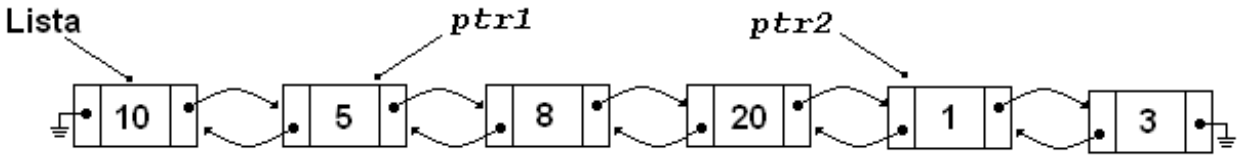
Remoção

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.



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 3números 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

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

Saída:

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

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

Saída:
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