Apoios: http://crbonilha.com/pt/

16ª Maratona de Programação

30 de abril de 2016

Caderno de Problemas

Regras: ACM *International Collegiate Programming Contest*

Brasil – **Desafio ALGAR TELECOM**

(Este caderno contém 11 problemas; as páginas estão numeradas de 1 a 14)

Problema A
Como resolver?

Arquivo fonte: resolver.c, resolver.cpp ou resolver.java

O reino do mister Okada, conhecido como o japonês pensador, está em constante ameaça. Recentemente ele descobriu que seus inimigos estavam usando algumas mensagens codificadas para atacá-lo. Analisando algumas destas mensagens ele percebeu que as mesmas eram formadas por um conjunto de 3 números inteiros positivos (N1 N2 N3). Com sua inigualável inteligência, ele percebeu que N1 representava o início de um intervalo, N2 o final do intervalo, N3 a chave a ser encontrada neste intervalo. Se a chave existisse ou não no intervalo, uma sequência decodificada poderia ser gerada. Entretanto, os números pertencentes aos intervalos são gerados obedecendo a uma função matemática simples **`f(x) = 5x + 1`** e utilizam os números entre N1 e N2 na geração do intervalo de busca.

Veja o exemplo:

Se N1 = 0 , N2 = 8 e N3 = 16, ou seja, no intervalo gerado através de f(x), deseja-se encontrar o número 16. Como a função geradora é **`f(x) = 5x + 1, teremos:`**

```
f(N1) = 5*0 + 1 = 1
f(N1+1) = 5*1 + 1 = 6
......
......
f(N2) = 5*8 + 1 = 41
```

O intervalo a ser procurado seria `{1,6,16,21,26,31,36,41}`

Você precisa bolar uma forma simples de automatizar o processo.

Entrada

A entrada é formada de vários casos de testes. Cada caso é formado por uma única linha contendo 3 números (N1 N2 N3) separados por espaço em branco. Considere 0 ≤ N1 < N2 ≤ `1000000000`, 0 ≤ N3 ≤ `5000000001` números inteiros positivos. A última linha indicando o fim dos casos de testes contém N1 = N2 = N3 = 0.

Saída

Para cada caso de teste, deverá ser gerada apenas uma linha contendo a string "`Achou.`" se a chave foi encontrada e "`Não Achou.`", caso contrário.

Exemplos de Entradas

```
0 8 16
8 12 61
0 0 0 0
```

Exemplos de Saídas

```
Achou.
Achou.
```

Problema B

Mostre a Sequência

Arquivo fonte: sequencia.c, sequencia.cpp ou sequencia.java

O problema de encontrar o próximo termo de uma sequência de números é geralmente proposta em testes de QI. Sua tarefa é gerar os N termos de uma sequência a partir de uma codificação desta sequência.

Considere que $S = (S_i)$, $i \in \mathbb{N}$, descreva uma sequência de números reais onde o i-ésimo elemento é o S_i. Uma sequência constante é definida com o seguinte operador:

$$S = [n] \text{ onde } S_i = n \quad \forall\, i \in \mathbb{N}, \text{ onde } n \in \mathbb{Z}.$$

Outros operadores em uma determinada sequência de números $S = (S_i)$, $i \in \mathbb{N}$, são definidos da seguinte maneira:

$$V = [m + S] \quad \text{onde} \quad V_i = \begin{cases} m & , i = 1 \\ V_{i-1} + S_{i-1} & , i > 1 \end{cases}$$

$$V = [m * S] \quad \text{onde} \quad V_i = \begin{cases} m * S_1 & , i = 1 \\ V_{i-1} * S_i & , i > 1 \end{cases}$$

Onde $m \in \mathbb{N}$.

Veja alguns exemplos:

```
[2 + [1]]               = 2, 3, 4, 5, 6 ···
[1 + [2 + [1]]]         = 1, 3, 6, 10, 15, 21, 28, 36 ···
[2 * [1 + [2 + [1]]]] = 2, 6, 36, 360, 5400, 113400 ···
[2 * [5 + [- 2]]]       = 10, 30, 30, -30, 90, -450, 3150 ···
```

Sua tarefa é gerar os N termos de uma sequência, uma vez apresentada sua codificação.

Entrada

A entrada é formada de vários casos de testes. Cada caso é formado por uma única linha contendo a codificação (sem espaços), seguido de um espaço separando a codificação de um número N ($2 \leq N \leq 50$). A última linha contém o número zero indicando o fim das entradas.

Saída

Para cada caso de teste, deverá ser gerada apenas uma linha contendo a sequência de N termos separados por espaço em branco. Lembrando que o último número da sequência também deverá ser seguido de um espaço em branco.

Exemplos de Entradas

```
[2+[1]] 3
[2*[5+[-2]]] 7
0
```

Exemplos de Saídas

```
2 3 4
10 30 30 -30 90 -450 3150
```

Problema C
Festa de Aniversário

Arquivo fonte: niver.c, niver.cpp ou niver.java

Badião, um sultão famoso ama dar grandes festas de aniversário por um motivo simples: ele adora decorar o palácio. Badião gosta de utilizar correntes de papel penduradas no teto como a principal decoração. Entretanto, o sultão é muito cuidadoso com as combinações utilizadas nas confecções das correntes. Por exemplo, se existem quatro tipos diferentes de elos - A, B, C e D - o sultão pode dizer que "apenas as combinações ABB, BCA, BCD, CAB, CDD e DDA poderão ser utilizadas nas correntes na festa de hoje".
Ou seja, se o tamanho da corrente tiver 5 elos, então as únicas combinações seriam BCABB e BCDDA(elos, tais como ABBCA não poderia ser utilizado porque BBC não é uma combinação aprovada). Como Badião gosta de variedade, é importante saber o número total de correntes possíveis e as combinações dos elos a serem utilizados nas correntes.

Entrada

A entrada consistirá de vários casos de testes. Cada caso de teste será composto de duas linhas. A primeira linha conterá três números inteiros positivos ***n, l*** e ***m,*** onde ***n*** indica o número de tipos de elos, ***l*** é o número de elos por corrente, ou seja, o tamanho da corrente e ***m*** indica o número de combinações permitidas. O valor máximo para ***n, l e m*** serão 26, 100 e 600 respectivamente. A próxima linha contém as ***m*** combinações. Cada combinação terá o mesmo tamanho (entre 1 e 10) e serão separadas usando um simples espaço. Todas as combinações serão feitas utilizando apenas letras maiúsculas do alfabeto. O final dos casos de testes será determinado por uma linha contendo n = l = m =0.

Saída

Para cada caso de teste deverá ser gerado uma única linha indicando o número de correntes possíveis. Todas as respostas deverão estar no range de um inteiro de 32 bits.

Exemplos de Entradas

```
4 5 6
ABB BCA BCD CAB CDD DDA
5 4 5
E D C B A
4 8 3
AA BB CC
0 0 0
```

Exemplos de Saídas

```
2
625
3
```

Problema D
Promoção

Arquivo fonte: promocao.c, promocao.cpp ou promocao.java

Dr Luis Cláudio, um sujeito antenado com as promoções oferecidas pelo supermercado VemQueTem, o qual fica próximo à sua residência, anda muito sorridente ultimamente. Descobriu-se que ele foi sorteado em uma promoção oferecida pelo supermercado. Nesta promoção, a pessoa poderia entrar no supermercado, sozinho, e levar todos os produtos que pudesse carregar. Porém, algumas regras foram estabelecidas.

1) Entrar sozinho
2) Apenas um produto de cada tipo pode ser levado
3) Uma lista L contendo os preços e pesos dos produtos deve ser seguida
4) Um peso P máximo foi estabelecido

Você foi contratado pelo vizinho curioso do Dr Luis Cláudio para descobrir qual o valor total em mercadorias que ele conseguiu levar para casa.

Entrada

A entrada consiste de T casos de testes. Cada caso de teste começa com um inteiro N (1 ≤ N ≤ 100) que indica o número de produtos da lista L. As N linhas seguintes são formadas por 2 inteiros p e P. O primeiro inteiro, p (1 ≤ p ≤ 1000), representa o preço do produto. O segundo inteiro P (1 ≤ P ≤ 30) representa o peso do produto. A próxima linha contém um inteiro M, que indica o peso máximo permitido. O fim da entrada é representado por um 0.

Saída

Para cada caso de teste imprima um inteiro que representa o total dos produtos que Dr Luis Cláudio conseguir levar para casa.

Exemplos de entradas

```
4
72 17
44 23
31 24
22 2
26
3
72 17
44 23
31 24
25
0
```

Exemplos de saídas

```
94
72
```

Problema E

Contagem

Arquivo fonte: contagem.c, contagem.cpp ou contagem.java

Os camaradas Caetano e Ludovico são pessoas muito inquieta com a normalidade das coisas. Um certo dia, os camaradas estavam assistindo uma palestra e como estava meio chato, ficaram observando a quantidade de pessoas que ficavam entrando e saindo da sala. Como são conhecedores de soluções para IoT(internet das coisas) bolaram uma solução para contar o número de vezes que a porta se abria e fechava.

Por exemplo, se a pessoa saísse da sala, ligava um Flag, se a pessoa entrasse na sala ligava outro Flag. A cada 32 pessoas, era feito uma contagem e zerava o contador e o processo começava novamente. Como a solução funcionava não importa muito pois eles precisam apenas de uma ajuda sua. Fazer o programa que faz a contagem dos flags ligados.

Entrada

A entrada consiste de vários casos de testes. Cada caso de teste é formado de um único número inteiro positivo N representando o número que contém os flags. O final da entrada ocorre quando N = 0.

Saída

Para cada caso de teste da entrada deverá ser impresso um número inteiro representando o total de flags ligados do número N informado.

Exemplos de Entradas

```
10
5
1000
0
```

Exemplos de Saídas

```
2
2
6
```

Problema F

Noite no Museu

Arquivo fonte: noitenomuseu.c, noitenomuseu.cpp ou noitenomuseu.java

A cidade de Viena é chamada “cidade da cultura“ porque, entre outras coisas, abriga uma grande quantidade de museus, mais de 100. Como consequência, é muito difícil e caro visitar todos os museus, não importando o tempo que ficar na cidade. Entretanto, tem uma noite especial, chamada “Noite no Museu”, que se permite a visita a vários museus com apenas um ingresso, das 18:00h até a 01:00h da manhã do próximo dia. Porém, é impossível visitar todos os museus da cidade por duas razões principais. A primeira razão é que alguns museus em Viena não entram nessa promoção porque fecham às 17:00 h. A segunda razão é que não há tempo suficiente para visitar os museus no tempo de 7 horas. Sua tarefa é construir um programa que dado o número de museus participantes, o tempo necessário para visitar cada museu e o tempo que leva para ir de um museu ao outro, encontre o melhor “tour” para os visitantes, ou seja, visitar o maior número de museus nessa noite.

Entrada

A entrada contém vários casos de testes. A primeira linha de um caso de teste contém um inteiro N, que indicará o número de museus participantes na promoção (1 <= N <= 8). Cada museu tem um identificador único variando de 1 a N. A segunda linha contém N inteiros indicando o tempo, em minutos, necessário para visitar cada museu, de 1 a N. Então, teremos mais **n** linhas descrevendo o tempo para ir de um museu para todos os outros. A i-ésima linha contém **N** inteiros $\mathbf{M_k}$ (1 <= **k** <= N) representando o tempo, em minutos, para ir de um museu **i** para um museu **k**. Assuma que o i-ésimo inteiro na i-ésima linha é igual a **0**. O final da entrada é indicado por N=0.

Saída

Para cada caso de teste, seu programa deverá produzir uma linha contendo o número máximo de museus que podem ser visitados durante a noite.

Exemplos de Entradas

```
2
500 500
0 120
200 0
2
220 220
0 30
20 0
2
150 150
0 120
200 0
0
```

Exemplo de Saídas

```
0
1
2
```

Problema G

Passe Livre

Arquivo fonte: passelivre.c, passelivre.cpp ou passelivre.java

O primeiro Show de Rock de Terrânia prometia ser um sucesso, e desde muito cedo formaram se várias filas em cada entrada da arena de espetáculos. Quase na hora da abertura da arena as filas já tinham grande quantidade de pessoas.
Existiam várias regras sobre os direitos e deveres do público. Uma destas regras é que o portador de um ***passe verde***, teria direito de em uma fila assumir o lugar da pessoa a sua frente, caso seu número de ingresso fosse maior que o da pessoa a frente.
A organização, como ação promocional entregou um passe verde para a última pessoa de cada fila. Chamaremos esta última pessoa de ***k***. A pessoa ***k*** deveria comparar seu bilhete com a pessoa a sua frente (***k+1***), e caso seu ingresso tivesse numeração maior que o do próximo, deveria passar à frente do próximo (***k+1***), e poderia repetir o processo com a pessoa seguinte. Caso fosse menor, deveria entregar o ***passe verde*** ao próximo da fila (***k+1***), que adquiriria o direito de executar o mesmo procedimento a pessoa a sua frente ***(k + 2)***. Todo este processo se repetiria até que a primeira pessoa da fila fosse aquela com o ***passe verde***.
Ajude a organização a prever qual será a nova organização da fila após a primeira pessoa da fila ser aquela com o passe verde.

Entrada

A primeira linha da entrada contém um inteiro positivo ***N***, representando o número de entradas da arena, que é igual ao número de filas.
Cada uma das ***N*** linhas seguintes contém as informações da fila. O primeiro valor é um número inteiro ***P*** que representa a quantidade de pessoas na fila. Na sequência aparecem ***P*** pares de valores que representam o nome da pessoa e o número de seu ingresso no formato **X Y**. Todas as informações são separadas por um espaço.

Saída

Seu programa deve imprimir o nome das pessoas na fila após a passagem do ***passe verde***. Cada fila deve ser impressa Separando cada inteiro, existe um espaço.

Restrições:

$1 \leq N \leq 10000$ e $1 \leq Y \leq 100000$

Exemplos de Entradas

```
2
3 A 10 B 5 C 20
6 A 10 B 20 C 5 D 35 E 30 F 40
```

Exemplos de Saídas

```
B A C
A C B E D F
```

Problema H
Código de Permutação

Arquivo fonte: permutacao.c, permutacao.cpp ou permutacao.java

Como o proprietário de uma empresa de computação forense, você acabou de receber o seguinte email de um novo cliente:

"Eu, Little John, acabo de descobrir um jeito surpreendente para criptografar mensagens. Funciona da seguinte maneira:

Para começar, você precisa definir um conjunto de símbolos, chame-o de S, por exemplo, contendo as letras R A T E. O tamanho do conjunto S deve ser potência de 2 e a ordem dos símbolos de S é importante. Você deve observar que a letra R no conjunto S está na posição 0, A na posição 1, T na posição 2 e E na posição 3. Você também vai precisar de uma permutação P de todos os símbolos, por exemplo T E A R.
Finalmente, você vai precisar de um número inteiro, chame-o de x. Juntos, estes elementos compõem a chave. Dada uma chave, você agora está pronto para converter uma mensagem de texto M de comprimento n (M [0], M [1] ... M[n-1]), que tem alguns, mas não necessariamente todos os símbolos S, em uma seqüência C, também de comprimento n (C[0] C[1]...C[n-1]), que tem alguns, mas não necessariamente todos os símbolos de S.
O algoritmo faz a criptográfica da seguinte maneira:

1) ***Calcula um inteiro d como sendo o resto após dividir a parte inteira de ($n^{1.5}$ + x) por n, ou seja d = (int) ($n^{1.5}$ + x) % n, onde % é o operador que calcula o resto da divisão.***
2) ***Faça C[d] ser um símbolo pertencente a S, cuja posição é a mesma que a posição de M[d] em P.***
3) ***Para cada j ≠ d (0 .. n-1), faça C[j] ser o símbolo pertencente a S cuja posição é o valor obtido pelo xor da posição de M[j] pertencente a P com a posição de M[(j+1)%n] pertencente a S. O operador binário xor em Java, c, c++ é o "^"***

Como exemplo, considere um cenário where S = RATE, P = TEAR, x = 102, M = TEETER, e n = 6. Para calcular d, primeiro calcula-se $6^{1.5}$ + 102 = 116.696938, então pega-se o resto após dividir por 6. Logo d = 116 % 6 = 2. A tabela a seguir mostra os passos para gerar o texto criptografado C. Note que a ordem dos passos não é importante.

	0	1	2	3	4	5	
S	R	A	T	E			
P	T	E	A	R			
M	T	E	E	T	E	R	
C	E						M[0]=T, T está em P[0], M[1]=E, E está em S[3]. C[0]=S[0 xor 3]=S[3]
	E	T					M[1]=E, E está em P[1], M[2]=E, E está em S[3]. C[1]=S[1 xor 3]=S[2]
	E	T	A				2 é d. M[2]=E, E está em P[1], então. C[2]=S[1]
	E	T	A	E			M[3]=T, T está em P[0], M[4]=E, E está em S[3]. C[3]=S[0 xor 3]=S[3]
	E	T	A	E	A		M[4]=E, E está em P[1], M[5]=R, R está em S[0]. C[4]=S[1 xor 0]=S[1]
	E	T	A	E	A	A	M[5]=R, R está em P[3], M[0]=T, T está em S[2]. C[5]=S[3 xor 2]=S[1]

Eu incluí alguns exemplos adicionais de mensagens criptografadas no final deste email para você testar o algoritmo. Entretanto, primeiramente eu preciso informá-lo do processo de decodificação."

Infelizmente, ocorreu um erro ao enviar o email e não foi possível visualizar a explicação do algoritmo de decodificação. Dado sua notável habilidade em desvendar enigmas, sua grande missão é escrever o decodificador baseado em seu conhecimento do algoritmo de codificação.

Entrada

As entradas são constituídas dos pares {chave, mensagem codificada}. A chave está em 3 linhas separadas. A primeira linha contém o inteiro x (0 < x < 10000). A segunda linha contém a string S. A terceira linha contém a string P, que é a permutação de S. O tamanho de S (e consequentemente de P) será sempre uma das seguintes potências de 2: 2,4,8,16 ou 32. A próxima linha contém a mensagem codificada C, que terá no mínimo 1 e no máximo 60 caracteres. As strings S,

P e C não contem espaços em brancos, mas podem conter caracteres diferentes de letras e dígitos. A linha final contém um 0 indicando o fim da entrada.

Saída

Para cada caso de teste, deverá ser gerado apenas uma linha contendo a mensagem decodificada.

Exemplos de Entradas

```
102
RATE
TEAR
ETAEAA
32
ABCDEFGHIJKLMNOPQRSTUVWXYZ._!?,;
;ABCDEFGHIJKLMNOPQRSTUVWXYZ._!?,
MOMCUKZ,ZPD
0
```

Exemplos de Saídas

```
TEETER
HELLO_WORLD
```

Problema I

Contando (Streams de) Números

Arquivo fonte: streams.c, streams.cpp ou streams.java

Imagine que você tenha que contar o número de valores distintos que existem em uma sequência. Fácil, né?! O problema é que essa sequência é um stream. Isso significa que a sequência pode ser muito grande (até mesmo infinita!) e que você jamais poderia armazenar todos os números em memória e rapidamente computar quantos são distintos.

Diante desse pequeno problema, vamos tentar executar um algoritmo que consegue uma solução aproximada?! É assim: considere um vetor bitmap com L bits, inicializado com 0. Para cada número x que chegar no stream, aplique uma função hash h(x). Depois, obtenha a posição do bit 1 menos significativo da representação binária de h(x), isto é: lsb(h(x)) (least significant bit). Atualize para 1 o valor no bitmap na posição obtida. Caso h(x) = 0, o bitmap não é atualizado. Ao final, a posição do bit 1 mais à esquerda, ajudará a indicar o número de valores distintos da sequência. Dessa maneira, a única estrutura que você tem que armazenar na memória é o bitmap e, de maneira aproximada, você conseguirá contar os números distintos. Interessante, né?!

Como exemplo, considere a sequência 1,3,2,1. Considere também a função hash h(x) = (3x+1) mod 7 e um bitmap de L = 8 bits. Assim, temos:

Bitmap inicializado:

posição	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
valor	0	0	0	0	0	0	0	0

	BITMAP
	00000000
x = 1 -> h(x) = 4 -> 0100 -> lsb(h(x)) = 2	00000100
x = 3 -> h(x) = 3 -> 0011 -> lsb(h(x)) = 0	00000101
x = 2 -> h(x) = 0	00000101
x = 1 -> h(x) = 4 -> 0100 -> lsb(h(x)) = 2	00000101

Ao final, o bitmap resultante é: 00000101.

Entrada

A entrada é composta por vários casos de teste.
Cada caso de teste é uma sequência de N números inteiros (1 < N < 100000) números inteiros i (0 ≤ i ≤ 65535) separados por vírgula. Considere h(x) = (3x + 1) mod 7 e L = 8 bits. O final da entrada é indicado por final de arquivo (EOF).

Saída

Para cada caso de teste imprima o bitmap resultante do processamento da sequência. Considere sempre o bitmap com L = 8 bits.

Exemplos de Entradas

```
1,3,2,1
3,5,6,7,8,9,2,3
```

Exemplos de Saídas

```
00000101
00000111
```

Problema J
Vírus H1N1

Arquivo fonte: h1n1.c, h1n1.cpp ou h1n1.java

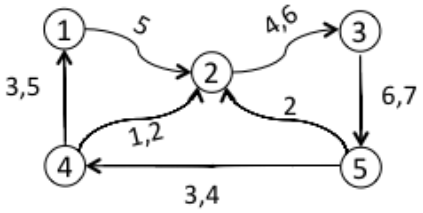
Ariosvaldo é um (raro) deputado preocupado com a saúde do Brasil. De olho no surto do vírus H1N1 no país, ele quer contratar (licitamente) um programador esperto, capaz de prever qual a velocidade mínima de contágio entre duas pessoas, baseada nos contatos que elas tiveram com outras pessoas.

Você rapidamente já pensou na solução: modelar o problema como um grafo temporal e obter os caminhos mais rápidos entre dois nós! Um grafo temporal é aquele em que as arestas existem apenas durante o instante de interação entre dois nós e depois desaparecem. Ele pode ser definido da seguinte maneira:

- Uma aresta temporal é do tipo *(u,v,i,)*, onde *u* é o nó de origem, *v* é o nó de destino e *i* é o instante de interação entre *u* e *v*.
- R é o tempo de retenção dos nós, ou seja, o tempo entre a chegada do vírus em um nó até o momento mínimo que ele pode ser repassado adiante.

Como exemplo, considere a seguinte rede temporal e R = 1 (na figura, os rótulos nas arestas indicam os instantes dos contatos):

(1,2,5)	(4,1,5)
(2,3,4)	(4,2,1)
(2,3,6)	(4,2,2)
(3,5,6)	(5,2,2)
(3,5,7)	(5,4,3)
(4,1,3)	(5,4,4)



Um caminho temporal é uma sequência de arestas temporais conectando uma sequência de nós e respeitando a ordem crescente dos tempos de contato e o tempo necessário de retenção R. Os caminhos temporais entre os vértices 4 e 5 do grafo acima são:

Caminho	Duração
C1: <(4,1,3),(1,2,5),(2,3,6),(3,5,7)>	4
C2: <(4,2,1),(2,3,4),(3,5,6)>	5
C3: <(4,2,1),(2,3,4),(3,5,7)>	6
C4: <(4,2,2),(2,3,4),(3,5,6)>	4
C5: <(4,2,2),(2,3,4),(3,5,7)>	5
C6: <(4,2,1),(2,3,6),(3,5,7)>	6
C7: <(4,2,2),(2,3,6),(3,5,7)>	5

A duração d(C) de um caminho C:<$(u_1,u_2,t_1),(u_2,u_3,t_2),...,(u_n,u_{n+1},t_n)$> é $d(C) = t_n - t_1$. O caminho mais rápido é aquele com menor duração. No exemplo acima, C1 e C4 são os caminhos mais rápidos entre 4 e 5.

Com tudo isso, o que o deputado Ariosvaldo precisa é: determinar a duração do caminho mais rápido que existe de um nó *a* (pessoa infectada) para um nó *b* (pessoa que poderá ser infectada) de um grafo temporal. A licitação está no papo!

Entrada

A entrada começa com um inteiro N indicando o número de casos de teste. Cada caso de teste contém um inteiro R indicando o tempo de retenção para todos os nós e, na linha seguinte, um inteiro A indicando o número de arestas. Em cada uma das A linhas seguintes uma aresta temporal *(u,v,i)* é definida por 3 inteiros separados por "," (sem aspas). A última linha contém dois inteiros separados por espaço indicando os nós *a* e *b*. Considere que:

- O grafo é dirigido
- Não existe self-loop (uma aresta que conecta um nó a ele mesmo)
- Os valores estão sempre na mesma unidade de tempo
- O tempo de travessia de uma aresta deve ser desconsiderado, ou seja, se existe contato entre dois nós, o vírus é transmitido instantaneamente.
- $1 \leq N,R \leq 10$
- $1 \leq u,v,a,b,i,A \leq 20$

Saída

Para cada caso de teste, imprima a duração do caminho mais rápido de *a* para *b*. Caso não exista caminho ou o caminho seja composto por apenas uma aresta, a duração é zero.

Exemplos de Entradas

```
2
1
12
1,2,5
2,3,4
2,3,6
3,5,6
3,5,7
4,1,3
4,1,5
4,2,1
4,2,2
5,2,2
5,4,3
5,4,4
4 5
1
5
1,2,5
2,3,5
2,3,6
3,5,6
3,5,7
1 2
```

Exemplos de Saídas

```
4
0
```

Problema K

Falha de impressão

Arquivo fonte: falha.c, falha.cpp ou falha.java

Sr Guigui, um funcionário exemplar está muito chateado. Após anos de trabalho sem cometer falhas, ele descobriu que vários protocolos foram informados aos clientes erroneamente. O motivo do erro foi devido a utilização de um computador que tinha o teclado com defeito. Sr Guigui percebeu que o teclado tinha uma falha em um dígito numérico, isto é, o dígito com problema não era impresso sobre a folha, como se a tecla não tivesse sido pressionada. Preocupado com o atendimento aos clientes, o Sr. Guigui quer saber, a partir dos valores originais dos protocolos (que ele mantinha em notas manuscritas) quais foram os números de protocolos gerados erroneamente.

Por exemplo, se o dígito com falha no teclado fosse o 5 e o valor original do protocolo fosse 1500, então o valor informado erradamente seria o 100, porque o dígito 5 não seria impresso. Note que o Sr. Guigui quer saber o valor numérico do protocolo, ou seja, (ainda considerando o exemplo anterior) o número 5000 corresponde ao valor numérico 0, não 000.

Entrada

A entrada consiste de vários casos de testes. Cada caso de teste é formado de 2 números inteiros D e N separados por um espaço em branco. Considere que $1 \leq D \leq 9$ e $1 \leq N < 10100$ e onde D representa o dígito que apresentou a falha e N representa o número original do protocolo.

O final da entrada ocorre quando N = D = 0.

Saída

Para cada caso de teste da entrada deverá ser impresso um número inteiro representando o protocolo enviado erroneamente para o cliente.

Exemplos de Entradas

```
5 5000000
3 123456
9 23454324543423
9 99999999991999999
7 777
0 0
```

Exemplos de Saídas

```
0
12456
23454324543423
1
0
```