7^a Seletiva da UFPR

5 de Agosto de 2016

Sevidor BOCA:
`http://maratona.c3sl.ufpr.br/`

Flávio Zavan
Ricardo Oliveira

Instruções Importantes

- Em cada problema, cada arquivo de entrada contém apenas um caso de teste. Sua solução será executada com vários arquivos de entrada.
- Se a solução der erro ou esgotar o tempo limite para um dado arquivo de entrada, você receberá a indicação de erro (estouro de tempo, resposta errada, etc.) para aquele arquivo, e a execução terminará. O arquivo que causou o erro não é identificado. Note que pode haver outros erros, de outros tipos, para outros arquivos de entrada, mas apenas o primeiro erro encontrado é reportado.
- Sua solução será compilada com a seguinte linha de comando:
  - C: `gcc -static -O2 -lm`
  - C++: `g++ -static -O2 -lm`
  - C++11: `g++ -std=c++11 -static -O2 -lm`
  - Java: `javac`
  - Pascal: `fpc -Xt -XS -O2`
- Sua solução deve processar cada arquivo de entrada no tempo máximo estipulado para cada problema, dado pela seguinte tabela:

Problema	**Nome**	**Tempo Limite (segundos)**
A	Ajuda da Geógrafa	1
B	Iu-di-oh	1
C	Guerra à Nlogônia	3
D	Analógimôn Go	2
E	Reinauguração do CEI	1
F	Quiz Universitário	1
G	Jogatina UFPR	1
H	Manyfile	1
I	FHBZMIPS	1

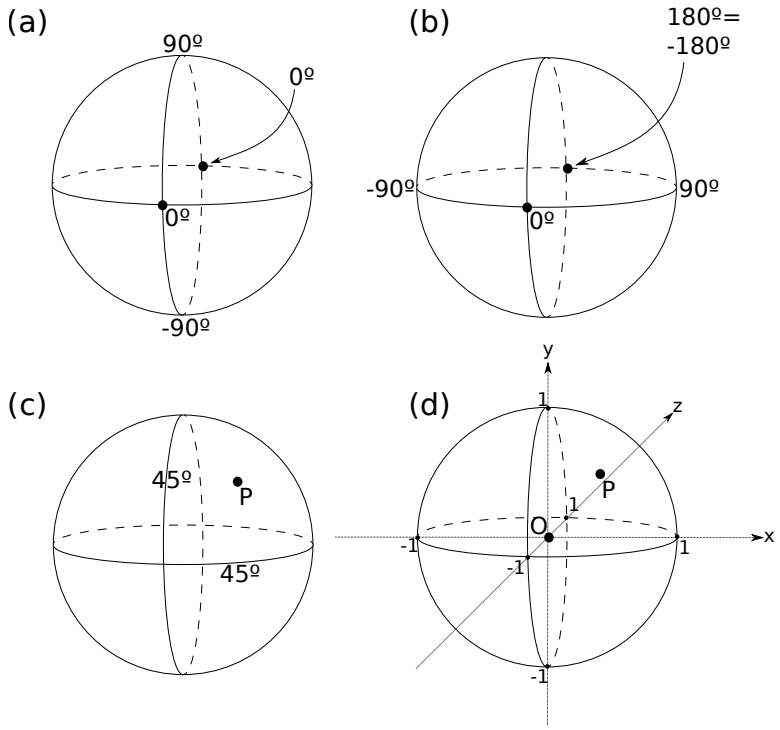
- Os juizes usam um sistema de 64 bits (idêntico às maquinas do DINF).
- Todas as linhas, tanto na entrada quanto na saída, terminam com o caractere de fim-de-linha ($\backslash n$), mesmo quando houver apenas uma única linha no arquivo.
- Para submissões em **JAVA**, a classe deverá ter o mesmo nome que o *basename* do problema (leia a linha entre o título e o texto do problema).

A: Ajuda da Geógrafa

`Arquivo: geografa.[c|cpp|java|pas]`

América e Vinícius estão estudando o sistema de coordenadas geográficas em esferas, com o qual é possível determinar a posição de qualquer ponto na superfície de uma esfera. Sua amiga Maísa, que é geógrafa, decidiu ajudá-los nesses estudos. Ela então esclareceu o funcionamento deste sistema de coordenadas.

Neste sistema, um ponto é determinado por sua *latitude* e sua *longitude*. A latitude varia de -90^o a 90^o e indica o ângulo do ponto em relação ao centro da esfera, no sentido vertical, como indica a figura (a). A longitude, por sua vez, varia de -180^o a 180^o e indica o ângulo do ponto em relação ao centro da esfera, no sentido horizontal, como indica a figura (b). Como exemplo, a figura (c) indica a posição de um ponto P com latidude 45^o e longitude 45^o.



Considere que o centro da esfera está na origem $O = (0, 0, 0)$ do sistema cartesiano do espaço, e, olhando a esfera de frente, o eixo x a cruza da esquerda para a direita, o eixo y de baixo para cima, e o eixo z do ponto menos profundo ao mais profundo. A figura (d) indica um exemplo para uma esfera de raio 1, onde o ponto P está em $(1/2, \sqrt{2}/2, -1/2)$.

Dado o raio da esfera e as coordenadas geográficas de um ponto P, determine suas coordenadas no sistema cartesiano do espaço.

Entrada

A entrada contém uma única linha contendo três inteiros r, la e lo ($1 \leq r \leq 50, -90 \leq la \leq 90, -180 \leq lo < 180$), indicando o raio da esfera, e a latitude e a longitude do ponto P, em graus.

Saída

Imprima uma única linha contendo três valores x, y e z, separados por espaço, indicando as coordenadas do ponto P. Note que o ponto impresso deve necessariamente estar a uma distância r da origem $O = (0, 0, 0)$. Arredonde e imprima cada coordenada com exatamente duas casas decimais.

Exemplo de entrada	**Exemplo de saída**
`1 45 45`	`0.50 0.71 -0.50`

Exemplo de entrada	**Exemplo de saída**
`1 0 0`	`0.00 0.00 -1.00`

Exemplo de entrada	**Exemplo de saída**
`1 90 0`	`0.00 1.00 0.00`

Exemplo de entrada	**Exemplo de saída**
`1 45 90`	`0.71 0.71 0.00`

B: Iu-di-oh

`Arquivo: iudioh.[c|cpp|java|pas]`

Iu-di-oh! é um jogo de cartas que virou uma verdadeira febre entre os jovens! Todo jogador de *Iu-di-oh!* tem seu próprio baralho, contendo várias cartas do jogo. Cada carta contém N *atributos* (como força, velocidade, inteligência, etc.). Os atributos são numerados de 1 a N e são dados por inteiros positivos.

Uma partida de *Iu-di-oh!* é sempre jogada por dois jogadores. Ao iniciar a partida, cada jogador escolhe exatamente uma carta de seu baralho. Após as escolhas, um atributo é sorteado. Vence o jogador cujo atributo sorteado em sua carta escolhida é maior que na carta escolhida pelo adversário. Caso os atributos sejam iguais, a partida empata.

Marcos e Leonardo estão na grande final do campeonato brasileiro de *Iu-di-oh!*, cujo prêmio é um Dynavision[1]. Dados os baralhos de ambos, a carta escolhida por cada um e o atributo sorteado, determine o vencedor!

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 100$), o número de atributos de cada carta. A segunda linha contém dois inteiros M e L ($1 \leq M, L \leq 100$), o número de cartas no baralho de Marcos e de Leonardo, respectivamente.

As próximas M linhas descrevem o baralho de Marcos. As cartas são numeradas de 1 a M, e a i-ésima linha descreve a $i-$ésima carta. Cada linha contém N inteiros $a_{i,1}, a_{i,2}, ..., a_{i,N}$ ($1 \leq a_{i,j} \leq 10^9$). O inteiro $a_{i,j}$ indica o atributo j da carta i. As próximas L linhas descrevem o baralho de Leonardo. As cartas são numeradas de 1 e L e são descritas de maneira análoga.

A próxima linha contém dois inteiros C_M e C_L ($1 \leq C_M \leq M, 1 \leq C_L \leq L$), as cartas escolhidas por Marcos e Leonardo, respectivamente. Por fim, a última linha contém um inteiro A ($1 \leq A \leq N$) indicando o atributo sorteado.

Saída

Imprima uma linha contendo "`Marcos`" se Marcos é o vencedor, "`Leonardo`" se Leonardo é o vencedor, ou "`Empate`" caso contrário (sem aspas).

Exemplo de entrada	**Exemplo de saída**
3 2 2 3 8 1 6 7 9 1 2 3 8 4 1 1 2 2	Marcos

Neste exemplo, $A = 2$, Marcos é o vencedor e logo a saída é uma linha contendo `Marcos`. No mesmo exemplo, mas com $A = 1$, Leonardo vence, e a saída é uma linha contendo `Leonardo`. Se $A = 3$, a partida empata, e saída é uma linha contendo `Empate` .

[1] que é quase um *Playstation* 2!

C: Guerra à Nlogônia

`Arquivo: guerra.[c|cpp|java|pas]`

A República Federal do Paranauê (RFPR) está em guerra contra a Província da Nlogônia! Durante a guerra, os soldados Nlogonenses utilizam um esquema de criptografia para enviar mensagens entre si. Felizmente para a RFPR, espiões descobriram como o esquema funciona!

No início de cada dia, uma *string* S é informada a todos os soldados. A *string* tem tamanho N e seus caracteres são numerados de 1 a N, da esquerda para a direita. Toda mensagem criptografada é acompanhada de dois números i e j, entre 1 e N, inclusive. O soldado que recebe a mensagem conta a quantidade de ocorrências de cada caracter entre `a` e `z` na *string* S entre as letras i e j, inclusive. Esta contagem serve de chave para descriptografar a mensagem.

Entretanto, para dificultar o trabalho dos espiões, o quartel-general (QG) da Nlogônia pode alterar partes da *string* S várias vezes ao dia! Para alterar a *string*, o QG envia a todos seus soldados três números i, j e p, indicando que cada letra entre i e j, inclusive, deve ser alterada para a letra a p posições seguintes no alfabeto (por exemplo, se $p = 1$, cada letra `a` entre i e j é alterada para `b`, cada `b` é alterado para `c`, etc. O alfabeto é cíclico, logo cada `z` é alterado para `a` neste exemplo).

Você é um agente da RFPR e descobriu todos os momentos em que o QG altera a *string* e em que uma mensagem é enviada. Sua tarefa é determinar, para cada mensagem, a contagem de cada caracter entre `a` e `z` entre i e j em S.

Entrada

A primeira linha contém dois inteiros N e Q ($1 \leq N \leq 10^5, 1 \leq Q \leq 2 \times 10^5$), o tamanho da *string* e o número de operações, respectivamente. A segunda linha contém a *string* inicial S. A *string* contém N caracteres de `a` a `z`, inclusive. As próximas Q linhas descrevem uma operação cada. Uma operação descrita pela linha `M` i j ($1 \leq i \leq j \leq N$) indica uma mensagem enviada. Uma operação descrita pela linha `A` i j p ($1 \leq i \leq j \leq N, 1 \leq p \leq 25$) indica uma alteração da *string* S pelo QG da Nlogônia.

Saída

Para cada mensagem enviada, imprima uma linha com 26 inteiros separados por espaços, contendo o número de ocorrências das letras `a`, `b`, ..., `z`, como especificado.

Exemplo de entrada	**Exemplo de saída**
20 5 invadamarfprdiacinco M 1 20 A 3 6 1 M 4 10 A 2 19 3 M 1 2	4 0 2 2 0 1 0 0 3 0 0 0 1 2 1 1 0 2 0 0 0 1 0 0 0 0 1 2 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

D: Analógimôn Go

`Arquivo: analogimongo.[c|cpp|java|pas]`

Cansado de esperar pela chegada de um certo jogo na pequena República de Tangyow, seu presidente Abdran criou seu próprio jogo, *Analógimôn Go!*.

Tangyow tem N cidades, numeradas de 1 a N, ligadas através de M estradas de mão dupla, onde cada estrada leva um certo tempo para ser atravessada.

O presidente Abdran colocou, em cada cidade, uma conjunto de monstrinhos, chamados *analógimôns*. Cada analógimôn é de uma espécie identificada por um inteiro entre 1 e 10^9 inclusive. Para cada cidade i existe uma lista L_i de inteiros e um número U_i. Os analógimôns presentes na cidade i têm espécies cujos identificadores são menores ou iguais a U_i e que são múltiplos de algum número em L_i. Como exemplo, se $U_i = 10$ e $L_i = (3, 5, 7)$, a cidade i tem analógimôns de espécies $3, 5, 6, 7, 9$ e 10. Não há mais de um analógimôn da mesma espécie na mesma cidade, mas analógimôns da mesma espécie podem ocorrer em cidades distintas. Quando um jogador chega em uma cidade, ele *obrigatoriamente* captura *todos* os analógimôns nela. Capturar um analógimôn leva exatamente 1 minuto, e deve-se capturar um de cada vez.

Todos os P tangyowenses estão na cidade 1 e irão para a cidade N capturando analógimôns no caminho. Pelas regras, os caminhos utilizados por cada tangyowense devem ser *disjuntos em cidades* entre si, isto é, para cada cidade (exceto 1 e N), no máximo um jogador pode passar por ela. Sua tarefa é encontrar uma rota para cada tangyowense de tal forma que o somatório do tempo total levado por cada jogador é mínimo.

Entrada

A primeira linha contém os inteiros N, M e P ($2 \leq N \leq 50$, $1 \leq M \leq \frac{N \times (N-1)}{2}$, $1 \leq P \leq 50$). As próximas N linhas descrevem as cidades. Cada cidade é descrita pelo inteiro U_i ($0 \leq U_i \leq 10^9$), um inteiro T_i ($0 \leq T_i \leq 16$) indicando o tamanho da lista L_i, seguido por T_i inteiros distintos, entre 1 e 10^9 inclusive, indicando a lista L_i. As próximas M linhas descrevem as estradas. Cada estrada é descrita por três inteiros a, b e t ($1 \leq a, b \leq N, a \neq b, 1 \leq t \leq 10^9$), indicando uma estrada entre as cidades a e b que é atravessada em t minutos.

É garantido que não há analógimôns nas cidades 1 e N, e também não há uma estrada ligando ambas as cidades diretamente.

Saída

Imprima uma linha com o mínimo somatório do tempo total levado por cada jogador possível. Se não há P caminhos distintos da cidade 1 para a cidade N, imprima a frase "`Espere Pokemon Go Chegar`" (sem aspas).

Exemplo de entrada	Exemplo de saída
7 10 2 0 0 10 3 3 5 7 5 1 2 10 1 2 8 2 2 3 20 1 1 0 1 1 1 2 5 2 3 1 3 7 3 1 4 1 4 5 5 5 7 1 2 5 2 3 4 3 4 6 4 5 6 2	33

Exemplo de entrada	Exemplo de saída
4 4 3 0 0 10 3 1 5 7 8 2 5 6 0 0 1 2 30 1 3 40 2 4 8 3 4 10	Espere Pokemon Go Chegar

E: Reinauguração do CEI

`Arquivo: cei.[c|cpp|java|pas]`

Para comemorar a reinauguração do espaço físico do Clube de Espanhóis Inteligentes (CEI), uma grande festa está acontecendo no clube neste exato momento!

Mateuz é um integrante do CEI que está ajudando na organização da festa. Sempre que um convidado chega ou vai embora da festa, Mateuz anota em um papel quantos minutos se passaram desde o início da festa até aquele momento.

Mateuz acabou de repassar os números anotados para os presidentes do CEI, Freitaz e Rodriguez. Note que os presidentes têm apenas os minutos em que convidados entraram e sairam da festa. Desta forma, para cada minuto recebido, Freitaz e Rodriguez não sabem se o convidado estava entrando ou saindo naquele momento. Sabe-se apenas que: a festa começou sem convidados; até este exato momento, nenhum convidado entrou na festa mais de uma vez; e, neste exato momento, não há convidados na festa, isto é, todos os convidados foram embora (pois foram participar de uma competição de programação, mas pretendem voltar à festa depois). Os números anotados também são todos distintos entre si, mas não são dados necessariamente em ordem.

Sua tarefa é ajudar Freitaz e Rodriguez a determinar qual o maior número possível de convidados que podem ter estado na festa *simultaneamente* em algum momento. Determine também a quantidade máxima de minutos que esta quantidade de convidados pode ter estado na festa simultaneamente.

Entrada

A primeira linha contém um inteiro N ($2 \leq N \leq 1000$), a quantidade de números anotados. A segunda linha contém N inteiros distintos $m_1, m_2, ..., m_N$, os números anotados por Mateuz e recebidos por Freitaz e Rodriguez. Para cada $1 \leq i \leq N$, o número m_i ($1 \leq m_i \leq 10^4$) indica que um convidado entrou ou saiu da festa m_i minutos após seu início.

Saída

Imprima uma linha com dois inteiros separados por um espaço. O primeiro é o maior número possível de convidados que podem ter estado na festa simultaneamente. O segundo é a quantidade de máxima de minutos que esta quantidade de convidados pode ter estado simultaneamente na festa.

Exemplo de entrada	**Exemplo de saída**
2 1 2	1 1

Exemplo de entrada	**Exemplo de saída**
4 7 3 8 1	2 4

F: Quiz Universitário

Arquivo: quiz.[c|cpp|java|pas]

A universidade está promovendo o *Quiz Universitário*, um jogo de perguntas e respostas sobre a universidade! O participante de hoje é Fernando, um jovem aluno da Computação.

Existem N perguntas, numeradas de 1 a N. As perguntas são feitas para Fernando em sequência, e se ele acertar a pergunta i, ele ganha P_i reais como prêmio! Entretanto, se ele errar uma pergunta, o jogo termina. Desta forma, se Fernando errar a pergunta 1, o jogo termina e ele não ganha nenhum prêmio; se acertar a pergunta 1 mas errar a pergunta 2, ele ganha apenas o prêmio da pergunta 1; se acertar as perguntas 1 e 2 mas errar a 3, ele ganha apenas o prêmio das perguntas 1 e 2; etc. O jogo também termina se todas as N perguntas forem acertadas. Neste caso, ele ganha a soma dos prêmios de todas as perguntas.

Fernando também pode usar até K *pulos*. Ao pular uma pergunta, ele ganha o prêmio da pergunta e o jogo continua. Na prática, o efeito de pular uma pergunta é o mesmo de acertá-la, mas sem respondê-la de fato. Para cada pergunta i, Fernando sabe que a chance dele acertar a pergunta i, caso não a pule, é de C_i %. Ele quer determinar quais perguntas ele vai pular (caso chege nelas) *antes* de começar o *Quiz*. Ajude-o a determinar quais perguntas ele deve pular, de tal forma que o prêmio total esperado seja máximo.

Entrada

A primeira linha contém os inteiros N e K ($1 \leq N \leq 1000, 0 \leq K \leq N$), o número de perguntas e o número máximo de pulos. A segunda linha contém N inteiros $P_1, P_2, ..., P_N$ ($1 \leq P_i \leq 100$), o prêmio de cada pergunta. A terceira linha contém N inteiros $C_1, C_2, ..., C_N$ ($0 \leq C_i \leq 100$), indicando a chance de Fernando acertar a pergunta i, sem pulá-la, em %.

Saída

Imprima uma linha com o prêmio total máximo que Fernando pode obter, em reais, arredondado com duas casas decimais.

Exemplo de entrada	**Exemplo de saída**
3 1 30 100 50 50 5 40	75.00

Exemplo de entrada	**Exemplo de saída**
3 2 30 100 50 50 5 40	150.00

No primeiro exemplo, a melhor estratégia é usar o único pulo na pergunta 2. Desta forma, ele tem 50% de chance de ganhar $P_1 = 30$ reais, 50% de ganhar $P_2 = 100$ reais (esta chance não é de 100% mesmo pulando esta pergunta, pois ele precisa ter acertado a primeira para ganhar esse prêmio), e 20% de ganhar $P_3 = 50$ reais. O prêmio total esperado é de $0.50 \times 30 + 0.50 \times 100 + 0.20 \times 50 = 75$. Não há outra estratégia cujo prêmio total esperado é maior.

G: Jogatina UFPR

`Arquivo: jogatina.[c|cpp|java|pas]`

Assim como a maioria dos estudantes de computação, você vive jogando os jogos eletrônicos mais populares atualmente: *League of Legends* (LOL) e *Counter-Strike* (CS). Embora você também jogue LOL, você gosta mais é de usar todas suas grandes habilidades para derrotar a equipe terrorista em *Counter-Strike*! Você é tão empenhado no combate ao terror que é frequentemente comparado com o presidente dos EUA que anunciou a captura e derrota de um grande terrorista da vida real.

Por ser bastante habilidoso, os vídeos de suas jogadas (seus famosos *gameplays*) vivem aparecendo na *Jogatina UFPR*, uma página na internet que publica *gameplays* de alunos da nossa universidade.

A página publica muitos vídeos diariamente. Por isso, pode ser dificil encontrar e contar todos os seus vídeos na página. Entretanto, como você também é programador, você decidiu escrever um programa para auxiliá-lo nesta tarefa. Dada a lista de *gameplays* publicados na página, determine quantos *gameplays* seus de *Counter-Strike* foram publicados.

Entrada

A primeira linha contém dois inteiros N e I ($1 \leq N \leq 10^4, 1000 \leq I \leq 9999$), o número de *gameplays* publicados na página e o seu identificador na universidade, respectivamente.

As próximas N linhas descrevem os *gameplays* publicados. Cada *gameplay* é descrito por dois inteiros i e j ($1000 \leq i \leq 9999, j = 0$ ou 1), onde i é o identificador do autor do *gameplay* na universidade, e $j = 0$ se o *gameplay* é de *Counter-Strike*, ou $j = 1$ se é de *League of Legends*.

Saída

Imprima uma única linha com um número indicando quantos *gameplays* seus de *Counter-Strike* foram publicados na página.

Exemplo de entrada	**Exemplo de saída**
7 5558 5693 1 5558 0 6009 1 5558 1 1566 0 5558 0 8757 1	2

H: Manyfile

`Arquivo: manyfile.[c|cpp|java|pas]`

No ano de 2569, Vasya recebe de sua mãe um grandioso presente de aniversário, o código fonte do seu vídeo-game favorito, Aranha Paciente. Vasya corre direto ao seu computador, com 4096 núcleos de processamento, insere o disquete, dá um `ls` no diretório do código e nota que ele é composto de N arquivos fonte e um Manyfile.

Um Manyfile é como uma receita de bolo para compilar o código. Ao se executar o comando `many`, o Manyfile é lido e os arquivos começam a ser compilados, de forma que o máximo de núcleos de processamento são utilizados simultaneamente. Se o mundo fosse perfeito, este processo seria muito rápido, uma vez que **cada arquivo fonte do jogo demora exatamente um minuto para ser compilado**, mas infelizmente a compilação de alguns arquivos depende da conclusão de outros, impossibilitando que todos os arquivos sejam processados simultaneamente.

Considerando a compilação da Aranha Paciente como terminada quando todos os seus N arquivos tiverem sido compilados e sabendo quais arquivos dependem de qual, escreva um programa que calcule para Vasya quantos minutos demorará para que a Aranha Paciente seja compilada.

Entrada

A primeira linha da entrada contém um inteiro N ($1 \leq N \leq 1000$), o número de arquivos fonte da Aranha Paciente. Os arquivos são numerados de 1 a N. As N linhas seguintes descrevem os arquivos. A i-ésima linha contém um inteiro M_i ($0 \leq M_i < N$) seguido de M_i inteiros com valor entre 1 e N e diferentes de i, representando o índice dos arquivos dos quais o arquivo i depende.

Saída

Imprima uma única linha contendo o tempo total em minutos que demorará para que a Aranha Paciente seja compilada. Caso seja impossível terminal a compilação, imprima `-1`.

Exemplo de entrada	**Exemplo de saída**
2 1 2 1 1	-1

Exemplo de entrada	**Exemplo de saída**
3 0 1 3 0	2

I: FHBZMIPS

`Arquivo: fhbzmips.[c|cpp|java|pas]`

O FHBZMIPS é um novo processador desenvolvido pela Neboscorp (r). Sua memória interna contém apenas um único registrador r, de 8 bits. Seu conteúdo é sempre interpretado como um inteiro sem sinal, isto é, é possível respresentar inteiros de 0 a 255 em seu registrador.

O valor inicial do registrador é 0. Além disso, o FHBZMIPS suporta as seguintes operações:

Instrução	Operação	Descrição
`add` n	$r \leftarrow r + n$	Soma n unidades no registrador
`sub` n	$r \leftarrow r - n$	Decrementa n unidades do registrador
`mul` n	$r \leftarrow r \times n$	Multiplica o valor do registrador por n
`div` n	$r \leftarrow r/n$	O registrador recebe o quociente de sua divisão por n
`and` n	$r \leftarrow r$ AND n	Operação E bit-a-bit com n
`or` n	$r \leftarrow r$ OR n	Operação OU bit-a-bit com n
`xor` n	$r \leftarrow r$ XOR n	Operação OU-exclusivo bit-a-bit com n
`gotoif` n I	Pular para I se $r \geq n$	Se o registrador tem valor maior ou igual a n, vá para a instrução de número I
`halt`	Desligar	Termina a execução do programa

Ocorrências de *overflow*, que ocorrem quando não é possível representar o resultado de alguma operação no registrador, são tratadas como em outros processadores, onde apenas o resto da divisão do resultado por 256 é mantido. Assim, por exemplo, se o registrador contém 240 e a instrução `add 20` é executada, então o registrador passa a conter 4. Se o registrador contém o valor 0 e executa-se `sub 2`, então passa a conter 254. Se o registrador contém 25 e executa-se `mul 25`, passa a conter 113.

Marcelo acabou de escrever um programa em *assembly* do FHBZMIPS. Sua tarefa é determinar o valor do registrador ao término da execução de seu programa, ou determinar se o programa é executado infinitamente.

Entrada

A primeira linha contém um inteiro N ($1 \leq N \leq 100$), o número de instruções no programa. As próximas N linhas descrevem o programa, uma instrução por linha.

Cada linha inicia com um inteiro indicando o número i da instrução. É garantido que este número é sequencial, isto é, a primeira instrução é a de número 1, a segunda instrução é a de número 2, etc. A linha é seguida pela descrição da instrução, como na tabela acima. Onde for aplicável, $0 \leq n \leq 255$, $1 \leq I \leq N$,e $I \neq i$.

É garantido que há uma única instrução `halt` no programa, e que ela é sempre a instrução de número N, isto é, a última instrução do programa. Também é garantido que $n \neq 0$ para toda instrução `div`.

Saída

Imprima uma linha contendo o valor do registrador ao término da execução. Se o programa é executado infinitamente, imprima a frase "`execucao infinita`" (sem aspas).

Exemplo de entrada	**Exemplo de saída**
6 1 add 10 2 gotoif 5 4 3 sub 20 4 mul 2 5 div 6 6 halt	3

Exemplo de entrada	**Exemplo de saída**
8 1 add 7 2 xor 2 3 gotoif 5 6 4 and 0 5 add 3 6 or 2 7 gotoif 4 2 8 halt	execucao infinita

Dica: Em C, C++ e Java, o operador `&` faz a operação de E bit-a-bit entre duas variáveis, com *todos* os bits das variáveis. O operador `|` faz a operação de OU bit-a-bit, e o operador `^` faz a operação de OU-exclusivo bit-a-bit, ambos também com todos os bits das variáveis. Em Pascal, os operadores `and`, `or` e `xor` fazem as mesmas operações, respectivamente.