

13a. EDIÇÃO



MARATONA DE PROGRAMAÇÃO

InterFatecs

Final

28 de setembro de 2024

CADERNO DE QUESTÕES

ORGANIZAÇÃO E APOIO

Fatec
Bragança Paulista
Jornalista Omar Fagundes
de Oliveira

CPS
Centro
Paula Souza

SÃO PAULO
GOVERNO
DO ESTADO
SÃO PAULO SÃO TODOS

WWW.INTERFATECS.COM.BR

1 Instruções

Este caderno contém 13 problemas – identificados por letras de A até M, com páginas numeradas de 3 até 33. 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 (um) computador dotado do ambiente de submissão de programas (BOCA), dos compiladores, link-editores e IDEs requeridos pelas linguagens de programação permitidas;
- (g) NÃO é permitido o uso de notebooks, smartphones, ou outro tipo de computador ou assistente pessoal;
- (h) Todos os problemas têm o mesmo valor na correção.

2. Sobre o arquivo de solução e submissão:

- (a) O arquivo de solução (o programa fonte) DEVE TER o mesmo nome que o especificado no enunciado (logo após o título do problema);
- (b) confirme se você escolheu a linguagem correta e está com o nome de arquivo correto antes de submeter a sua solução;
- (c) NÃO insira acentos ou outros caracteres especiais no arquivo-fonte.

3. Sobre a entrada

- (a) A entrada de seu programa deve ser lida da entrada padrão (não use interface gráfica);
- (b) Seu programa será testado em vários casos de teste válidos além daqueles apresentados nos exemplos. Considere que seu programa será executado uma vez para cada caso de teste.

4. Sobre a saída

- (a) A saída do seu programa deve ser escrita na saída padrão;
- (b) Não exiba qualquer outra mensagem além do especificado no enunciado.

5. Versões das linguagens

- (a) gcc version 11.3.0 (lembre-se que não existem bibliotecas do Windows)
- (b) Python 3.10.6
- (c) Java 17.0.7+7-Ubuntu-0ubuntu122.04.2 (lembre-se que as classes devem estar fora de pacotes)

Problema A

Region

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

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

Donald Harris is an american entrepreneur beginning business in Brazil. He is famous due to his chain of stores that provides in-person service at several addresses and makes home deliveries regardless of the purchase value.

The only restriction on his delivery system is that the address must be within a region close to one of the physical stores in his network. Upon his arrival in Brazil, he learned about the CEP code system used by the brazilian company Correios and determined that a survey be carried out to determine which CEP codes would be considered for each store.

Now that work is done and he needs a computer program that can determine whether a customer's address is within a service region or not, and you, lucky guy, have been hired to develop it.

Input

The input consists of a test case, the first line of which contains an integer QR ($0 < QR \leq 100$) that defines the number of regions served by a store. Next, there are QR lines containing two valid CEP codes according to the Brazilian standard. This section is not supposed to be ordered in any way.

The next line contains an integer Q ($0 < Q \leq 1000$) that represents the number of delivery requests, followed by Q lines containing the customer's CEP code, according to the Brazilian standard. This section is also not ordered.

It is ensured that all the CEP codes in the input are in a valid format.

Output

The program should provide information for each customer's CEP code about whether or not they are served by the delivery system.

First, the output should contain, in ascending order, the list of customer CEP codes served by the delivery system in a format consisting of the CEP code number followed by the phrase *is served by our delivery system* (in lowercase letters).

Then, the unserved CEP codes should be listed, in ascending order, in a similar format with the CEP code followed by the phrase *is not served by our delivery system* (in lowercase letters).

Don't forget the newline character at the end of each line.

Exemplo de Entrada 1

```
1
15510-000 15512-000
5
15509-000
15510-000
15511-000
15512-000
15513-000
```

Exemplo de Saída 1

```
15510-000 is served by our delivery system
15511-000 is served by our delivery system
15512-000 is served by our delivery system
15509-000 is not served by our delivery system
15513-000 is not served by our delivery system
```

Exemplo de Entrada 2

```
3
07503-000 07550-900
07840-500 07850-000
08200-020 08240-500
10
07520-010
07395-550
08236-200
07835-060
07921-370
12410-030
07526-490
08209-000
08541-210
07550-900
```

Exemplo de Saída 2

```
07520-010 is served by our delivery system
07526-490 is served by our delivery system
07550-900 is served by our delivery system
08209-000 is served by our delivery system
08236-200 is served by our delivery system
07395-550 is not served by our delivery system
07835-060 is not served by our delivery system
07921-370 is not served by our delivery system
08541-210 is not served by our delivery system
12410-030 is not served by our delivery system
```

Problema B

Bateria Anti-aérea

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

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

Joãozinho, o conselheiro científico e militar dos Esbornianos, foi convocado para mais uma missão de alta importância. O exército dos Sneakys está utilizando um perigoso canhão de prótons, posicionado em um ponto estratégico no campo de batalha, que pode destruir completamente a **unidade de controle de bateria anti-aérea móvel**, uma estrutura circular crucial para a defesa do espaço aéreo dos Esbornianos.

Se o canhão dos Sneakys estiver mirando exatamente no centro da unidade de controle, é **impossível escapar** da destruição. No entanto, graças à ajuda de Munarinho, amigo de Joãozinho, foi desenvolvido um sistema de scramble que desorienta o radar e a mira do canhão dos Sneakys. Com isso, a unidade de controle pode se mover para escapar do disparo, desde que o canhão não mire exatamente no centro da unidade.

O objetivo é, dado a posição do canhão, o ângulo de disparo e a posição da unidade de defesa, determinar:

1. Se a unidade será **atingida fatalmente** (quando o canhão estiver mirando exatamente no centro).
2. Se a unidade **não será atingida**.
3. Quando possível escapar, calcular o **menor deslocamento necessário** para que a unidade de controle evite o disparo.

Sua tarefa é ajudar Joãozinho a garantir a segurança da unidade de controle, verificando se ela será atingida ou não e calculando o deslocamento mínimo, se possível.

Entrada

A entrada contém os seguintes valores:

- x_k, y_k : Coordenadas do canhão.
- θ : Ângulo do canhão em relação ao eixo X, dado em graus.
- x_c, y_c : Coordenadas do centro da unidade de controle circular.
- r : Raio da unidade de controle circular.

Saída

Imprima:

- **"Atingido Fatalmente"** se o canhão estiver mirando exatamente no centro da unidade de controle.
- **"Nao Atingido"** se a unidade de controle não estiver na trajetória do projétil.
- **"Atingido"** se a unidade estiver na trajetória do disparo, e também imprima a menor distância que o centro da unidade deve se mover para não ser atingido.

Restrições

- $0 \leq x_k, y_k, x_c, y_c \leq 10000$
- $1 \leq r \leq 100$
- $0 \leq \theta < 360$
- Tolerância para comparação de ponto flutuante : 10^{-6}

Exemplo de Entrada 1

590.00 489.00 70.64 590.00 489.00 98.64

Exemplo de Saída 1

Atingido Fatalmente

Exemplo de Entrada 2

981.00 285.00 238.54 1015.82 281.81 37.13

Exemplo de Saída 2

Atingido
5.76

Exemplo de Entrada 3

131.00 206.00 162.88 763.00 484.00 7.74

Exemplo de Saída 3

Não Atingido

Problema C

Coroinha Munarinho

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

Autor: Prof. Dr. Leandro Luque (Fatec Mogi das Cruzes)

Munarinho decidiu ser coroinha e para comemorar a decisão tentará reproduzir uma tentativa feita no passado por um padre que fisicamente mantinha com ele profunda semelhança física. Ele quebrou o porquinho, investiu todas as suas economias em balões e cilindros de gás hélio e está se preparando para voar preso apenas a balões, partindo de Sorocaba e indo até Araçoiaba da Serra, onde pousará na casa de um colega para uma festa de comida de caminhão (truck food). Procurando evitar acidentes, Munarinho estudou bem o uso do GPS e o clima da região. No entanto, ele entende que voar de balão é arriscado, pois não há controle da direção para a qual se vai. Desta forma, ele está escrevendo um programa de computador para prever possíveis locais de pouso dado o padrão de vento da região.

Neste programa, a região pode ser vista como uma matriz bidimensional onde cada célula é um possível local de pouso. Munarinho sairá de uma célula e, dados golpes de vento, será levado a outras células. Para simplificar, Munarinho está assumindo que um golpe de vento o leva de uma célula para outra imediatamente acima, abaixo, à esquerda ou à direita.

Ajude Munarinho a escrever este programa. Dado o tamanho da matriz da região, a célula onde terá a comida de caminhão e o número/direção de golpes de vento, determine a célula da qual Munarinho teria que partir para chegar inteiro à festa.

Entrada

A entrada é composta por um caso de teste. A primeira linha contém cinco números inteiros separados por um espaço em branco L C N CL CC representando, respectivamente, o número de linhas ($1 \leq L \leq 100$) e de colunas ($1 \leq C \leq 100$) da matriz da região, o número N de golpes de vento ($1 \leq N \leq 1000$), bem como a linha ($1 \leq CL \leq L$) e coluna ($1 \leq CC \leq C$) do local onde será a festa de comida de caminhão. A segunda linha contém N caracteres D ('C', 'D', 'B', 'E'), representativos da direção do golpe de vento (Cima, Direita, Baixo, Esquerda).

Saída

A saída deve conter dois números inteiros X Y indicativos da linha e coluna da célula de onde Munarinho deverá partir para chegar à festa de comida de caminhão. Caso esta célula não exista na matriz, imprima -1 -1.

Exemplo de Entrada 1

5 5 8 3 3
CDBEEDDC

Exemplo de Saída 1

4 2

Exemplo de Entrada 2

10 10 5 0 0
DDDDD

Exemplo de Saída 2

-1 -1

Esta página foi propositadamente deixada em branco.

Problema D

Triângulo

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

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

Dado um triângulo com três lados a , b e c , sua tarefa é calcular a área desse triângulo. A área A de um triângulo cujos lados possuem os comprimentos a , b e c pode ser calculada com base nos lados fornecidos, e você deve imprimir o valor da área com **duas casas decimais** de precisão.

Entrada

A entrada contém três números reais a , b e c representando os lados do triângulo, onde:

- a , b , e c são valores em ponto flutuante de dupla precisão.

Saída

Imprima um número real representando a área do triângulo, com **duas casas decimais**.

Restrições

- $0 < a, b, c \leq 10^6$
- Os valores fornecidos sempre formarão um triângulo válido.
- Utilize pontos flutuantes de precisão dupla.

Exemplo de Entrada 1

8.00 6.00 10.00

Exemplo de Saída 1

24.00

Exemplo de Entrada 2

3.00 4.00 5.00

Exemplo de Saída 2

6.00

Esta página foi propositadamente deixada em branco.

Problema E

ACME

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

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

As Indústrias ACME estão se modernizando e para isso fizeram um investimento considerável em novas máquinas totalmente automatizadas.

As novas máquinas, dispondo de um conjunto completo de ferramentas e considerando que tenham seus silos alimentados com as matérias primas necessárias, podem funcionar ininterruptamente e produzir sequencialmente vários lotes de peças. Para isso elas tem a capacidade de receber uma programação vinda de um servidor central. Essa programação se caracteriza por uma fila de lotes de peças a serem produzidas. Cada lote contém a indicação do tipo de peça e a quantidade a ser produzida, mas essa informação não é importante para nós. As únicas informações importantes no momento são: o **momento** em que o lote chegou à fila e quanto **tempo** demora para o lote ser produzido. Esses tempos e momentos são sempre contados em minutos.

A cada 800 horas (48.000 minutos) de funcionamento da máquina é feita uma parada para manutenção preventiva, com lubrificação, substituição das ferramentas desgastadas e quaisquer outras medidas previstas no plano de manutenção. Uma máquina raramente quebra, mas se quebrar isso é tratado à parte e também não é importante para nós. O tempo em que a máquina está trabalhando entre duas paradas de manutenção é chamado de **jornada**.

A parada de manutenção sempre é feita ao término de algum lote de produção e se existirem lotes na fila de esses lotes ficam no aguardo da reativação pós-manutenção. Neste caso, esses lotes presentes na fila antes da parada recebem o valor 0 (zero) para o momento de chegada à fila.

Por política da empresa, o planejamento de uma jornada é feito de forma a organizar a fila dos lotes visando minimizar o tempo médio com que cada lote é completado. Para minimizar esse tempo médio usa-se um método de gestão em que a fila de produção é ordenada de forma crescente pelo tempo de produção. Assim, após uma parada, o lote com menor tempo sempre será executado primeiro, depois o lote de segundo menor tempo e sucessivamente.

Todo lote tem um número de identificação. Essa numeração é sequencial e sempre que ocorrer um empate no tempo de produção de dois ou mais lotes, os lotes de menor número de identificação devem ser produzidos antes.

Nessas máquinas a produção é contínua. Isso significa que, assim que a última peça de um lote seja finalizada, imediatamente a primeira peça do próximo lote é iniciada.

Quanto à operação, suponha a existência dos lotes A e B com tempo de 50 e 70 minutos, respectivamente. Como o lote A requer 50 minutos e iniciou no tempo 0, então ele ficará na máquina entre os tempos 0 e 50; já o lote B terá início no momento 50 e ficará na máquina até o momento 120; o próximo lote após B iniciará em 120, e assim por diante.

A figura a seguir ilustra uma situação típica com três lotes na fila durante a parada: 210, 211 e 212, cada um com seu tempo de execução e o momento de chegada configurado para zero. Na retomada dos trabalhos, a fila será rearranjada em ordem crescente de tempo de execução e os trabalhos recomeçam.

Uma vez que a fábrica encontra-se em pleno funcionamento, a entrada de novos lotes na fila de produção pode ocorrer a qualquer momento e verificam-se duas situações:

Figura E.1: Fila em momento de parada

Antes da parada			Início da nova jornada		
Chegada	Lote	Tempo	Chegada	Lote	Tempo
0	210	2000	0	211	450
0	211	450	0	212	950
0	212	950	0	210	2000

1. O novo lote chega quando a máquina está parada. Neste caso o momento da chegada é registrado como 0 (zero);
2. O novo lote chega durante uma jornada. Neste caso o momento da chegada é registrado com um tempo em minutos contados a partir do momento em que a máquina iniciou a jornada. A partir disso, duas situações podem ocorrer:
 - (a) O novo lote chega enquanto a máquina está produzindo um lote;
 - (b) O novo lote chega enquanto a máquina está em estado de aguardo, pois os lotes anteriores já terminaram.

Importante ressaltar que a política da empresa quanto à ordem da fila deve sempre ser respeitada, ou seja, na ocorrência do caso 1. acima a fila deve ser ordenada segundo tempos crescentes na retomada dos trabalhos. Na ocorrência do caso 2., em suas duas variações o novo lote deve ser alocado na fila de lotes pendentes segundo as regras de ordenamento descritas, ressaltando que se houver algum lote em produção (caso 2.a), em hipótese alguma essa produção será interrompida.

Deste modo, ampliando o exemplo acima considere a chegada de novos lotes durante a jornada teremos o seguinte resultado:

Figura E.2: Fila durante a produção

Lotes na jornada			Sequência de produção na jornada			
Chegada	Lote	Tempo	Lote	Início	Término	Observação
0	210	2000	211	0	450	Estava na fila na parada
0	211	450	214	450	650	Chegou durante a jornada
0	212	950	215	650	930	Chegou durante a jornada
300	213	1360	212	930	1880	Estava na fila na parada
350	214	200	213	1880	3240	Chegou durante a jornada
500	215	280	210	3240	5240	Estava na fila na parada

Sequência de eventos que geram esse resultado:

O lote 211, com menor tempo, foi o primeiro a ser iniciado na jornada.

Enquanto o lote 211 estava em produção chegaram os lotes 213 e 214. O 213 chegou antes, mas por ter um tempo longo ficou em penúltimo na fila, à frente apenas do lote 210. O lote 214, por ter o tempo mais curto dentre os pendentes, ficou no topo da fila.

O lote 215 chegou quando estava em produção o lote 214. Por ter um tempo mais curto que os demais pendentes assumiu a primeira posição na fila. Ao final da jornada e antes da parada de manutenção o lote que estiver em execução será terminado.

Sua tarefa é escrever um programa que fará a gestão da fila da máquina, levando em conta tudo o que foi apresentado acima.

Entrada

A entrada é constituída por um caso de teste contendo os lotes a serem produzidos e seus momentos e tempos. Cada linha contém três números inteiros, sendo: MT ($0 \leq MT < 48000$) o momento em que o lote chega à fila; $LOTE$ ($LOTE > 0$) é o número do lote; DT ($0 < DT \leq 10000$) é a duração da execução do lote. A entrada termina quando forem encontrados três zeros (0 0 0).

Saída

O programa deve imprimir em cada linha da saída três inteiros: o número do lote, o momento de início de sua produção e o momento de término de sua produção. Todos os lotes presentes na entrada deverão estar presentes na saída, na ordem exata de produção gerada a partir da aplicação das regras descritas no enunciado. Na última linha imprima o final de linha.

Exemplo de Entrada 1

```
0 210 2000
0 211 450
0 212 950
300 213 1360
350 214 200
500 215 280
0 0 0
```

Exemplo de Saída 1

```
211 0 450
214 450 650
215 650 930
212 930 1880
213 1880 3240
210 3240 5240
```

Exemplo de Entrada 2

```
0 3095 180
0 3097 90
0 3098 150
0 3100 25
0 3101 120
0 3102 40
105 3103 50
210 3104 90
320 3105 60
450 3106 90
1200 3107 1500
1210 3108 800
1350 3109 60
0 0 0
```

Exemplo de Saída 2

```
3100 0 25
3102 25 65
3097 65 155
3103 155 205
3101 205 325
3105 325 385
3104 385 475
3106 475 565
3098 565 715
3095 715 895
3107 1200 2700
3109 2700 2760
3108 2760 3560
```

Esta página foi propositadamente deixada em branco.

Problema F

Posfixa

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

Em geral quando alguém precisa escrever uma expressão aritmética utiliza uma forma de notação conhecida como notação Infixa. Você deve estar pensando: “Caramba! Começou mal esse texto, o que é essa tal de Infixa?”

Bem, não se espante, o nome pode não ser familiar, mas você conhece essa notação. Ao escrever as expressões aritméticas como essas:

$$A + B * 2$$

$$(A + B) * 2$$

$$(A - B)/(C - D)$$

you está usando a notação Infixa. Essa forma de escrita tem esse nome porque os operadores são posicionados no meio da expressão, entre os operandos. Agora, acho que vem outra questão no seu pensamento: “Como assim, essa forma de notação? Por acaso, existe outra?”

Pois é: a resposta é SIM!! Existem as notações Prefixas e Posfixas. E aqui vamos nos concentrar nesta segunda, Posfixa, também conhecida como Notação Polonesa Reversa em homenagem ao matemático polonês Jan Łukasiewicz que a desenvolveu no ano de 1924.

Essa proposta é muito interessante pois elimina a necessidade de uso dos parênteses na expressão. Por exemplo, as expressões acima podem ser reescritas em notação Posfixa como mostrado a seguir

Infixa	Posfixa	Interpretação da Posfixa
$A + B * 2$	AB2*+	Tome A e guarde, tome B e guarde, tome 2 e guarde, multiplique os dois anteriores ($B*2$) e guarde, some os dois anteriores ($A+B*2$). Esse é o resultado.
$(A + B) * 2$	AB+2*	Tome A e guarde, tome B e guarde, some os dois anteriores ($A+B$) e guarde, tome o 2 e guarde, multiplique os dois anteriores. Esse é o resultado.
$(A-B)/(C-D)$	AB-CD-/	Tome A e guarde, tome B e guarde, subtraia os dois anteriores e guarde, tome C e guarde, tome D e guarde, subtraia os dois anteriores e guarde, divida os dois anteriores (que são os resultados A-B e C-D)

Na interpretação acima, A, B, C, D e algarismos são operandos da expressão. Quando usamos o termo “guarde” pense em uma pilha de itens guardados. Cada operando (variável ou algarismo) que você guarda é colocado nessa pilha. Quando você atinge um operador a operação deve ser feita com os dois itens que estiverem no topo da pilha, que são removidos dela, e o resultado obtido fica no topo dessa pilha. Quando a expressão chega ao final, na pilha só existirá um valor, que é o resultado final.

As operações disponíveis são cinco e as prioridades matemáticas de sua execução devem ser respeitadas, ou seja, as operações de maior prioridade são feitas antes.

Saiba que esse sistema Posfixo tem sido muito utilizado em calculadoras eletrônicas e na computação. Os compiladores quando analisam uma expressão aritmética Infixa eles a transformam em Posfixa para efeitos

Figura F.1: Símbolos usados e ordem de prioridade

Operação	Símbolo	Maior prioridade	Potenciação
Adição	+	Média prioridade	Multiplicação e Divisão
Subtração	-	Baixa prioridade	Adição e Subtração
Multiplicação	*		
Divisão	/		
Potenciação	^		

de realização dos cálculos. Como curiosidade saiba que a empresa Hewlett-Packard desenvolveu toda uma geração de calculadoras eletrônicas que utilizavam essa forma de expressão aritmética.

Muito bem, agora que você já conhece o que são as notações Infixa e Posfixa e tem uma noção de que a notação Posfixa é muito importante, chegou a hora de trabalhar.

Escreva um programa capaz de converter expressões aritméticas válidas em Notação Infixa e as converta para Notação Posfixa.

Entrada

A entrada tem vários casos de teste. Cada caso de teste é um string contendo variáveis, valores literais, operadores aritméticos e parênteses. Nomes de variáveis contém uma única letra maiúscula (de A a Z); valores literais contém um único algarismo (de 1 a 9); estes símbolos (+, -, *, /, ^) são os operadores aritméticos. Esse string não tem qualquer outro caractere além desses, nem mesmo espaços em branco.

Garante-se que todas as expressões contidas na entrada são válidas em formato Infixo e aceitam a devida conversão para Posfixo.

O arquivo de entrada termina com uma linha contendo um único caractere que é o ponto final "." (sem as aspas).

Saída

Para cada string da entrada o programa deve imprimir a correspondente expressão Posfixa seguida de um pulo de linha.

Exemplo de Entrada 1

```
A+B
A/2
A+B*C
(A+B)*C
(A+B)/(C-D)
X^Y/Z
X^(Y/Z)
2*A+B
X+Y^2
(X+Y)^2
.
```

Exemplo de Saída 1

```
AB+
A2/
ABC*+
AB+C*
AB+CD-/
XY^Z/
XYZ/^
2A*B+
XY2^+
XY+2^
```


Problema G

Estrados

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

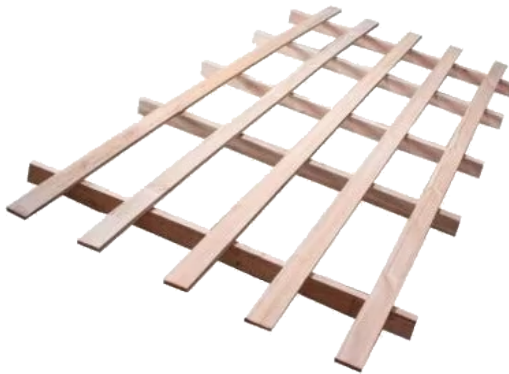
Autor: Prof. Me. Sérgio Luiz Banin (Fatec São Paulo)

Meu jovem, você acaba de receber grande oportunidade na sua vida. O sr. Temístocles Leopoldo, velho amigo do seu pai, passou a administrar a empresa Bons Sonhos fabricante de estrados de madeira de alta qualidade para camas luxuosas.

O estrado é a estrutura que sustenta o colchão e é essencial para garantir uma base estável e uniforme para alguém dormir. Eles podem ser de madeira, metal ou plástico. A Bons Sonhos fabrica estrados de madeira, montados com ripas longitudinais pregadas em caibros transversais grossos e resistentes, como mostrado na figura.

O estrado sempre usa 5 caibros transversais e o que varia são 3 parâmetros: a largura da madeira da ripa; a quantidade de ripas; o espaçamento entre as ripas.

Figura G.1: Um estrado é assim



Há estrados de várias larguras: criança, solteiro, casal, king size, etc.

Embora a empresa seja lucrativa, o sr. Temístocles tem detectado diversas falhas administrativas, que se forem sanadas podem ajudar a empresa a lucrar mais ainda.

Uma das falhas mais severas está na área de produção onde ocorrem dois tipos de problemas que levam ao retrabalho dos estrados produzidos, diminuindo a eficiência e a lucratividade.

Alguns estrados são especificados com excesso de ripas, com gasto excessivo de material. Outros estrados são especificados com ripas a menos, fazendo com que o espaço entre elas seja grande demais, o que causa desconforto ao dorminhoco, insatisfação e pedidos de reembolso. Estudos técnicos mostram que o espaçamento entre duas ripas vizinhas deve estar entre 10 cm e 20 cm.

Se o espaçamento for menor que 10 cm há excesso de ripas e o projeto está superdimensionado. Se esse espaçamento for maior que 20 cm há poucas ripas e o projeto está subdimensionado.

Caro programador, dito isto, chega sua vez de agir. Por conhecê-lo desde pequeno e saber de seu potencial, o sr. Temístocles, o contrata para escrever um programa que verifique todas as especificações de projeto antes de enviá-los para a produção.

Seu trabalho é escrever um programa que leia a largura total do estrado, a quantidade de ripas e a largura da ripa. Em seguida calcule o espaçamento entre ripas, de modo que a largura total não seja ultrapassada e imprima uma de três possíveis situações: projeto ok, projeto superdimensionado ou projeto subdimensionado.

Entrada

A primeira linha da entrada contém um número inteiro Q ($0 < Q \leq 1000$) que é a quantidade de projetos a serem analisados. Em seguida estão Q linhas de dados de estrados contendo 3 números inteiros separados por um espaço em branco, a saber:

- Largura total do estrado LT ($60 \leq LT \leq 210$)
- Quantidade de ripas QR ($4 \leq QR \leq 15$)
- Largura de uma ripa LR ($3 \leq LR \leq 8$)

As medidas de largura são expressas em centímetros.

Saída

Para cada linha de dados de estrados o programa deve imprimir o seguinte:

- Se o espaçamento entre ripas for menor que 10 cm imprima **projeto superdimensionado**
- Se o espaçamento entre ripas for maior que 20 cm imprima **projeto subdimensionado**
- Se o espaçamento estiver entre 10 e 20 cm, incluindo estes valores imprima **projeto ok**

Exemplo de Entrada 1

```
12
88 8 8
88 8 5
88 6 4
88 6 3
96 4 3
96 7 3
96 10 4
120 8 3
120 10 4
200 8 3
200 12 4
200 14 5
```

Exemplo de Saída 1

```
projeto superdimensionado
projeto superdimensionado
projeto ok
projeto ok
projeto subdimensionado
projeto ok
projeto superdimensionado
projeto ok
projeto superdimensionado
projeto subdimensionado
projeto ok
projeto ok
```

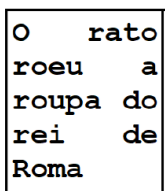
Problema H

Justificando apenas ...

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

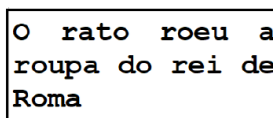
Autor: Prof. Antonio Cesar de Barros Munari (Fatec Sorocaba)

Armando está fazendo a manutenção de uma funcionalidade para um sistema que ele mantém já há algum tempo. Trata-se de um aperfeiçoamento em uma rotina de exibição de textos descritivos contidos em alguns cadastros, com o objetivo de permitir que o conteúdo apareça justificado, ou seja, fazendo o alinhamento das linhas tanto no seu início como no seu final. Em um texto com alinhamento justificado toda linha se inicia e se encerra com um conteúdo útil (uma letra, número ou caracter especial que não seja nem o espaço em branco nem alguma tabulação), alinhados tanto à esquerda como à direita. A última linha de um texto justificado é sempre alinhada apenas à esquerda. Para que isso seja possível é necessário cuidar do espaçamento, e há várias maneiras de se conseguir esse efeito, algumas mais sofisticadas e outras mais simples. Armando está trabalhando em algo que seria uma solução de complexidade intermediária, que poderá ser aperfeiçoada no futuro. Para conseguir o efeito desejado ele vai imprimir o texto com caracteres de largura fixa, ou seja, todos os caracteres imprimíveis serão exibidos com a mesma largura, como ocorre com fontes monoespaciais tipo Courier New, por exemplo. Para ajustar a largura das linhas para que todas estejam ajustadas tanto à esquerda como à direita, ele vai operar sobre os espaços em branco, aumentando a largura de cada um conforme a necessidade. Ou seja, todo o conteúdo do texto terá os caracteres com a mesma largura, com exceção dos espaços em branco, que poderão aparecer com largura variável. Nesta abordagem caso uma linha precise conter apenas uma palavra, esta estará alinhada apenas à esquerda, já que não haverá espaço em branco algum que permita ajustar o comprimento da linha. As figuras 1 e 2 ilustram um mesmo texto justificado conforme duas larguras distintas para apresentação do conteúdo.



```
O  rato
roeu  a
roupa do
rei   de
Roma
```

Fig. 1 Texto com largura 10



```
O  rato roeu a
roupa do rei de
Roma
```

Fig. 2 Texto com largura 15

Enquanto Armando escreve essa parte gráfica, ele atribuiu a você, seu estagiário, a tarefa de determinar qual a largura que o espaço em branco deve possuir em cada linha de maneira que o efeito de texto justificado seja conseguido.

Entrada

A entrada contém um caso de teste composto por duas linhas. Na primeira linha encontra-se um inteiro L ($0 < L \leq 120$) que indica a largura do espaço onde o texto será exibido. Assuma que cada caracter imprimível tem largura igual a 1 com a já mencionada exceção dos espaços em branco, que podem ter largura ≥ 1 . A segunda linha da entrada contém um texto composto por até 5000 caracteres variados (letras, dígitos e caracteres especiais diversos, inclusive espaços em branco).

Saída

Para cada linha que o texto ocupará ao ser exibido, imprima qual a largura que um espaço em branco precisará ter para que ela fique justificada. Essa largura deve ser um número real de precisão dupla com 3 casas depois da vírgula, conforme os exemplos fornecidos a seguir. Um espaço em branco deve separar os valores exibidos, com uma quebra de linha final após o último valor.

Exemplo de Entrada 1

```
10
O rato roeu a roupa do rei de Roma
```

Exemplo de Saída 1

```
5.000 5.000 3.000 5.000 0.000
```

Exemplo de Entrada 2

```
12
O rato roeu a roupa do rei de Roma
```

Exemplo de Saída 2

```
1.500 2.000 1.000
```

Exemplo de Entrada 3

```
15
O rato roeu a roupa do rei de Roma
```

Exemplo de Saída 3

```
1.667 1.000 0.000
```

Problema I A Fuga

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

Luquinho, um jovem corajoso e pueril, despertou a ira dos terríveis **Zeboinos**, seres humanoides malvados que dominam a perigosa galáxia de *Koiláda Sideím*, habitantes do mundo Bizarro Admá localizada na quarta dimensão. Em uma vingança cruel, os Zeboinos o teletransportaram para a tenebrosa Admá repleta de humanoides antropófagos. O destino de Luquinho é sombrio, além de estar preso numa região inóspita e correndo risco de ser capturado e servido ao deleite dos Zeboinos de Admá espalhados pela região.



Figura I.1: Luquinho em fuga auxiliado por Baninzinho e Munarinho

Felizmente, seus amigos **Baninzinho** e **Munarinho** vieram socorrer-lo. Por sorte, Luquinho ao ser teletransportado portava um *pager topológico* inventado por Baninzinho, este pager é capaz de geo referenciar espaços topológicos, assim ele conseguiu identificar a posição de Luquinho e com a ajuda da máquina de *criação de singularidades* inventada por Munarinho (inspirada Motores de Singularidade Romulanos), criaram um **portal de saída** para que Luquinho possa escapar.

A tarefa é guiar Luquinho do ponto inicial onde ele foi teletransportado até a saída, evitando os mortais Zeboinos no caminho. Crie um programa para encontrar o caminho mais curto para que Luquinho possa escapar, se possível!

A região será representada por uma área quadrática de 100×100 .

Entrada

A entrada contém as seguintes informações:

- Dois inteiros $x_{inicial}$ e $y_{inicial}$ ($0 \leq x_{inicial}, y_{inicial} < 100$) indicando a posição inicial de Luquinho no labirinto.
- Dois inteiros x_{saida} e y_{saida} ($0 \leq x_{saida}, y_{saida} < 100$) indicando a posição da saída do labirinto, que está sempre em uma das bordas.
- Um número inteiro k ($0 \leq k \leq 3000$) representando o número antropófagos no labirinto.
- As próximas k linhas contêm duas coordenadas x_{trap} e y_{trap} ($0 \leq x_{trap}, y_{trap} < 100$) que indicam a posição dos antropófagos.

Saída

Imprima o menor número de movimentos que Luquinho precisa para escapar do labirinto. Se não for possível escapar, imprima -1 .

Exemplo de Entrada 1

```
50 50
0 99
5
10 10
20 20
30 30
40 40
60 60
```

Exemplo de Saída 1

```
99
```

Exemplo de Entrada 2

```
45 45
0 99
7
44 45
46 45
45 44
45 46
40 40
50 50
60 60
```

Exemplo de Saída 2

```
-1
```

Problema J

Uma Ligação Eletrizante

Arquivo fonte: energia.{ c | cpp | java | py }
Autor: Prof. Dr. Rodrigo Plotze (Fatec Ribeirão Preto)

Mr. Joules Watts é um recém formado Engenheiro Eletricista e está participando de um ambicioso projeto que será capaz de interligar diversas estações de energia espalhadas pelo país.

O principal problema enfrentado por *Mr. Joules* está relacionado ao impacto ambiental. Para isso, ele precisa otimizar a quantidade de cabeamento que será utilizado para conectar todas as estações e, com isso, minimizar o custo do projeto. Em outras palavras, ele precisa interligar todas as estações de energia utilizando a menor quantidade possível de cabeamento.

Na Figura 1 é possível observar o problema enfrentado por ele, em que é apresentado um conjunto de 5 (cinco) estações de energia identificadas pelos números inteiros 0, 1, 2, 3, 4, e as ligações entre elas. O peso das ligações está associado à distância, em quilômetros, entre cada estação de energia. Na Figura 2 é apresentado o resultado desejado, em que se destaca a maneira otimizada para minimizar a quantidade necessária de cabeamento para interligar todas as estações de energia. Neste exemplo, a quantidade total é de 14 quilômetros.

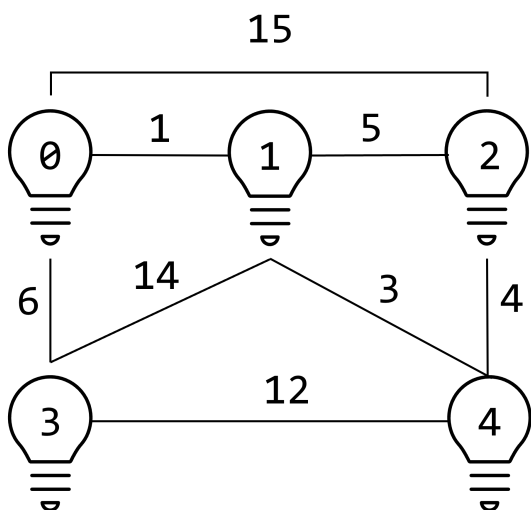


Figura 1: As estações de energia são identificadas pelos números inteiros {0,1,2,3,4} e as respectivas distâncias (em quilômetros) necessárias para interligação.

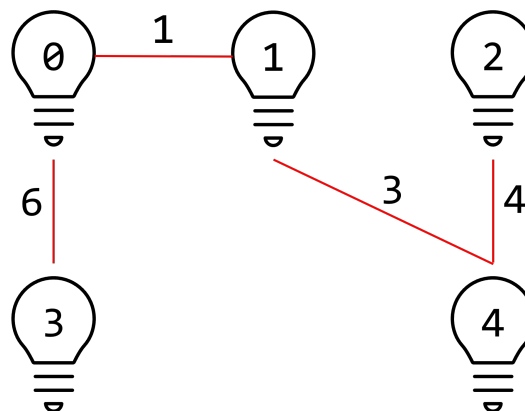


Figura 2: A solução otimizada indica que é necessário um total de 14 quilômetros de cabos para interligar todas as estações de energia.

Dentro deste contexto, elabore um algoritmo capaz de auxiliar *Mr. Joules* a resolver o problema de interligar as estações de energia.

Entrada

A primeira linha contém um número inteiro E ($3 \leq E \leq 30$) que indica a quantidade de estações de energia que precisam ser interligadas e um número inteiro L que representa a quantidade de ligações entre as estações ($3 \leq L \leq 100$). As próximas E linhas contém 3 inteiros (X, Y, Z), separados por espaço em

branco, em que, X e Y são números inteiros que indicam as estações de energia que podem ser interligadas e Z um número inteiro que representa a distância em quilômetros entre as estações.

Saída

Imprimir um número inteiro que representa o custo total mínimo, em quilômetros, para interligar todas as estações de energia.

Exemplo de Entrada 1

```
7 10
1 2 5
1 3 3
1 4 9
2 4 2
2 5 7
3 4 4
3 6 8
4 6 6
5 6 1
5 7 3
```

Exemplo de Saída 1

```
19
```

Exemplo de Entrada 2

```
5 8
1 2 4
1 3 6
1 4 7
2 3 1
2 4 5
3 4 2
3 5 8
4 5 3
```

Exemplo de Saída 2

```
10
```

Exemplo de Entrada 3

```
4 5
1 2 10
1 3 15
2 3 20
2 4 25
3 4 30
```

Exemplo de Saída 3

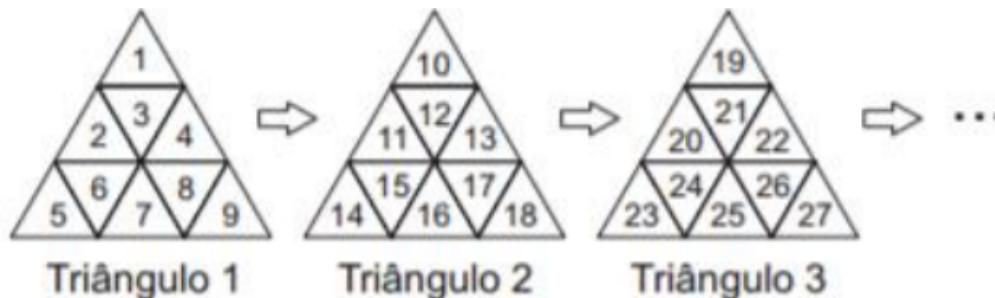
```
50
```


Problema K

Figura triângulo

Arquivo fonte: figuratriangulo.{ c | cpp | java | py }
Autor: Prof. Dr. Reinaldo Arakaki (Fatec São José dos Campos)

Manuel é uma criança muito ativa, ele começa a escrever os números naturais em figuras triangulares de acordo com o padrão da figura abaixo:



E, a seguir, escreve as letras A, B, C, D, E, F, G, H e I nestes triângulos conforme a ilustração:



E começa uma brincadeira de codificação de números, por exemplo o número 5 é codificado como 1E, pois se encontra no triângulo 1 e sua posição no triângulo de letras é E! Ajude Manuel a codificar os números!

Entrada

Um número inteiro N ($1 \leq N \leq 1.000.000$).

Saída

O número codificado segundo o padrão descrito no texto.

Exemplo de Entrada 1

5

Exemplo de Saída 1

1E

Exemplo de Entrada 2

16

Exemplo de Saída 2

2G

Exemplo de Entrada 3

27

Exemplo de Saída 3

3I

Esta página foi propositadamente deixada em branco.

Problema L

Coisa de Calouro

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

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

Todos nós sabemos como é ser calouro em uma faculdade, é um sentimento de alegria por estar em um ambiente repleto de oportunidades e, ao mesmo tempo, receio de tomar decisões erradas que podem afetar uma vida inteira. Victória sabe bem disso! Essa jovem estudante muito dedicada quer começar sua graduação com o pé direito, isto é, não quer cometer erros típicos que nós conhecemos bem.

Como Victória é esperta, resolveu começar por uma situação mais evidente e, à medida que se acostumar com o novo ambiente, se preparará para outras mais sutis. Talvez, a questão mais notável é em relação às avaliações das disciplinas. Existem diversas disciplinas por semestre, cada uma com suas exigências e graus de dificuldade, por isso, algumas demandam mais tempo dedicado a estudos extraclasse e outras menos.

Há disciplinas com avaliações como entrega de trabalhos realizados dentro e fora do horário de aula, algumas também cobram que os alunos pesquisem e apresentem algo sobre algum tema pertinente, porém o que todas têm em comum são as avaliações do tipo prova presencial, agendadas para um dia específico de acordo com o calendário escolar. É justamente esse último tipo de avaliação que mais preocupa a caloura.

Victória já percebeu que, diferentemente do modo como aconteceu durante seu ensino fundamental e médio, na graduação precisará estudar com mais constância e bastante antecedência para conquistar uma boa nota, porém ainda não sabe quanto tempo antes das provas precisará começar a estudar, a única coisa que ela tem certeza é que deixar para estudar no dia da prova é "coisa de calouro ingênuo", tanto que ela assumiu como regra pessoal não estudar no dia da aplicação das provas, apenas nos que antecedem.

A caloura elaborou uma estratégia interessante, decidiu que consultará o veterano que teve o melhor desempenho em cada disciplina no semestre anterior, perguntando quanto tempo foi investido para estudar para a respectiva disciplina antes do dia da prova. Após a conversa com o veterano, estipulará a data em que começará a estudar, porém como está ansiosa, poderá definir uma data problemática por engano, que não segue a dica do veterano ou, pior, as vezes será uma data posterior a aplicação da prova! Ela precisará ficar atenta e tomar cuidado com isso.

Pronto, agora está quase tudo resolvido, exceto que existe outro problema! Houve uma confusão na construção do calendário escolar deste ano, em algumas disciplinas estão trocadas as quantidades de dias dos meses. Por exemplo, o mês de janeiro, que deveria ter trinta e um dias, pode ter ficado com a quantidade de dias de setembro, isto é, trinta dias, e setembro com a quantidade de julho, que são trinta e um dias. Que bagunça!

Como o calendário foi distribuído aos professores, que também distribuíram cópias aos seus alunos antes do erro ser descoberto, as datas das provas foram definidas com base nesse calendário mesmo... paciência. Por conta da pressa, alguns professores até podem ter definido a prova em um dia inexistente nesse calendário potencialmente defeituoso. Pelo menos a sequência correta dos meses e a regra de que "após o último dia de um mês está o primeiro dia do próximo mês" continua intacta, ufa!

Você, alguém mais experiente na faculdade, colega de Victória e excelente em programação, decidiu ajudá-la! Construirá um programa que solicitará:

1. A data em que a prova será aplicada;

2. A dica dada pelo veterano, que é a quantidade de tempo que ele usou para estudar para a prova daquela disciplina, sendo que o veterano poderá informar esse tempo em uma das seguintes unidades de medida: dias, semanas ou meses, considerando que cada mês tem exatamente trinta dias;
3. A data em que a jovem caloura começará a estudar;
4. Os meses, em sequência arbitrária, e a respectiva quantidade de dias de cada mês, segundo o calendário distribuído pelo professor que, lembre-se, pode estar com problemas.

Após as entradas, o programa exibirá uma frase cadastrada pelo veterano que ajudou Victória, basicamente um resumo da opinião dele sobre a data em que Victória decidiu começar a estudar e, consequentemente, a quantidade de dias que ela terá para estudar antes da prova. As opiniões podem ser uma das seguintes, todas propositalmente sem acentuação e escritas em minúsculo:

- *"data nao existe!"*: quando o dia em que o professor agendou a aplicação da prova, ou em que Victória definiu como início dos estudos, não existir no calendário da disciplina;
- *"esta de brincadeira?"*: quando a quantidade de dias de estudo antes do dia da aplicação da prova for zero ou quando a data para começar a estudar for posterior à data da aplicação da prova;
- *"olha a reprovacao chegando!"*: quando a quantidade de dias de estudo antes da prova for de pelo menos um dia, porém inferior à quantidade de tempo que o veterano usou e informou como dica;
- *"que caloura ousada!"*: quando a quantidade de dias de estudo antes da aplicação da prova for idêntica àquela usada pelo veterano;
- *"jovem consciente!"*: quando a quantidade de dias de estudo antes da aplicação da prova for superior àquela usada pelo veterano.

Entrada

A entrada está descrita no próprio enunciado e deve ser considerada conforme os casos de teste de exemplo. Ainda assim, destaca-se:

- Tanto a data da aplicação da prova (1ª linha), quanto a data do início dos estudos de Victória (3ª linha), contêm apenas o dia e o mês, ambos por extenso, em minúsculo e sem acentuação. Ambas as datas estão no mesmo ano;
- A 2ª linha contém a dica do veterano, sendo sempre um número natural seguido de uma das unidades de medida mencionadas no texto, tanto no singular quanto no plural ("*dia*" ou "*dias*", "*semana*" ou "*semanas*", "*mes*" ou "*meses*"), sem acentuação e em minúsculo;
- Da 4ª até a 15ª linha estarão os meses, abreviados com apenas suas três primeiras letras, em minúsculo e em sequência arbitrária, além do sinal " : " (dois pontos, sem aspas), um espaço e a quantidade de dias do respectivo mês.

Saída

A opinião do veterano, conforme descrito no texto, em apenas uma linha, com o texto sem acentuação e em minúsculo. Lembre-se de finalizar a exibição da opinião com uma quebra de linha.

Exemplo de Entrada 1

```
quatorze de outubro  
1 dia  
treze de outubro  
jan: 31  
fev: 28  
mar: 31  
abr: 30  
mai: 31  
jun: 30  
jul: 31  
ago: 31  
set: 30  
out: 31  
nov: 30  
dez: 31
```

Exemplo de Saída 1

```
que caloura ousada!
```

Exemplo de Entrada 2

```
dezesseis de janeiro  
10 dias  
sete de janeiro  
jan: 31  
fev: 28  
mar: 31  
abr: 30  
mai: 31  
jun: 30  
jul: 31  
ago: 31  
set: 30  
out: 31  
nov: 30  
dez: 31
```

Exemplo de Saída 2

```
olha a reprovacao chegando!
```

Exemplo de Entrada 3

```
dezessete de dezembro  
1 semana  
tres de dezembro  
ago: 31  
set: 30  
out: 31  
mar: 31  
abr: 30  
mai: 31  
jun: 30  
nov: 30  
jan: 31  
fev: 28  
jul: 31  
dez: 31
```

Exemplo de Saída 3

```
jovem consciente!
```

Exemplo de Entrada 4

```
trinta e um de fevereiro  
3 semanas  
dez de fevereiro  
ago: 28  
set: 30  
dez: 30  
out: 31  
mar: 31  
abr: 31  
jun: 31  
nov: 30  
jan: 30  
fev: 31  
jul: 31  
mai: 31
```

Exemplo de Saída 4

```
que caloura ousada!
```

Exemplo de Entrada 5

```
um de marco  
1 mes  
um de abril  
ago: 28  
set: 30  
dez: 30  
out: 31  
mar: 31  
abr: 31  
jun: 31  
nov: 30  
jan: 30  
fev: 31  
jul: 31  
mai: 31
```

Exemplo de Saída 5

```
esta de brincadeira?
```

Exemplo de Entrada 6

```
trinta de agosto  
2 meses  
quatro de maio  
dez: 30  
out: 31  
mar: 31  
abr: 31  
mai: 31  
jun: 31  
ago: 28  
set: 30  
nov: 30  
jan: 30  
fev: 31  
jul: 31
```

Exemplo de Saída 6

```
data nao existe!
```

Esta página foi propositadamente deixada em branco.

Problema M

Eleição

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

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

As eleições municipais estão chegando e o prefeito de Fatecalandia quer construir postinhos de saúde pela cidade para conseguir se eleger! Ele pede para o secretário da Saúde da cidade para organizar isto. Temos várias localidades onde podem ser instalados os postinhos de saúde, cada localidade tem um custo de construção e a quantidade de pessoas atendidas. No entanto, a verba disponível para isto é escassa e o prefeito quer que sejam construídos os postinhos de saúde que atendam o maior número de pessoas, mas que respeitem a verba dada!

Entrada

Um número inteiro V ($1 \leq V \leq 10000$) que representa a verba disponível, um número N ($1 \leq N \leq 100$) que representa a quantidade de localidades e três números que representam o local ($1 \leq L \leq 100$), o custo de construção ($1 \leq C \leq 1000$) e o número de pessoas atendidas ($1 \leq P \leq 10000$).

Saída

A quantidade máxima de pessoas que poderão ser atendidas com a construção dos postos de saúde, considerando, é claro, a melhor combinação de acordo com o limite de verba.

Exemplo de Entrada 1

```
100
10
1 40 5
2 30 45
3 10 67
4 23 12
5 26 11
6 11 23
7 14 23
8 50 33
9 2 3
10 1 100
```

Exemplo de Saída 1

```
273
```