Cache Simplificado

[1]**Autoria: Emanuel Catão**

Muito provavelmente você já deve ter ouvido ou utilizado cache em algum momento. O grande ponto a ser relembrado aqui é que o armazenamento de um cache não é infinito, o que nos leva a lançar mão de estratégias de invalidação para que registros novos possam entrar e outros possam sair.

Uma estratégia muito comum é a LRU (Least Recently Used). Nessa estratégia os itens que foram acessados menos recentemente são invalidados. Isso significa que se um item na cache não foi acessado há muito tempo, ele será removido do cache primeiramente.

Sua tarefa aqui é elaborar um programa em linguagem C, usando listas duplamente encadeadas, que recebe N linhas de entrada contendo um comando de operação bem como os parâmetros para que essa operação seja possível;

Temos nesse programa três tipos de operações:

1 - Cache: nessa operação é informado um inteiro que indica a capacidade do nosso cache;

2 - Get: nessa operação buscamos um item por seu inteiro chave e retornamos o valor inteiro correspondente à essa chave. Veja que toda vez que a operação de Get for chamada, é preciso ordenar a lista de forma que o valor buscado pelo Get ocupe a posição de valor mais recentemente acessado. Se o valor não for encontrado é preciso retornar "-1".

3 - Put: nessa operação inserimos um item, inteiro chave e valor inteiro correspondente, na cache, eliminando da cache o valor menos recentemente acessado. Uma operação bem sucedida retorna "200" como mensagem. Se um novo put for feito para um valor existente em chave na lista, o valor desse item deve ser atualizado com o valor passado no put. Exemplo: existe {4, 4} na lista, um put(4, 5), atualiza o valor desse item e agora ele passa a ser {4, 5}.

Para melhor elucidação do problema, veja mais abaixo a execução passo a passo, dos comandos do exemplo 1.

Entrada: Cada entrada é por N linhas separadas por quebra de linha ('\n'), até que "0" seja lido. Cada linha contém o comando a ser executado e os parâmetros a serem utilizados neste comando.

Saída: Uma lista com os resultados esperados após cada execução de comando com retorno, repassados na entrada. O formato é "['retorno1', 'retorno2']", vide exemplo. Após a lista, deve-se quebrar a linha ('\n').

[1] Adaptado de questão de Code Interview da Twitch

Exemplo 1, execução passo a passo:

```
Cache cache = new Cache(2);
put(cache, 1, 1);
```

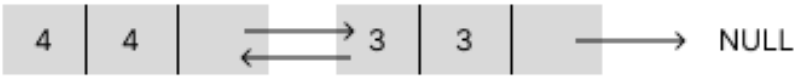

```
put(cache, 2, 2);
```


```
get(cache, 1);    // retorna 1 e altera ordem na cache
```


```
put(cache, 3, 3);
```

Aqui, como a cache já está funcionando em lotação máxima (tamanho 2), é preciso deletar o menos recentemente acessado, e adicionar o {3, 3} a ser inserido. Logo, perdemos o dado {2, 2} e agora temos a lista acima.

```
get(cache, 2);    // retorna -1, ja que {2, 2} foi perdido na operacao anterior
put(cache, 4, 4);
```



A cache já está funcionando em lotação máxima (tamanho 2),é preciso deletar o menos recentemente acessado, e adicionar o {4, 4} a ser inserido. Logo, perdemos o dado {1, 1} e agora temos a lista acima.

```
get(cache, 1);    // retorna -1, ja que {1, 1} foi perdido na operacao anterior
get(cache, 3);    // retorna 3 e reordena, se necessario
```


```
get(cache, 4);    // retorna 4 e reordena, se necessario
```

Exemplo de entrada 1:	Exemplo de saída 1:
Cache 2 put 1 1 put 2 2 get 1 put 3 3 get 2 put 4 4 get 1 get 3 get 4 0	[200, 200, 1, 200, -1, 200, -1, 3, 4]
Exemplo de entrada 2:	**Exemplo de saída 2:**
Cache 3 put 1 1 put 2 2 get 1 put 3 3 get 2 put 4 4 get 1 get 3 get 4 0	[200, 200, 1, 200, 2, 200, -1, 3, 4]
Exemplo de entrada 3:	**Exemplo de saída 3:**
Cache 3 put 1 2 put 2 1 get 1	[200, 200, 2, 200, 1, 200, 20, 1, 3, 5]

put 3 3 get 2 put 4 4 put 4 5 get 2 get 3 get 4 0	