sagaz amigo, é convidado a elaborar um programa para determinar de quantas formas diferentes é possível decompor completamente um número discado usando os padrões cadastrados. Nosso amigo pede sua ajuda para solucionar este problema.

Cada padrão pode ser usado quantas vezes forem necessárias, e todas as formas de segmentação possíveis devem ser consideradas. A sequência deve ser totalmente utilizada; prefixos ou sufixos parciais não são permitidos.

**Entrada** A entrada é composta por duas partes:

- A primeira linha contém uma string  $S$  com até  $10^5$  dígitos decimais (0 a 9).
- A segunda linha contém um inteiro  $N$  ( $1 \leq N \leq 10^4$ ), o número de padrões cadastrados.
- Em seguida, seguem  $N$  linhas, cada uma contendo uma string de 1 a 5 dígitos representando um padrão válido.

### Saída

O programa deve imprimir um único número inteiro: a quantidade total de maneiras distintas de decompor a string  $S$  usando os padrões fornecidos.

#### Exemplo de Entrada 1

```
123123
3
123
1
23
```

#### Exemplo de Saída 1

```
2
```

#### Exemplo de Entrada 2

```
1234
3
1
2
3
```

#### Exemplo de Saída 2

```
0
```

**Exemplo de Entrada 3**

```
11111
2
1
11
```

**Exemplo de Saída 3**

```
8
```

**Exemplo de Entrada 4**

```
2020
1
2020
```

**Exemplo de Saída 4**

```
1
```

## Problema L

# Sequência de Conway

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

Autor: Prof. Dr. Reinaldo Gen Ichiro Arakaki (Fatec São José dos Campos)

Um professor de Matemática Discreta da Fatec apresentou em sua aula a curiosa Sequência de Conway. A sequência é formada da seguinte maneira:

A primeira linha é 1.

A linha seguinte descreve a anterior: “um 1”  $\rightarrow$  11

A próxima: “dois 1s”  $\rightarrow$  21

“um 2, um 1”  $\rightarrow$  1211

“um 1, um 2, dois 1s”  $\rightarrow$  111221

... e assim por diante.

Ou seja, cada termo descreve quantas vezes cada dígito aparece consecutivamente na linha anterior.

1,  
11,  
21,  
1211,  
111221,  
312211,  
13112221,  
1113213211,  
31131211131221

Para variar o professor pediu para os alunos programarem esta sequência. Dado a linha, produzir a sequência da linha

### Entrada

A entrada consiste no número da linha ( $2 \leq l \leq 30$ )

### Saída

Imprimir a sequência ( $1 \leq s \leq 10^{4500}$ ).

#### Exemplo de Entrada 1

5

#### Exemplo de Saída 1

111221

#### Exemplo de Entrada 2

6

#### Exemplo de Saída 2

312211

#### Exemplo de Entrada 3

9

#### Exemplo de Saída 3

31131211131221

## Problema M

### ScoreBoard - Final

Arquivo fonte: scoreboard.{ c | cpp | java | py }  
Autor: Prof. Me. Sérgio Luiz Banin (Fatec São Paulo)

Na primeira etapa da Maratona InterFatecs, vocês tiveram que criar um programa para classificar os times para a final. Esse desafio gerou algumas dúvidas, principalmente sobre como a pontuação de cada equipe é calculada.

A quantidade de problemas resolvidos é um critério claro, mas o fato é que muitos times acertam a mesma quantidade. Por isso, é preciso de um segundo critério: o tempo. Esse critério é simples, mas nem sempre fica claro como é feito o cálculo.

Quando um time acerta um problema, registra-se o minuto em que isso acontece. Depois, somam-se todos esses minutos, que vão formar o tempo total do time. A esse tempo, adicionam-se 20 minutos de multa para cada tentativa errada, mas só se o time tiver conseguido resolver o problema. Para os problemas não resolvidos, as tentativas erradas não contam.

Veja este exemplo em que um time resolveu 4 problemas:

	Tentativas	Tempo	Resolvido	Pontuação
Problema A	1	30	Sim	30
Problema B	0	0	Não	0
Problema C	3	0	Não	0
Problema D	3	210	Sim	210 + 40
Problema E	1	70	Sim	70
Problema F	2	150	Sim	150 + 20
<b>Totais</b>			<b>4</b>	<b>520</b>

Figura M.1: Apuração dos pontos de um time

Para definir a classificação dos times na Maratona são usados esses dois parâmetros: a quantidade de problemas em ordem decrescente e o tempo em ordem crescente. O campeão será o time que resolver o maior número de problemas no menor tempo.

Muito bem, agora que você já entendeu o processo, deve escrever um programa que leia os dados envolvidos e produza a classificação dos times.

#### Entrada

A entrada contém um único caso de teste. Na primeira linha há um número inteiro  $NP$  que representa o número de problemas da Maratona ( $4 \leq NP \leq 20$ ). Na segunda linha há um número inteiro que representa a quantidade de times  $QT$  participantes da Maratona ( $6 \leq QT \leq 1000$ ).

Em seguida há  $QT$  pares de linhas. A primeira linha de cada par contém dois strings de no máximo 100 caracteres cada: o nome do time e o nome da Fatec, separados pelo caractere pipe ". A segunda linha contém  $NP$  pares de inteiros representando a solução dos problemas, sendo que o primeiro inteiro é o número de tentativas e o segundo é o tempo da solução correta. Se este segundo inteiro for zero o problema não foi resolvido.

## Saída

A saída deve conter uma linha para cada time presente na entrada. Cada linha precisa exibir quatro informações: o nome do time, o nome da Fatec, o número de problemas resolvidos e o tempo total. O formato da saída deve seguir literalmente o exemplo mostrado. Entre o nome do time e o da Fatec deve haver um hífen com um espaço em branco antes e depois dele. Após o nome da Fatec deve haver um espaço em branco e em seguida entre parênteses devem estar o número de problemas e o tempo.

As linhas devem estar ordenadas do melhor para o pior time. O primeiro critério de ordenação é a quantidade de problemas resolvidos em ordem decrescente. Havendo empate nesse critério o desempate deve ser feito usando o tempo em ordem crescente.

Em caso muito remoto de dois times empatarem segundo os dois critérios, a ordem deve ser alfabética pelo nome do time.

### Exemplo de Entrada 1

```
6
8
Time 01|Fatec A
0 0 1 0 4 0 1 0 3 170 2 0
Time 02|Fatec A
0 0 1 90 1 50 0 0 1 180 0 0
Time 03|Fatec B
0 0 2 80 0 0 1 50 2 110 2 170
Time 04|Fatec B
4 150 1 20 2 70 2 0 1 40 2 110
Time 05|Fatec B
0 0 1 60 1 30 0 0 1 160 1 100
Time 06|Fatec C
0 0 1 120 0 0 0 0 3 320 0 0
Time 07|Fatec C
0 0 1 50 1 80 0 0 1 120 0 0
Time 08|Fatec D
0 0 1 40 2 140 0 0 3 80 0 0
```

### Exemplo de Saída 1

```
Time 04 - Fatec B (5,490)
Time 05 - Fatec B (4,350)
Time 03 - Fatec B (4,470)
Time 07 - Fatec C (3,250)
Time 02 - Fatec A (3,320)
Time 08 - Fatec D (3,320)
Time 06 - Fatec C (2,480)
Time 01 - Fatec A (1,210)
```

## Problema N

# Os Cheques do Mário

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

Autor: Prof. Me. Lucio Nunes de Lira (Fatecs Diadema, Ferraz de Vasconcelos e São Caetano do Sul)

Você conhece o Mário? Que Mário?! Mário é diferenciado, pois mesmo com a modernidade das transferências bancárias eletrônicas, praticamente instantâneas como o Pix, ainda gosta de usar cheques em suas compras. Você não sabe o que é um cheque? Vamos voltar um pouco no tempo.

Há muitos séculos as pessoas perceberam que seria mais fácil e seguro manter seus bens em um local protegido e com pessoas dedicadas a armazená-los, algo próximo ao que hoje chamamos de "bancos". Com os bancos, se tornou viável realizar transações financeiras por meio de "títulos de crédito", que são documentos que expressam a existência de uma dívida a ser paga e um valor a ser recebido por alguém.

O cheque é um título de crédito! Uma pessoa emite um cheque, com determinado valor, para que outra leve-o até uma instituição financeira específica e receba esse valor. Simples assim! Talvez nem tanto, estamos simplificando, pois existem riscos e várias especificidades no preenchimento dos cheques. Por exemplo, o valor deve ser escrito numericamente (usando algarismos) e, também, por extenso (usando letras), uma forma de aumentar a segurança contra adulteração e reduzir a chance de erros por distração.

Quem recebe o cheque tem que confiar que haverá fundos, isto é, que a pessoa que emitiu a dívida terá saldo em sua conta para que esse valor possa ser descontado. Como Mário emite muitos cheques, ele quer a sua ajuda para criar um programa de computador que receba os valores dos cheques e indique qual deverá ser o valor mínimo em sua conta para que todos possam ser devidamente descontados.

### Entrada

A entrada é um único caso de teste com uma ou mais linhas. Cada linha é uma *string* representando um valor monetário  $V$  em reais (R\$), por extenso, em português e terminando com um ponto final, simbolizando um cheque emitido por Mário. A quantidade de linhas é indeterminada, as entradas acabam com o final do arquivo. Os valores dos cheques estarão no intervalo ( $R\$ 0,01 \leq V \leq R\$ 999.999.999.999.999,99$ ), serão no máximo 1000 cheques e a soma não ultrapassará  $R\$ 999.999.999.999.999,00$ .

A grafia das palavras pode estar em maiúsculo, minúsculo ou uma intercalação de ambos. Os valores estarão escritos corretamente, considerando ortografia, flexão de quantidade (singular e plural) e a ordem, isto é, primeiro trilhões, depois bilhões, milhões, milhares etc., com reais antecedendo os centavos, quando houver. Porém, sem acentuação. Entre as palavras haverá só um espaço. Veja os casos de teste de exemplo.

### Saída

Imprima o valor mínimo, em reais, que Mário precisará ter em sua conta para honrar todos os cheques que emitiu. O valor deve estar na exata configuração dos casos de teste de exemplos: (I) sem acentuação, (II) com o símbolo monetário de real, (III) com separação de milhares por ponto, (IV) com a parte decimal separada da parte inteira usando uma vírgula, (V) com duas casas decimais, (VI) com ponto final.

#### Exemplo de Entrada 1

```
um real.  
cem reais e um centavo.  
mil e cento e tres reais e quatorze centavos.
```

#### Exemplo de Saída 1

```
Mínimo: R$ 1.204,15.
```



### Exemplo de Entrada 2

dois mil reais.  
mil reais.  
cinquenta e tres centavos.  
vinte e cinco mil reais e noventa e seis centavos.

### Exemplo de Saída 2

Mínimo: R\$ 28.001,49.

### Exemplo de Entrada 3

cem reais.  
cento e um reais.  
mil reais.  
um milhao de reais.  
um bilhao de reais.  
um trilhao de reais.  
um centavo.  
dois centavos.

### Exemplo de Saída 3

Mínimo: R\$ 1.001.001.001.201,03.

### Exemplo de Entrada 4

trinta e oito milhoes e duzentos reais.  
cem reais.

### Exemplo de Saída 4

Mínimo: R\$ 38.000.300,00.

### Exemplo de Entrada 5

noventa e nove trilhoes de reais e tres centavos.  
um centavo.

### Exemplo de Saída 5

Mínimo: R\$ 99.000.000.000.000,04.

### Exemplo de Entrada 6

nove trilhoes e oitocentos e oitenta e oito bilhoes e setecentos e setenta e sete milhoes e seiscentos e sessenta e seis mil e quinhentos e quarenta e tres reais e vinte e um centavos.

### Exemplo de Saída 6

Mínimo: R\$ 9.888.777.666.543,21.

### Exemplo de Entrada 7

Um MILHao E sete reais e DOIS centavos.  
DOIS REAIS e noventa e OITO CENTAVOs.

### Exemplo de Saída 7

Mínimo: R\$ 1.000.010,00.