Problema A

The Mayans

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

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

The Mayan civilization, one of the most fascinating cultures in history, flourished in Mesoamerica for over 3,000 years. Known for their advanced knowledge of astronomy, mathematics, and writing, the Maya built incredible cities with stunning pyramids and temples. Their vibrant culture, rich in art, architecture, and spirituality, continues to captivate us today. The Maya's connection to the cosmos and their deep respect for nature made them a truly unique civilization, whose legacy still influences modern understanding of the world.

The Mayan numerical system was highly sophisticated and based on a vigesimal (base-20) system, which means it used the number 20 as its base, unlike our decimal (base-10) system. This system combined dots, bars, and a shell symbol to represent numbers.

• **Dots** represented values of 1.
• **Bars** represented values of 5.
• **The shell symbol** represented 0, one of the earliest uses of a zero in history, which was a remarkable concept for the time.

For example, a dot and a bar together would represent *6 (1+5)*. The Mayans also used positional notation, where the value of a symbol depended on its position, much like how our place values work in the decimal system.

This system was used for various purposes, including their intricate calendar systems, where it helped them track time, celestial events, and even create long-count calendars that could measure thousands of years. It was a key part of their advanced knowledge in mathematics and astronomy.

Since their system is based on *20*, they used twenty distinct symbols to represent values under twenty, including one for zero. The Figure A.1 shows these symbols.

Numbers greater than *19*, which require more than one digit, were written vertically in powers of twenty. The Mayans used powers of twenty, just as we, in Western culture, use powers of ten, a system that originated with the Hindu-Arabic numeral system.

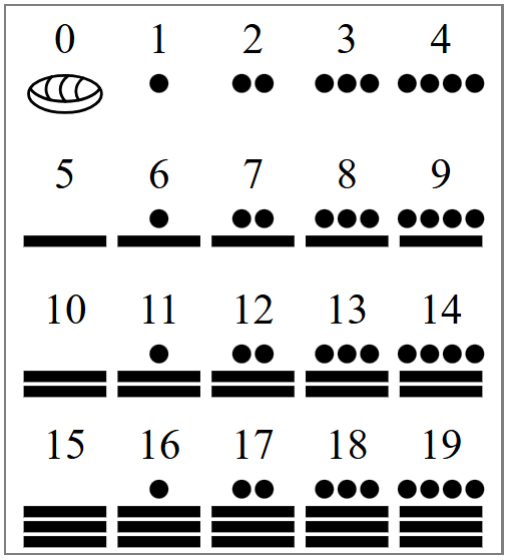


Figura A.1: The Mayan Symbols

So have a look at Figure A.2 for some examples:

Positional Coefficients	Example 1	Example 2	Example 3
$20^3 = 8000$			
$20^2 = 400$			
$20^1 = 20$			
$20^0 = 1$			
	1 x *20* + **7** x *1* Total = 27	**6** x *400* + **10** x *20* +**14** x *1* Total = 2614	**1** x *8000* + **8** x *1* Total = 8008

Figura A.2: Examples of Mayan System

Now that you know about the Mayan numeric system, your task is to write a computer program that converts Mayan numbers to decimal numbers. For the input of this program, the Mayan symbols, represented by dots, hyphens (for bars), and asterisks (for shells, which represent zero), will be input horizontally. For this conversion, the Mayan symbols must be read from bottom to top, with the characters written from left to right. When a Mayan number contains more than one digit, a single blank space will separate the digits.

For ilustration see Figure A.3.

The symbol  will be represented by	. (one dot)
The symbol  will be represented by	-.. (one hyphen and two dots)
The symbol  will be represented by	-- (two hyphens)
The symbol  will be represented by	--.... (two hyphens and four dots)
The symbol  will be represented by	* (one asterisk)

Figura A.3: Conversion from Pictographic to Digital

Input

The input consists of several lines, ending when a Mayan zero is encountered, e.g., when an asterisk is found. Each line contains a sequence of valid Mayan numeric symbols, limited to a maximum of *8* symbols. This means that the maximum possible decimal value is Vmax = $20^8 - 1$.

Each line is guaranteed to contain a valid combination of characters that results in a natural number.

Output

For each input line, convert and print the value represented by the symbols to our decimal system. The final asterisk must be converted as well, so the output ends with a zero, followed by a newline character.

Exemplo de Entrada 1	Exemplo de Saída 1
. --.. -. -- --.... . * * -... .... --- - - - -. -. -. -. -. -. -. -. *	32 2614 8008 4 8421 15 2105 8084210526 0

Problema B

Scoreboard Classificatória

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

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

Como você bem sabe Maratona InterFatecs é um evento organizado em duas etapas. Hoje está acontecendo a etapa classificatória e a etapa final será no mês de agosto/2025 na Fatec de Ferraz de Vasconcelos.

Para que tudo corra bem, os profs Rogério e Lúcio, da Fatec Ferraz, estão se desdobrando nos preparativos. Eles conversaram com o prof. Sakaue, responsável pelo BOCA da InterFatecs, e descobriram que um dos gargalos desta etapa de hoje é a apuração dos times que se classificaram para a etapa Final.

Assim, eles estão aqui para pedir a sua ajuda. Eles precisam de um programa de computador que receba como dado de entrada o desempenho de cada time e com base nisso e nos critérios de classificação produza uma lista de classificados, bem como uma lista de espera que poderá ser usada em caso de desistência de algum time, dentre os classificados.

Critérios de Classificação

É simples esse critério e é composto destes itens:

1. Times que não resolveram nenhum problema são desclassificados;

2. A Fatec que sediará a etapa final classifica seu time que melhor pontuou, mais alguns times cuja quantidade é definida como "vagas extras da sede". Se a Fatec sede não tiver times suficientes para preencher todas as vagas a que tem direito, então as vagas não preenchidas serão usadas para os times que se enquadram no critério 4;

3. O time mais bem pontuado de cada Fatec se classifica;

4. As vagas remanescentes são preenchidas pelos times mais bem pontuados e que não foram contemplados nos itens 2 e 3 acima, limitados ao número de vagas disponível.

Para calcular a pontuação são usados dois parâmetros na seguinte ordem:

- Maior quantidade de problemas resolvidos;
- Em caso de empate no item acima, considera-se o menor tempo total apurado na solução dos problemas.

Considere o pequeno exemplo mostrado no quadro a seguir, supondo que a Sede seja a "Fatec E", com 8 vagas totais e 2 vagas extras para a sede.

- Pelo item 1. dos critérios os times 08 (Fatec D) e 14 (Fatec F) estão desclassificados;
- Pelo item 2. dos critérios os times 10 (E), 09 (E) e 11 (E) estão classificados, nessa ordem;
- Pelo item 3. dos critérios os times 05 (B), 02 (A), 07 (C) e 13 (F) estão classificados, nessa ordem;
- Pelo item 4. dos critérios o time 03 (B) está classificado;
- A lista de espera será constituída pelos times: 01 (A), 06 (C), 04, (B), 12 (E) nessa ordem;

Agora que você já conhece as regras de classificação sua tarefa é escrever um programa capaz de produzir esse resultado.

Resultados

Time	Fatec	Resolvidos	Tempo
Time 01	Fatec A	3	150
Time 02	Fatec A	3	110
Time 03	Fatec B	3	100
Time 04	Fatec B	2	280
Time 05	Fatec B	4	210
Time 06	Fatec C	2	90
Time 07	Fatec C	3	151
Time 08	Fatec D	0	0
Time 09	Fatec E	2	130
Time 10	Fatec E	3	140
Time 11	Fatec E	2	330
Time 12	Fatec E	1	180
Time 13	Fatec F	1	200
Time 14	Fatec F	0	0

Classificação

Time	Fatec	Resolvidos	Tempo
Time 05	Fatec B	4	210
Time 03	Fatec B	3	100
Time 02	Fatec A	3	110
Time 10	Fatec E	3	140
Time 01	Fatec A	3	150
Time 07	Fatec C	3	151
Time 06	Fatec C	2	90
Time 09	Fatec E	2	130
Time 04	Fatec B	2	280
Time 11	Fatec E	2	330
Time 12	Fatec E	1	180
Time 13	Fatec F	1	200
Time 08	Fatec D	0	0
Time 14	Fatec F	0	0

Figura B.1: Exemplo de Classificação

Entrada

A entrada é constituída por um caso de teste com diversas linhas de dados. A primeira linha contém um String, com no máximo 25 caracteres incluindo eventuais espaços em branco, que é o nome da Fatec Sede, escrito de modo exatamente igual ao que aparecerá em seguida no cadastro dos times.

Na segunda linha estão dois números inteiros maiores que 1. O primeiro valor V $(1 < V \leq 100)$ é a quantidade de vagas totais da etapa final. O segundo valor VeS $(1 < VeS \leq 6)$ é a quantidade de vagas extras a que a sede tem direito, ou seja, o número total de vagas da sede é $VeS + 1$.

Na terceira linha está um número inteiro NT $(10 < NT \leq 500)$ que é a quantidade total de times participantes da maratona.

Nas demais NT linhas teremos os dados dos times. São quatro informações separadas pelo caractere pipe "|" (aquela barra reta vertical que em geral é pouco usada): string com o nome do time com o máximo de 25 caracteres, string com o nome da Fatec do time com o máximo de 25 caracteres, inteiro com a quantidade de problemas resolvidos, inteiro com o tempo total usado nas resoluções.

Saída

A saída é constituída de 3 partes

Primeira parte

Contém NT times os classificados para a final e se inicia com o texto literal "Classificados para a Final" na primeira linha (sem as aspas). Em seguida são exibidos todos os times em ordem alfabética e no formato:

XXXtimeXXX – XXXfatecXXX (9,999)

onde:

XXXtimeXXX é o nome do time

XXXfatecXXX é o nome da Fatec do time

9 é a quantidade de problemas resolvidos pelo time

999 é o tempo total usado pelo time nas soluções

Entre XXXtimeXXX e XXXfatecXXX deve haver um hífen, antecedido e sucedido por espaço em branco

Após XXXfatecXXX deve haver um espaço em branco

E o par de informações 9,999 deve estar entre parênteses

Entre a primeira e segunda partes deve ser deixada uma linha em branco.

Segunda parte

É a lista de espera e contém todos os demais times que resolveram pelo menos um problema. Esta seção se inicia com o texto literal “Lista de Espera” seguida dos times no mesmo formato descrito acima e ordenados por ordem da pontuação obtida na competição, formada por quantidade decrescente de problemas resolvidos e valor crescente de tempo total.

Entre a segunda e terceira partes deve ser deixada uma linha em branco.

Terceira parte

É a lista de desclassificados e contém os times que não resolveram nenhum problema. Esta seção se inicia com o texto literal “Desclassificados” seguida dos times em ordem alfabética pelo nome do time e no mesmo formato descrito acima.

Após essa terceira parte deve-se pular uma linha e acrescentar o texto literal “Apuracao concluida!” seguido de quebra de linha.

Atenção: ao imprimir os literais na saída, prestem atenção nas letras maiúsculas e minúsculas, e também nas linhas que devem ser deixadas em branco.

Exemplo de Entrada 1

```
Fatec E
8 2
14
Time 01|Fatec A|3|150
Time 02|Fatec A|3|110
Time 03|Fatec B|3|100
Time 04|Fatec B|2|280
Time 05|Fatec B|4|210
Time 06|Fatec C|2|90
Time 07|Fatec C|3|151
Time 08|Fatec D|0|0
Time 09|Fatec E|2|130
Time 10|Fatec E|3|140
Time 11|Fatec E|2|330
Time 12|Fatec E|1|180
Time 13|Fatec F|1|200
Time 14|Fatec F|0|0
```

Exemplo de Saída 1

```
Classificados para a Final
Time 02 - Fatec A (3,110)
Time 03 - Fatec B (3,100)
Time 05 - Fatec B (4,210)
Time 07 - Fatec C (3,151)
Time 09 - Fatec E (2,130)
Time 10 - Fatec E (3,140)
Time 11 - Fatec E (2,330)
Time 13 - Fatec F (1,200)

Lista de Espera
Time 01 - Fatec A (3,150)
Time 06 - Fatec C (2,90)
Time 04 - Fatec B (2,280)
Time 12 - Fatec E (1,180)

Desclassificados
Time 08 - Fatec D (0,0)
Time 14 - Fatec F (0,0)

Apuracao concluida!
```

Problema C

Decisão do Agricultor

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

Autor: Prof. Dr. Wiliam Galvão (Fatec Santana de Parnaíba)

No ano de 2030, em meio às imprevisíveis mudanças climáticas, o pequeno agricultor Seu João enfrenta um desafio crítico: decidir o momento ideal para regar sua plantação, visando evitar o desperdício de água e, ao mesmo tempo, garantir a produtividade da colheita. Para auxiliá-lo, foi desenvolvido um sistema de inteligência artificial chamado AGRO-DECISOR, capaz de fornecer recomendações precisas com base em dados coletados em tempo real.

O AGRO-DECISOR utiliza uma árvore de decisão simples para determinar se a irrigação é necessária. A decisão é baseada em três variáveis ambientais:

1. Temperatura Atual (em graus Celsius, valor real): Indica o risco de evaporação da água no solo.
2. Umidade do Solo (em porcentagem, valor real): Reflete a quantidade de água presente no solo e disponível para as plantas.
3. Previsão de Chuva (valor inteiro: 0 para "Não há previsão de chuva", 1 para "Há previsão de chuva"): Indica se há expectativa de chuva para o período.

As regras de decisão implementadas no sistema são as seguintes:

> **Regra 1**: Se a Previsão de Chuva for igual a 1 (ou seja, há previsão de chuva), a recomendação é NAO REGAR, pois a chuva natural suprirá a necessidade hídrica.
>
> **Regra 2**: Caso contrário (se a Previsão de Chuva for igual a 0):
>
> > **Sub-regra 2.1**: Se a Temperatura for maior que 30.0°C E a Umidade do Solo for menor que 50.0%, a recomendação é REGAR. Esta condição indica solo seco e calor extremo, necessitando de irrigação.
> >
> > **Sub-regra 2.2**: Em todas as outras condições não cobertas pelas regras anteriores (ou seja, se a Temperatura não for maior que 30.0°C ou a Umidade do Solo não for menor que 50.0%), a recomendação é NAO REGAR.

Sua tarefa é implementar o núcleo do AGRO-DECISOR. Dado um conjunto de N leituras de sensores (contendo Temperatura, Umidade do Solo e Previsão de Chuva), seu programa deve classificar cada leitura, indicando se a ação recomendada é "*REGAR*" ou "*NAO REGAR*".

Entrada

A entrada consiste em múltiplas linhas. A primeira linha da entrada contém um único inteiro N ($1 \leq N \leq 1000$), representando o número de conjuntos de leituras de sensores que serão processados. As N linhas seguintes contêm, cada uma, três valores separados por espaço: T, U e P.

- T é um valor real representando a Temperatura em graus Celsius.
- U é um valor real representando a Umidade do Solo em porcentagem.

- P é um valor inteiro (0 ou 1) representando a Previsão de Chuva.

Saída

Para cada um dos N conjuntos de leituras da entrada, seu programa deve imprimir uma única linha contendo a recomendação do sistema: “REGAR” ou “NAO REGAR”, de acordo com as regras descritas.

Restrições

- $1 \leq N \leq 1000$
- Os valores de Temperatura e Umidade do Solo serão números reais que podem ser representados por tipos de ponto flutuante padrão (ex: float em Python, double em Java/C++).
- O valor da Previsão de Chuva será sempre 0 ou 1.

Exemplo de Entrada 1	**Exemplo de Saída 1**
3 35.0 40.0 0 28.0 60.0 1 32.0 45.0 0	REGAR NAO REGAR REGAR

Exemplo de Entrada 2	**Exemplo de Saída 2**
2 25.0 60.0 1 28.0 40.0 1	NAO REGAR NAO REGAR

Problema D

À Helicon e além

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

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

Trantor não era mais a joia do Império. Onde antes se erguiam torres infinitas e corredores administrativos fervilhavam com decisões que moldavam a galáxia, agora restavam ruínas silenciosas e vegetação selvagem forçando passagem entre estruturas corroídas. Era ali, entre os escombros da outrora majestosa Biblioteca Imperial, que Arcadia Darell caminhava com determinação. Seus olhos examinavam um terminal semi-enterrado, vestígios da civilização que Hari Seldon previra desmoronar.

Arcadia, agora mais velha e mais decidida, sabia que havia mais no Plano Seldon do que o mundo jamais soubera. A psico-história, essa ciência quase mítica que previa os rumos da humanidade em larga escala, ainda era envolta em mistério — e ela, neta de Bayta Darell, decidira que era hora de entender seus verdadeiros contornos. Mas para isso, teria que sair de Trantor.

Sua nave, Lumen, repousava silenciosa nos arredores da cidade devastada. Equipada com motores gravitacionais, ela era uma maravilha da engenharia oculta: uma tecnologia raríssima, capaz de manipular os campos gravitacionais ao redor da nave para gerar movimento sem propulsores tradicionais. Sem combustão, sem vibração — apenas um silêncio elegante enquanto dobrava espaço e inércia à sua vontade. A bordo, Arcadia podia acelerar a velocidades incríveis, pairar sobre mundos sem perturbar a atmosfera, ou descer suavemente mesmo nos ambientes mais inóspitos.

Um dos grandes desafios de Arcadia é encontrar as coordenadas certas para que a nave, que funciona de maneira autônoma, possa traçar rotas levando assim Arcadia aos lugares mais inóspitos do universo. Para isso, ela está nas ruínas da Biblioteca Imperial em Trantor vasculhando manuais e informações que vieram de Terminus séculos atrás quando as primeiras naves gravitacionais surgiram, ela achou vários fragmentos dos manuais de operações e configuração de rotas, cada rota que ela encontra é uma sequência de valores que ela deverá inserir no console navegador nave, entretanto essas sequências de valores foram contaminadas com informação espúria, uma vez que os cubos de informação quântica que armazenam essas rotas estão deteriorados pelo tempo. Entretanto, ela conseguiu descobrir que dada uma sequência qualquer, os valores que fazem parte de um rota, ou seja, não são ruídos de informação podem estar separados por k passos de distância, e assim ela deve encontrar a maior soma de uma sequência que esteja a k passos de distância considerando todos os valores possíveis para $1 \leq k \leq N/2$, onde N é o tamanho das sequência encontrada que reúne valores verdadeiros e ruidosos.

Sua jornada seria longa. Pretendia começar em Helicon, planeta natal de Seldon, em busca de vestígios acadêmicos esquecidos. De lá, seguiria para Terminus, onde a Primeira Fundação fora plantada — talvez ainda houvesse registros não corrompidos pela política. Planejava visitar Rossem, onde Seldon surgira em projeção décadas atrás, e Gaia, cuja consciência coletiva parecia compreender a psico-história de um modo totalmente diferente. E, se restasse tempo e coragem, arriscaria uma passagem discreta por Kalgan, antigo domínio do Mulo, na esperança de entender como uma anomalia tão poderosa havia alterado o curso do destino galáctico.

Na cabine de comando, ao observar o céu encoberto de Trantor, Arcadia registrou em seu diário:

Se a psico-história ainda vive, ela deixou rastros. E se o futuro pode ser guiado, preciso primeiro entender o passado.

Entrada

A primeira linha contém um número inteiro N $(1 \leq N \leq 10^5)$, o tamanho do vetor. A segunda linha contém N inteiros $A_1, A_2, \ldots, A_n (-10^4 \leq A_i \leq 10^4)$, os elementos do vetor.

Saída

Imprima um único inteiro que é a maior soma possível de uma subsequência de rotas.

Exemplo de Entrada 1	**Exemplo de Saída 1**
6 3 -1 4 -1 5 -9	12

Exemplo de Entrada 2	**Exemplo de Saída 2**
12 -5 2 1 7 -3 -4 -8 10 -6 1 -1 3	20

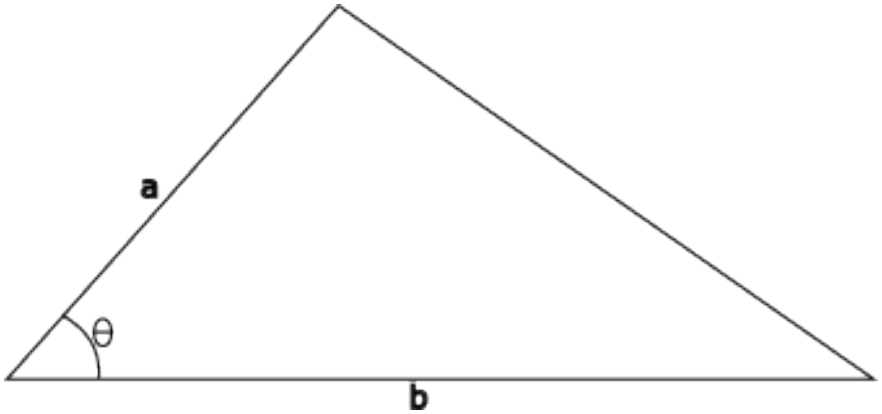
Problema E

Triangulo

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

Durante uma expedição científica em um exoplaneta recém-cartografado, uma equipe de sondas automatizadas está triangulando posições com base em sinais de rádio. Cada sonda forma triângulos com duas antenas fixas, e os dados recebidos contêm apenas dois lados do triângulo e o ângulo entre eles, medido em graus.

Por conta das severas restrições de processamento da estação remota, os cálculos devem ser feitos com **ponto flutuante de dupla precisão**, e os resultados devem ser reportados com **exatamente quatro casas decimais**.



Sua missão é escrever um programa que, a partir dos dois lados a e b, e do ângulo θ entre eles (em graus), calcule a **área** do triângulo formado. Utilize $\pi \approx 3.14159265358979323846$.

Entrada

A entrada consiste em várias linhas. Cada linha contém três números reais com até duas casas decimais:

- a — o comprimento do primeiro lado ($1 \leq a \leq 10^4$),
- b — o comprimento do segundo lado ($1 \leq b \leq 10^4$),
- θ — o ângulo entre os dois lados, em graus reais ($0 < \theta < 180$).

A entrada termina com uma linha contendo `0 0 0`, que **não deve ser processada**.

Saída

Para cada caso de teste, imprima uma linha contendo a área do triângulo correspondente, com **precisão de 4 casas decimais**, utilizando ponto flutuante de dupla precisão.

Exemplo de Entrada 1	Exemplo de Saída 1
3.00 4.00 90.00 5.00 7.00 60.00 10.00 10.00 30.00 0 0 0	6.0000 15.1554 25.0000

Problema F

Suprimentos

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

O conflito entre os **Sneakys** e a gloriosa **República da Esbórnia** se intensifica. Para apoiar os agentes infiltrados atrás das linhas inimigas, o espião Joãzinho 001 foi incumbido de administrar os **suprimentos de campo** durante uma missão de sabotagem de alta periculosidade.

Durante sua jornada, Joãzinho enfrentará uma sequência de eventos, nos quais ele poderá tanto **receber** quanto **gastar suprimentos**. Cada evento pode representar:

- a chegada de um novo carregamento (`+t`), ou
- um consumo urgente de recursos para evitar ameaças (`-t`).

Como Joãzinho **não pode ficar sem suprimentos em momento algum**, sua missão é determinar a **menor quantidade de suprimentos com que deve iniciar** a operação, de forma a garantir que nunca fique com o saldo negativo.

Sua tarefa é ajudar Joãzinho a calcular esse valor.

Entrada A entrada contém:

- Um inteiro n $(1 \leq n \leq 1000)$, representando o número de eventos de suprimento;
- n linhas subsequentes, cada uma contendo um inteiro t $(-10^6 \leq t \leq 10^6, t \neq 0)$:
  - $t > 0$: representa chegada de suprimentos;
  - $t < 0$: representa consumo de suprimentos.

Saída

Imprima um único número inteiro representando a **menor quantidade de suprimentos iniciais** necessários para que Joãzinho nunca fique com saldo negativo durante os eventos da missão.

Exemplo de Entrada 1	**Exemplo de Saída 1**
3 3 -5 3	2

Exemplo de Entrada 2	**Exemplo de Saída 2**
4 2 -3 1 -1	1

Problema G

Base Inimiga

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

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

O maléfico império dos **Sneakys**, inimigos jurados da gloriosa *República da Esbórnia*, mantém um laboratório secreto de **armas de destruição em massa** escondido nas montanhas de Prostilor.

Dois guardas patrulham corredores diferentes da base, cada um com seu próprio **ciclo de ronda**. Eles não iniciam suas patrulhas ao mesmo tempo, mas eventualmente estarão **simultaneamente** no mesmo ponto de vigilância. Esse será o único instante em que o sistema de segurança estará vulnerável.

É neste momento que nosso herói, **Joãzinho**, espião 001 da Esbórnia, poderá agir. Ele precisa invadir o laboratório, neutralizar os guardas e destruir os projetos malignos.

Sua missão é descobrir em **quantos minutos a partir de agora** os dois guardas estarão novamente no mesmo ponto de vigilância.

Entrada A entrada é composta por duas linhas.

A primeira linha contém dois inteiros d_s e y_s $(0 \leq d_s < y_s \leq 50)$, onde:

- d_s é o número de minutos desde a última vez que o **primeiro guarda** esteve em seu posto;
- y_s é o número de minutos que ele leva para **retornar** ao posto.

A segunda linha contém dois inteiros d_m e y_m $(0 \leq d_m < y_m \leq 50)$, onde:

- d_m é o número de minutos desde a última vez que o **segundo guarda** esteve em seu posto;
- y_m é o número de minutos que ele leva para **retornar** ao posto.

Garante-se que os guardas não estão nos postos neste exato momento, e que haverá uma sincronização dentro dos próximos 5000 minutos.

Saída

Seu programa deve imprimir uma única linha contendo um número inteiro: o menor número de **minutos** a partir de agora em que os dois guardas estarão simultaneamente em seus respectivos postos de vigilância.

Exemplo de Entrada 1	**Exemplo de Saída 1**
3 10 1 2	7

Exemplo de Entrada 2	**Exemplo de Saída 2**
5 12 3 8	24

Problema H

FatecTok

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

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

FatecTok é uma rede social criada pelo renomado prof. Dr. Marin Alexemberg e integrada por alunos e ex-alunos de Fatecs. É uma rede muito popular entre os alunos devido ao conteúdo de grande interesse que nela é publicado, como listas resolvidas de exercícios, dicas para passar nas matérias difíceis, status de relacionamentos de alunos e alunas e, claro, postagens com soluções dos problemas das Maratonas InterFatecs.

É uma rede social onde um aluno pode convidar outro para se tornarem amigos. Quando o convite é aceito, os dois passam a ver as publicações um do outro.

Com a FatecTok bombando, o prof. Alexemberg percebeu seu potencial e está interessado em explorar melhor as conexões que estão se formando na rede - e, a partir disso, encontrar formas de gerar valor que reconheçam todo o esforço que ele dedicou ao desenvolvimento da plataforma. Porém, como ele anda na correria, entre dar conta das aulas, corrigir trabalhos, participar de mil reuniões e preparar palestras para a SPAP, sobra pouco tempo para programar.

É aí que você entra na história. Ele o convidou para desenvolver uma nova funcionalidade para o software da rede. Considere o exemplo mostrado na figura a seguir:

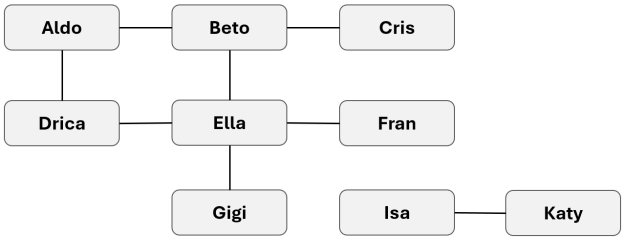


Figura H.1: Exemplo da Rede FatecTok

Perceba que há uma conexão direta entre Aldo e Beto e neste caso dizemos que a distância entre eles é 1.

Também há uma conexão entre Aldo e Cris, mas não é direta, passa pela conexão com o Beto. Nesse caso, dizemos que a distância entre Aldo e Cris é 2. E assim por diante. Já entre Aldo e Isa não há conexão, de modo que não existe um caminho na rede que comece em Aldo e chegue até a Isa.

Neste programa que você irá escrever deve primeiro ler os dados dessas conexões, carregando-os em alguma estrutura de dados. Note que havendo um par designado por "ALDO BETO", significa que ALDO enxerga as publicações de BETO e vice-versa. Depois você deve ler dois nomes e dizer qual é a distância, ou se for o caso, dizer que não há conexão entre eles.

Entrada

A entrada é constituída de um único caso de teste e os dados de entrada são divididos em duas partes.

A primeira parte começa com um número inteiro QC $(0 < QC <= 20000)$ na primeira linha. Em seguida há QC linhas contendo pares de nomes indicando as conexões existentes. Cada nome tem até 15 caracteres. Os nomes são separados por um espaço em branco e garante-se que não há espaços nos nomes dos alunos. Em seguida há uma linha separadora que contém um hífen e que deve ser desprezado.

Na segunda parte estão os pares de nomes para os quais se quer saber a distância da conexão, ou se não há conexão. São várias linhas com dois nomes separados por um espaço em branco. Essa parte termina quando forem encontrados dois asteriscos separados por espaço em branco “* *” (sem as aspas).

Saída

Para cada linha da segunda parte da entrada, a saída deve exibir o nome do primeiro aluno, seguido por um hífen, o nome do segundo aluno, o sinal de igual com espaços em ambos os lados, e, por fim, a distância entre eles ou, caso não exista uma conexão, a mensagem "sem conexao"(sem as aspas).

Exemplo de Entrada 1	**Exemplo de Saída 1**
8 Aldo Beto Aldo Drica Beto Cris Beto Ella Drica Ella Ella Fran Ella Gigi Isa Katy - Aldo Beto Aldo Cris Aldo Drica Aldo Ella Aldo Fran Aldo Gigi Aldo Isa Aldo Katy Beto Aldo Beto Gigi Cris Aldo Cris Drica Cris Gigi Cris Isa Drica Fran Drica Katy Fran Aldo Fran Beto Gigi Aldo Isa Katy Katy Isa Isa Aldo Katy Gigi * *	Aldo-Beto = 1 Aldo-Cris = 2 Aldo-Drica = 1 Aldo-Ella = 2 Aldo-Fran = 3 Aldo-Gigi = 3 Aldo-Isa = sem conexao Aldo-Katy = sem conexao Beto-Aldo = 1 Beto-Gigi = 2 Cris-Aldo = 2 Cris-Drica = 3 Cris-Gigi = 3 Cris-Isa = sem conexao Drica-Fran = 2 Drica-Katy = sem conexao Fran-Aldo = 3 Fran-Beto = 2 Gigi-Aldo = 3 Isa-Katy = 1 Katy-Isa = 1 Isa-Aldo = sem conexao Katy-Gigi = sem conexao

Problema I

A Nonna vai?

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

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

Dorothy e Dagmar são primas da mesma idade, moram na mesma rua e têm a Nonna Bina como avó. Elas costumam sair juntas nos fins de semana e usam bolinhas de bingo numeradas de 1 a 30 para decidir o que fazer. Toda quinta-feira, após o jantar na casa da Nonna, cada uma sorteia uma bolinha; quem tirar o maior número decide o programa do fim de semana.

A Nonna só permite o uso das bolinhas se, às vezes, for incluída nos passeios. Quando a soma dos dois números sorteados for maior que 40, ela vai junto, e o programa deve ser mais tranquilo (como cinema, teatro ou pizzaria). Caso contrário, as primas podem ir a baladas e festivais, que são suas preferências.

Como agora precisam se planejar com antecedência para comprar ingressos, querem fazer os sorteios para várias semanas e registrar quem decide e se a Nonna irá. Para isso, você foi encarregado de escrever um programa que organize essas informações.

Entrada

A primeira linha da entrada contém a quantidade Q ($2 \leq Q \leq 1000$) de pares de bolinhas sorteadas. Em seguida, há $2Q$ linhas, cada uma com um número inteiro entre 1 e 30. Estas linhas devem ser lidas aos pares: o primeiro valor é o número sorteado por Dorothy, e o segundo é o de Dagmar. Como elas sorteiam bolinhas físicas e não as devolvem ao globo, os valores sorteados por ambas nunca serão iguais.

Saída

Para cada par de valores o programa deve imprimir na saída quem decide e se a Nonna vai.

Se o valor da Dorothy for maior que o da Dagmar e a soma dos valores for maior que 40 o programa deve escrever "DOROTHY DECIDE E A NONNA VAI". Caso a soma não ultrapasse 40 escreva apenas "DOROTHY DECIDE".

Se o valor da Dorothy for menor que o da Dagmar e a soma dos valores for maior que 40 o programa deve escrever "DAGMAR DECIDE E A NONNA VAI". Caso a soma não ultrapasse 40 escreva apenas "DAGMAR DECIDE".

As aspas não devem ser incluídas na saída. A saída é toda em letras maiúsculas. Ao final da última linha não se equeçam da quebra de linha.

Exemplo de Entrada 1	**Exemplo de Saída 1**
4 23 12 26 20 12 23 20 26	DOROTHY DECIDE DOROTHY DECIDE E A NONNA VAI DAGMAR DECIDE DAGMAR DECIDE E A NONNA VAI

Problema J

Camisetas Tecnológicas

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

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

A famosa marca Informer é famosa pelas suas camisetas tecnológicas que são feitas de um material agradável em que não há necessidade de passar e não tem cheiro. (não ganho nada com isto!) O CEO de empresa deseja contratar um estagiário da Fatec para saber quantas camisetas no total ele poderá entregar de cada tipo e com isto o lucro que ele obterá. Atualmente a marca produz 3 tipos de camisetas – o modelo Alfa, Beta e Gama. Cada modelo de camiseta exige uma quantidade mínima de tecido para ser produzida e o lucro que ela obtém! O estagiário deve calcular dado a quantidade de tecido existente quantas camisetas de cada modelo poderão se produzidas para obter o lucro máximo!

Entrada A primeira linha da entrada contém 1 inteiro: T (indicando o comprimento de tecido disponível para fazer as camisetas) $(1 \leq T \leq 1000, T \geq Q)$, . Seguem então 3 linhas (que são os tipos de camisetas), cada uma com dois inteiros: Q $(1 \leq Q \leq T)$, (indicando a quantidade de tecido necessária para fazer cada tipo de camiseta) e L $(1 \leq L \leq 10000)$, (o lucro gerado ao vender a camiseta respectiva).

Saída

O Lucro máximo possível que pode ser conseguido com o máximo de camisetas produzidas.

Exemplo de Entrada 1	**Exemplo de Saída 1**
100 5 10 17 100 9 1000	11000

Exemplo de Entrada 2	**Exemplo de Saída 2**
10 4 4 5 5 6 6	10

Exemplo de Entrada 3	**Exemplo de Saída 3**
40 4 10 3 7 2 1	100

Problema K

Fonte

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

Em uma aldeia montanhosa, três comunidades vivem em diferentes níveis de altitude: o **Vale (nível 1)**, a **Colina (nível 2)** e o **Topo da Montanha (nível 3)**. Todos os dias, os moradores dessas comunidades transitam até uma fonte de água potável para encher seus recipientes e retornam para suas casas.

A prefeitura decidiu construir uma nova fonte em apenas um desses três níveis, mas o deslocamento entre níveis não é uniforme: as trilhas variam em dificuldade.

A prefeitura sabe que cada morador faz uma viagem de ida e volta por dia, e quer decidir o nível onde instalar a fonte para minimizar o tempo total gasto pelos moradores.

Entrada A entrada consiste em 4 linhas, contendo:

- Um número inteiro T $(0 \leq T \leq 100)$, indicando o tempo gasto para se deslocar entre dois níveis adjacentes.
- Um número inteiro V $(0 \leq V \leq 1000)$, número de moradores que vivem no Vale (nível 1).
- Um número inteiro C $(0 \leq C \leq 1000)$, número de moradores que vivem na Colina (nível 2).
- Um número inteiro M $(0 \leq M \leq 1000)$, número de moradores que vivem no Topo da Montanha (nível 3).

Saída

Seu programa deve imprimir uma única linha contendo um número inteiro: o menor tempo total possível (em minutos) gasto por todos os moradores para pegar água na nova fonte.

Exemplo de Entrada 1	**Exemplo de Saída 1**
3 10 20 30	240

Exemplo de Entrada 2	**Exemplo de Saída 2**
1 10 30 20	60