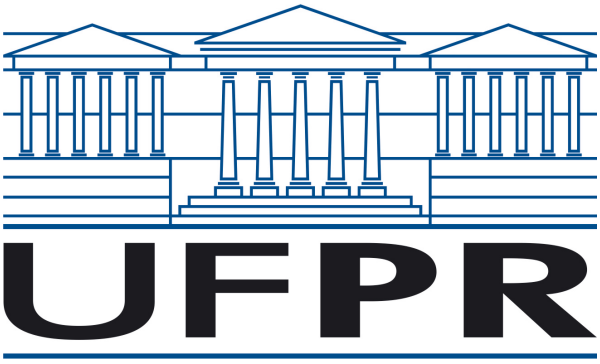
5^a Seletiva da UFPR

8 de Agosto de 2014

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



Problemas:
Bruno César Ribas
Cristhian Bonilha
Ricardo Oliveira
Rodolfo Rodovalho
Vinicius Ruoso

Revisão:
Marcos Alexandre Castilho
André Luiz Pires Guedes
Roberto A. Hexsel
Renato Carmo

Lembretes:

- É permitido consultar livros, anotações ou qualquer outro material impresso durante a prova.
- A correção é automatizada, portanto, siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa. Deve-se considerar entradas e saídas padrão.
- Procure resolver o problema de maneira eficiente. Se o tempo superar o limite pré-definido, a solução não é aceita. As soluções são testadas com outras entradas além das apresentadas como exemplo dos problemas.
- Teste seu programa antes de submetê-lo. A cada problema detectado (erro de compilação, erro em tempo de execução, solução incorreta, formatação imprecisa, tempo excedido ...), há penalização de 20 minutos. O tempo é critério de desempate entre duas ou mais equipes com a mesma quantidade de problemas resolvidos.

Página intencionalmente deixada em branco.

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 sua 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`
  - 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	Encontro Politécnico	1
B	Shrek	1
C	Vigilância	1
D	Penalties	1
E	Desafio das Moedas Prateadas	1
F	Escadas Rolantes	1
G	Senha da Tia	1
H	Fliperama	1
I	Remoção	1
J	Seleção	2

- Os juizes usam um sistema de 64 bits (idêntico às máquinas no DINF).
- Todas as linhas, tanto na entrada quanto na saída, terminam com o caracter 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).

Página intencionalmente deixada em branco.

Problema A: Encontro Politécnico

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

Dois professores (PA e PB)[1] combinaram de encontrar-se no Centro Politécnico (CP) às 15h00 numa quarta-feira. Como os professores são grandes estudiosos de movimentos retilíneos uniformes, pensaram em aplicar um de seus estudos mais recentes. Os dois professores dividiram o CP em um grande quadriculado. A cada passo, cada professor escolhe uma direção (Norte, Sul, Leste ou Oeste) e anda até o quadrado imediatamente vizinho na direção escolhida.

O quadriculado do CP é um plano cartesiano com origem em $(1, 1)$ como sendo o limite inferior esquerdo e as direções Norte e Sul andam no eixo Y, e as direções Leste e Oeste no eixo X. As direções Norte e Leste incrementam a posição de seus respectivos eixos e Sul e Oeste fazem o oposto.

Dada uma sequência de passos, você deve dizer se os professores se encontraram em algum momento, i.e, se eles ficaram no mesmo quadrado ao mesmo tempo, se algum professor saiu do CP ou se eles não se encontraram.

Entrada

A primeira linha da entrada contém dois inteiros N e M que indicam respectivamente o número de colunas e o número de linhas do CP ($1 \leq N, M \leq 10^5$). A segunda linha contém um inteiro P ($0 \leq P \leq 1000$) que indica quantos movimentos cada professor fez. Depois são apresentadas P linhas contendo dois números inteiros A e B, indicando a direção tomada pelos professores PA e PB, respectivamente, naquele passo. Os inteiros A e B podem assumir os seguintes valores: 1 (Norte), 2 (Sul), 3 (Leste) e 4 (Oeste). O professor PA inicia seu trajeto sempre na posição $(1, 1)$ e o professor PB na posição (N, M).

Saída

Seu programa deve imprimir:

- Caso os professores tenham se encontrado: as coordenadas do encontro e o passo em que ocorreu.
- Caso o(s) professor(es) tenha(m) saído do CP: as coordenadas em que saíram e o passo em que ocorreu. Se ambos professores saíram no mesmo passo imprima apenas a informação sobre o professor PA.
- Caso nenhuma das anteriores ocorra, imprimir: "Nao se encontraram".

Verifique os exemplos para entender melhor o formato da saída.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
5 5 4 3 2 3 2 3 2 3 4	Nao se encontraram

[1] A saber: PA é o prof. Castellone, e PB é o prof. Anthov, ambos do DINF

Exemplo de entrada	Saída para o exemplo de entrada
5 5 4 3 2 3 2 3 2 3 2	Encontraram-se na posicao (5,1) no passo 4

Exemplo de entrada	Saída para o exemplo de entrada
5 5 4 1 2 1 2 4 1 1 4	PA saiu na posicao (0,3) no passo 3

Problema B: Shrek

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

Na animação *Shrek (2001)*, o ogro Shrek e seu companheiro Burro partem em uma grande aventura para resgatar a princesa Fiona das guarras da Dragoa. Em uma das cenas da animação, Shrek atravessa um espaço para formação de fila contendo apenas um cidadão. O cidadão foge de Shrek, mas respeita o trajeto da fila. Shrek, por sua vez, a ignora, o que dá o tom cômico à cena.

Considere o chão como um plano cartesiano, e Shrek e o cidadão como pontos neste plano. O trajeto da fila pode ser descrita por N pontos (x_1, y_1), (x_2, y_2), ..., (x_N, y_N). O trajeto inicia no ponto (x_1, y_1), segue em linha reta até o ponto (x_2, y_2), de onde segue em linha reta até (x_3, y_3) e assim por diante, até acabar no ponto (x_N, y_N).

Inicialmente, Shrek está no ponto (x_s, y_s) e o cidadão está no começo do trajeto da fila (isto é, no ponto (x_1, y_1)). Shrek se movimenta a uma velocidade constante v_s m/s, e o cidadão se movimenta a uma velocidade constante v_c m/s, com $v_c \leq v_s$. Shrek pode se movimentar livremente no plano, enquanto o cidadão irá seguir rigorosamente o trajeto da fila até o seu final. Ao completar o trajeto, o cidadão sairá da fila.

Shrek deseja alcançar o cidadão antes dele sair da fila. Se é possível que Shrek alcance o cidadão antes dele sair da fila, quanto tempo, no mínimo, Shrek deve levar para alcançá-lo?

Considere que Shrek nunca tem sua velocidade reduzida enquanto se movimenta, inclusive quando atravessa o trajeto da fila. Considere também que pode ser possível que o cidadão seja alcançado exatamente no término da fila (isto é, no ponto (x_N, y_N)).

Entrada

A entrada inicia com uma linha contendo um inteiro N ($2 \leq N \leq 5 \times 10^4$) indicando o número de pontos no trajeto da fila. A próxima linha contém dois inteiros x_s e y_s ($0 \leq x_s, y_s \leq 10^3$) indicando a posição inicial de Shrek.

As próximas N linhas descrevem o trajeto. Cada linha contém dois inteiros x_i e y_i ($0 \leq x_i, y_i \leq 10^3$) descrevendo um ponto do trajeto. Os pontos são dados na ordem (x_1, y_1), ..., (x_N, y_N). É garantido que $(x_i, y_i) \neq (x_{i+1}, y_{i+1})$ para $1 \leq i < N$. As coordenadas são dadas em metros.

A última linha contém dois inteiros v_s e v_c ($1 \leq v_c \leq v_s \leq 10^3$) indicando a velocidade de Shrek e do cidadão, respectivamente, em metros por segundo.

Saída

Imprima uma linha contendo um número real t indicando que Shrek levará, no mínimo, t segundos para alcançar o cidadão. Arredonde e imprima o valor de t com exatamente duas casas decimais. Se não é possível alcançá-lo, imprima `impossivel`.

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
6 4 0 3 7 5 7 5 8 3 8 3 9 5 9 2 1	4.00

Exemplo de entrada	Saída para o exemplo de entrada
6 4 2 3 7 5 7 5 8 3 8 3 9 5 9 2 1	3.04

Exemplo de entrada	Saída para o exemplo de entrada
2 1 0 4 0 6 0 2 1	impossivel

Exemplo de entrada	Saída para o exemplo de entrada
3 0 0 4 3 2 7 1 2 1 1	5.88

Problema C: Vigilância

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

A rua que liga a entrada do campus ao prédio do Departamento de Informática (DINF) é uma das mais utilizadas no centro politécnico. Por isso, a Prefeitura da Cidade Universitária (PCU) investiu em um esquema de vigilância para garantir a segurança na rua.

A rua é composta por N segmentos contínuos numerados sequencialmente de 1 a N (o segmento 1 é o mais próximo à entrada, e o segmento N é o mais próximo ao DINF).

A prefeitura comprou C câmeras, numeradas de 1 a C. A câmera i ($1 \leq i \leq C$), quando ligada, filma todos os segmentos da rua entre a_i e b_i, inclusive ($1 \leq a_i \leq b_i \leq N$).

O consumo de energia de cada câmera é dado de maneira peculiar. Um vetor (w_1, w_2, ..., w_M) é fornecido pelo fabricante das câmeras. A câmera i, quando ligada, consome $\sum_{j=c_i}^{d_i} w_j$ unidades de energia, onde c_i e d_i ($1 \leq c_i \leq d_i \leq M$) são inteiros associados a cada câmera.

Sua tarefa é determinar quais câmeras devem estar simultaneamente ligadas para que todos os segmentos da rua sejam filmados, e para que o total de energia gasta pelas câmeras seja mínima.

Entrada

A entrada inicia com uma linha contendo três inteiros, N, M e C ($1 \leq N \leq 10^3, 1 \leq C \leq 5 \times 10^3$, $1 \leq M \leq 10^6$). A segunda linha contém M inteiros w_1, w_2, ..., w_M ($1 \leq w_i \leq 10^3$ para $1 \leq i \leq M$). As próximas C linhas contém a descrição das câmeras. Cada linha contém quatro inteiros a_i, b_i, c_i e d_i ($1 \leq a_i \leq b_i \leq N$, $1 \leq c_i \leq d_i \leq M$).

Saída

Se não é possível filmar toda a rua simultaneamente, imprima `impossivel`. Caso contrário, imprima a menor quantidade de energia necessária para filmá-la.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
5 5 4 1 3 5 7 9 1 3 1 5 2 4 2 4 3 5 1 3 2 5 2 5	34

Exemplo de entrada	**Saída para o exemplo de entrada**
5 4 2 8 3 1 5 1 3 4 4 5 5 2 3	impossivel

Página intencionalmente deixada em branco.

Problema D: Penalties

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

Ocorreu um torneio de futebol na sua cidade, e neste torneio, quando dois times empatavam, a disputa era resolvida nos pênaltis. Como o juiz não era muito experiente, ele decidia na hora quantas cobranças cada time faria, e após essas cobranças o time que fizesse mais gols seria o campeão.

Cada time realiza N cobranças no total. A primeira cobrança é realizada pelo time A; a segunda pelo time B; a terceira pelo time A; e assim por diante, até termos $2N$ cobranças no total.

Após observar uma sequência de cobranças de pênaltis, você logo percebeu que em alguns casos, após algumas cobranças, era possível dizer com certeza qual time seria o campeão, não importando a performance de ambos os times nas cobranças seguintes.

Por exemplo, se $N = 3$, e após a quarta cobrança o resultado da disputa estivesse 2 a 0, a vitória para o time A já estaria certa, pois não há como o time B empatar ou ganhar na sua última cobrança.

Sua tarefa é, dado o resultado de cada cobrança, descobrir após qual cobrança é possível saber quem será o campeão, ou relatar que a disputa terminou em empate.

Entrada

A entrada inicia com um inteiro N ($1 \leq N \leq 100$), o número de cobranças que cada time fará.

Em seguida há duas linhas, cada uma contendo N caracteres cada, onde a primeira linha representa as cobranças realizadas pelo time A, e a segunda pelo time B. O primeiro caractere se refere a primeira cobrança, o segundo a segunda cobrança, e assim por diante. Cada cobrança pode resultar em gol (caractere `o`) ou erro (caractere `x`).

Saída

Imprima uma linha contendo um único inteiro representando a cobrança após a qual é possível saber quem era o campeão, ou a palavra `Empate`, caso a disputa termine em empate.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
3 ooo xxx	4

Exemplo de entrada	**Saída para o exemplo de entrada**
3 ooo ooo	Empate

Exemplo de entrada	**Saída para o exemplo de entrada**
5 oxoox xoooo	10

Página intencionalmente deixada em branco.

Problema E: Desafio das Moedas Prateadas

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

O *Desafio das Moedas Prateadas* é um esporte individual popular no reino de Diddykongolândia. O campo e as regras do jogo são descritos a seguir.

O campo do jogo consiste em *locais* e *trechos*. Há N locais no campo. Um desses locais é o local de *largada*, no qual o jogador inicia o jogo.

Há M trechos no campo. Cada trecho é uma via unidirecional que liga um local do campo a outro local distinto. Os trechos são os únicos meios pelos quais o jogador pode se mover entre os locais do campo. Os trechos do campo são dados de tal forma que:

- é possível ir do local de largada a qualquer outro local usando os trechos;
- é possível ir de qualquer local do campo ao local de largada usando os trechos;
- é impossível sair de um local l e voltar para o mesmo local l sem passar pelo local de largada.

Há também K *moedas prateadas* no jogo. Cada moeda está em um local distinto do campo. Se o jogador chegar a um local que contém uma moeda, o jogador pode coletá-la. Não há moeda no local de largada.

O jogador começa o jogo no local de largada. Uma *volta* é completada pelo jogador quando ele retorna ao local de largada após passar por outro(s) local(is).

O jogador deve completar exatamente *três* voltas. Se o jogador conseguir coletar todas as moedas de prata antes de completar a última volta, ele vence. Caso contrário, ele perde.

Após analizar o campo de jogo, você descobriu quanto tempo leva para atravessar cada trecho. Agora, você deve descobrir se é possível vencer no campo dado e, em caso positivo, qual o tempo mínimo que você deve levar para vencer. Considere instantâneo o tempo para coletar uma moeda e para atravessar um local (sair de um trecho e entrar em outro adjacente).

Entrada

A entrada inicia com uma linha contendo três inteiros N, M e K ($2 \leq N \leq 1000$, $2 \leq M \leq (N^2 + N - 2)/2$, $1 \leq K \leq min\{12, N - 1\}$), indicando o número de locais, de trechos e de moedas no campo. Os locais são numerados de 1 a N. O local 1 é o local de largada.

As próximas M linhas descrevem os trechos. Cada trecho é descrito por três inteiros l_a, l_b e t ($1 \leq l_a, l_b \leq N$, $l_a \neq l_b$, $1 \leq t \leq 10^4$), indicando que há um trecho que leva do local a para o local b que é atravessado em t segundos.

A última linha contém K inteiros distintos k_i ($2 \leq k_i \leq N$ para $1 \leq i \leq K$) indicando os locais das moedas prateadas.

Saída

Se não é possível vencer o jogo, imprima uma linha contendo `impossivel`. Caso contrário, imprima uma linha contendo o menor tempo possível, em segundos, para vencer o jogo.

Exemplos

Exemplo de entrada	Saída para o exemplo de entrada
4 6 3 1 2 1 1 3 1 1 4 1 2 1 1 3 1 1 4 1 1 2 3 4	6

Exemplo de entrada	Saída para o exemplo de entrada
5 7 2 1 2 1 1 3 2 2 3 3 3 4 7 3 5 3 4 5 2 5 1 4 2 4	35

Exemplo de entrada	Saída para o exemplo de entrada
5 8 4 1 2 3 1 3 2 1 4 10 1 5 7 2 1 5 3 1 3 4 1 1 5 1 12 4 5 3 2	impossivel

Problema F: Escadas Rolantes

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

O shopping que fica na frente do campus conta com duas escadas rolantes que ligam seu piso térreo ao seu piso superior. Curiosamente, as escadas não têm um comportamento fixo: às vezes, a escada da esquerda sobe e a da direita desce; nas outras vezes, a da esquerda desce e da direita sobe.

Após hackear o sistema do shopping, você descobriu que o comportamento das escadas é definido por um gerador de números pseudoaleatórios `sjdarand(s,d)` que é executado todos os dias, antes do shopping abrir. A função `sjdarand(s,d)` recebe como entrada uma *semente* s e um *parâmetro* d (dois números inteiros positivos) e retorna 0 ou 1. A escada da esquerda sobe (e da direita desce) se o gerador retornar 0, e a da esquerda desce (e a da direita sobe) caso contrário. O pseudocódigo da função `sjdarand(s,d)` é:

```
r = 0
enquanto s > 0 faça
    se s é par então
        r = 1 - r
    fim-se
    s = quociente da divisão de s por d
fim-enquanto
retorna r
```

Sua tarefa é, dados a semente e o parâmetro, definir a configuração das escadas.

Entrada

A primeira linha da entrada contém a semente s ($1 \leq s < 10^{1000}$).
A segunda linha da entrada contém o parâmetro d ($2 \leq d \leq 32$).

Saída

Imprima uma linha contendo `sobe desce` se a escada da esquerda irá subir e a direita descer, ou `desce sobe` caso contrário.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
8 2	`desce sobe`

Exemplo de entrada	**Saída para o exemplo de entrada**
14 3	`sobe desce`

Página intencionalmente deixada em branco.

Problema G: Senha da Tia

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

Seu maior pesadelo está acontecendo: sua tia quer se cadastrar na rede social que você acessa. Comentários constrangedores, compartilhamentos desnecessários e fotos da sua infância farão parte do seu dia-a-dia, não há saída.

Ela está se cadastrando, mas por motivos de segurança há algumas regras em relação à senha a ser escolhida por novos usuários. É necessário que a senha tenha no mínimo N caracteres, entre eles no mínimo M devem ser letras minúsculas, A devem ser letras maiúsculas, e K devem ser números.

Sua tia está confusa, e pediu sua ajuda. Dada a senha que ela escolheu, diga se ela será aceita pelo site ou não.

Entrada

A entrada inicia com 4 inteiros N, M, A e K ($6 \leq N \leq 1000, 0 \leq M, A, K \leq 1000, M+A+K \leq N$), conforme descrito no enunciado.

Em seguida há uma sequência de caracteres indicando a senha, com no mínimo 1 e no máximo 1000 caracteres, sendo estes letras e números, apenas.

Saída

Imprima uma linha contendo `Ok =/`, se a senha atende todos os requisitos citados, ou `Ufa :)` caso contrário.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
`6 1 1 1` `beterraba`	`Ufa :)`

Exemplo de entrada	**Saída para o exemplo de entrada**
`6 3 1 6` `20Maio1994`	`Ok =/`

Exemplo de entrada	**Saída para o exemplo de entrada**
`10 3 4 0` `TeenageMutantNinja`	`Ufa :)`

Exemplo de entrada	**Saída para o exemplo de entrada**
`8 4 2 2` `123456`	`Ufa :)`

Página intencionalmente deixada em branco.

Problema H: Fliperama

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

No bairro onde você mora uma nova loja de jogos abriu. Esta não traz jogos da última geração, mas sim jogos antigos e clássicos. Tal loja fez muito sucesso, principalmente com os fliperamas. Para atrair ainda mais clientes, essa loja resolveu organizar uma competição: o jogador com maior número de pontos em um determinado jogo ganhará um prêmio misterioso.

Isso de fato atraiu muitos jogadores para a loja, mas também levantou um problema: para registrar o número de pontos feitos, o jogador deve inserir seu nome ao final do jogo, e o jogo tem um limite de três caracteres para registrar cada nome.

De acordo com as regras da competição, o nome a ser registrado deve ser escolhido da seguinte maneira: escolha três caracteres de posições distintas do seu nome original, e digite-os da esquerda para a direita, em caracteres minúsculos.

Por exemplo, se o jogador `Pedro` fosse registrar seu nome, as seguintes sequências são válidas: `ped`, `pdr`, `pdo`, `dro`, entre outras. Ainda para o jogador `Pedro`, as seguintes sequências são inválidas: `edp`, `dre`, `pda`, `ord`, entre outras.

O organizador da competição pediu então sua ajuda. Dados os nomes de todos os jogadores que vão participar, é possível que todos consigam registrar seus nomes, de modo que não haja mais de um jogador com o mesmo nome registrado?

Entrada

A entrada inicia com um inteiro $N(1 \leq N \leq 1000)$, indicando o número de nomes diferentes que vão participar da competição.

Em seguida há N linhas. Cada linha contém um inteiro M e uma sequência de caracteres S $(1 \leq M \leq 10)$, indicando que há M jogadores com o nome S na competição. Cada uma das N linhas conterá um nome distinto, contendo entre 3 e 5 caracteres, apenas com letras minúsculas.

Saída

Imprima uma linha contendo a palavra `Possivel`, caso seja possível realizar o que foi pedido no enunciado, ou `Impossivel` caso contrário.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
`1` `1 ana`	`Possivel`

Exemplo de entrada	**Saída para o exemplo de entrada**
`1` `2 ana`	`Impossivel`

Exemplo de entrada	Saída para o exemplo de entrada
3 3 pedro 4 marco 2 carlo	Possivel

Exemplo de entrada	Saída para o exemplo de entrada
2 1 ana 10 tania	Impossivel

Problema I: Remoção

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

Professor Breno é um jovem e ambicioso professor de Algoritmos. Ao longo do tempo em que ensina na disciplina, percebeu erros recorrentes entre os alunos em diversos semestres e, por isso, resolveu criar ferramentas para auxiliar o acompanhamento dos alunos. Afinal, a cada semestre que passa, a quantidade de alunos aumenta e fica cada vez mais difícil de fazer um acompanhamento "mais pessoal".

O professor Breno é muito ocupado e não está conseguindo tempo para implementar uma ferramenta e por isso pediu a sua ajuda.

A ferramenta que Breno deseja implementar é um identificador de vazamento de memória durante a remoção de elementos de uma lista duplamente encadeada. No momento atual, o professor pede que os elementos removidos da lista duplamente encadeada sejam liberados da memória. Por enquanto, Breno deseja apenas que seja implementada a ferramenta que obtém o *dump* de memória atual e mostra como a lista ficará após a remoção de alguns elementos, permitindo ao aluno comparar com a maneira que o seu programa se comportou.

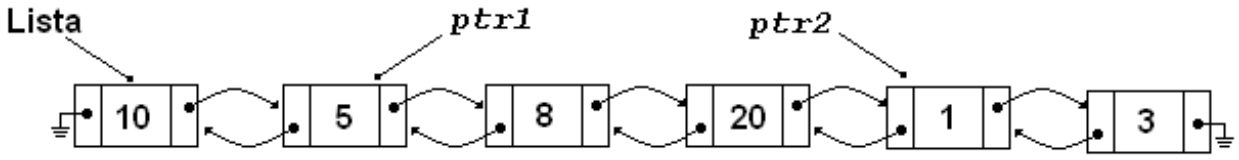


Figura 1: Exemplo de uma lista duplamente encadeada

Para o exercício que os alunos devem implementar, o professor pediu o método da remoção que elimina da lista duplamente encadeada um conjunto de nós. São dados dois ponteiros `PTR1` e `PTR2` pertencentes a uma lista duplamente encadeada. É garantido que:

- Estes ponteiros existem e não são valores nulos ($NULL$);
- Estes ponteiros podem estar em qualquer posição da lista;
- Há um caminho entre `PTR1` e `PTR2` na lista;
- `PTR1` vem "antes" de `PTR2` na lista;
- Pode existir qualquer quantidade de nós antes de `PTR1`, depois de `PTR2` e entre `PTR1` e `PTR2` na lista.

Todos os nós entre `PTR1` e `PTR2`, inclusive `PTR1` e `PTR2`, devem ser removidos da lista duplamente encadeada, e a memória que usam deve ser liberada.

Como praxe, os alunos confundem os ponteiros e perdem referências, atualizam os ponteiros do nó anterior e próximo de forma incorreta, causando o caos.

Para permitir que os alunos consigam identificar onde estão errando, e até confirmar se estão corretos, a ferramenta que você desenvolver receberá um *dump* da memória do programa do aluno contendo os nós apontados por `PTR1` e `PTR2` e outros nós, que podem ou não fazer parte da lista duplamente encadeada destes ponteiros, e determinar como a lista deverá ficar após a remoção, bem como quais elementos serão liberados da memória.

Entrada

A entrada contém o *dump* de memória do programa de um aluno. A entrada possui diversas linhas, e cada linha descreve um nó. Cada linha possui 3 números inteiros em formato hexadecimal representando, respectivamente, o endereço do nó, o endereço do nó anterior e do nó do próximo elemento. O endereço 0 representa $NULL$. As duas primeiras linhas do caso de teste representam os nós apontados por `PTR1` e `PTR2`, nesta ordem.

Nem todos os ponteiros fornecidos na entrada precisam, necessariamente, fazer parte da lista duplamente encadeada. Além disso, o *dump* pode não conter a lista completa, ou seja, o nó mais anterior a `PTR1` não é, necessariamente, o primeiro elemento da lista (com o endereço anterior apontando para $NULL$), bem como o nó mais posterior a `PTR2` não é, necessariamente, o último nó da lista (com o endereço de próximo apontando para $NULL$). Entretanto, é garantido que existe pelo menos 1 nó anterior a `PTR1` e 1 nó posterior a `PTR2`.

O arquivo de entrada não contém mais de 250000 linhas.

Para representar os ponteiros, utilize inteiro sem sinal de 64bits.

Saída

A saída deverá conter diversas linhas, contendo o estado final da lista duplamente encadeada, i.e, após a remoção dos nós entre `PTR1` e `PTR2`. Os nós deverão ser ser impressos na ordem da lista encadeada: o primeiro nó impresso deverá ser o nó fornecido mais anterior de `PTR1` na lista, e o último nó impresso deverá ser o nó fornecido mais posterior a `PTR2` na lista.

Após a impressão da lista, uma linha deverá ser pulada e devem ser impressos os endereços de todos os nós removidos, na ordem em que apareciam na lista encadeada originalmente.

Para entender melhor a saída, verifique os exemplos abaixo.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
a 9 b c b d 1 0 2 2 1 3 3 2 4 4 3 5 5 4 6 6 5 7 7 6 8 8 7 9 9 8 a b a c d c e	1 0 2 2 1 3 3 2 4 4 3 5 5 4 6 6 5 7 7 6 8 8 7 9 9 8 d d 9 e a b c

Exemplo de entrada	Saída para o exemplo de entrada
a 9 b c b d 2 1 3 fd fc fb d c e ff fb c e d 10 10 e 20 1 0 2 fe fa fc 7 6 8 fb fd ff 8 7 9 fc fe fd 9 8 a b a fa fa b fe	7 6 8 8 7 9 9 8 d d 9 e e d 10 10 e 20 a b fa fe fc fd fb ff c

Página intencionalmente deixada em branco.

Problema J: Seleção

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

A Seleção Brasileira passou por maus momentos durante a Copa do Mundo disputada em seu país. A preocupação entre a comissão técnica foi tão grande que agora estão investindo em análise de dados para verificar como é possível melhorar o time.

A equipe de estatísticos da seleção descobriu que o minuto em que os gols ocorrem nas partidas é uma variável importante. Além disso, um número ainda mais importante para eles é a mediana destes valores. Para realizar uma análise de todos os jogos da seleção, eles agregaram os gols de vários jogos em uma sequência. Por exemplo, se no jogo 1 ocorreu um gol aos 85 minutos (e não houve prorrogação) e no jogo 2 ocorreu um gol aos 5 minutos, os números de interesse são 85 e 95.

Os estatísticos não são muito bons com programação e pediram a sua ajuda para determinar os números que eles precisam. Para cada um dos N gols contabilizados, eles estão pedindo que você calcule a mediana dos minutos de todos os gols que foram contabilizados até ele. Em outras palavras, para todo gol i ($1 \leq i \leq N$), calcular a mediana dos minutos em que os gols $[g_1, g_2, \cdots, g_i]$ ocorreram. Para facilitar a interpretação dos dados, retorne a soma destas medianas módulo 1000000007 ($10^9 + 7$). Como eles não tiveram tempo para organizar os números, os gols podem não ter sido contabilizados na mesma ordem em que ocorreram.

Lembrando que a definição de mediana utilizada é a seguinte: a mediana de uma sequencia de n números é o valor na sequência, quando ordenada, da posição $(n+1)/2$ quando n é impar, ou o valor da posição $n/2$ quando n é par (considera-se a sequência $1-$indexada). Por exemplo, a mediana da sequência $[1, 2, 3, 4, 5]$ é 3, enquanto da sequência $[1, 2, 3, 4, 5, 6, 7, 8]$ é 4.

Entrada

A entrada inicia com uma linha contendo um inteiro N ($1 \leq N \leq 10^6$) indicando o número de gols que foram contabilizados. As próximas N linhas contém a descrição dos gols contabilizados. A linha i ($1 \leq i \leq N$) contém um inteiro g_i ($1 \leq g_i \leq 10^9$) descrevendo o instante de tempo em que o gol i ocorreu.

Saída

Imprima uma linha contendo o número de interesse dos estatísticos.

Exemplos

Exemplo de entrada	**Saída para o exemplo de entrada**
8 11 23 24 26 29 69 79 90	168