Tabelas Hash – endereçamento aberto usando *containers*

Implemente uma tabela hash com **endereçamento aberto usando *containers*** em python 3.x conforme explicado na vídeo aula de Tabelas Hash a partir do instante de tempo de 1h18min0s: https://youtu.be/1sT2p4U5On8?t=4678

Para uma explicação geral sobre endereçamento aberto (obs: cuidado, esta explicação não usa *containers*), veja o mesmo vídeo a partir do instante 46min35s:
https://youtu.be/1sT2p4U5On8?t=2795

Se desejar, altere a qualidade dos vídeos para 720p para melhorar a qualidade da imagem.

Note que a implementação **deve usar *containers***, conforme explicado a partir de 1h18min0s.

No endereçamento aberto, todos os elementos estão armazenados na própria tabela hash (você deverá utilizar uma única lista de python para o armazenamento das chaves diretamente, sem usar orientação a objetos). A quantidade de *containers* será informada na entrada, assim como o tamanho de cada *container*. Você não precisará de posições extras ao final para o *overflow* (considere que os casos de teste nunca utilizarão o *overflow*).

A função de hash a ser utilizada para descobrir o *container* será: CHAVE % QTDE_CONTAINERS, onde CHAVE é o elemento a ser inserido e/ou buscado e QTDE_CONTAINERS é a quantidade de *containers* existente (dado fornecido pela entrada). Note que um container *c* começará na posição *c*TAM_CONTAINER*. **Dentro do container, será utilizada a sondagem linear**. Considere que não será utilizado o *overflow*.

> *Obs: se o exercício exigisse o uso do overflow, e algum container desejado estivesse cheio (isto é, após percorrer todas as TAM_CONTAINER posições dentro do container), deveria ser utilizado o overflow para a inserção da chave e/ou continuação da busca. O overflow também usa a sondagem linear. Note que o overflow só começa após todos os containers (posição inicial do overflow: QTDE_CONTAINERS*TAM_CONTAINER) e o seu tamanho interno é dado na entrada (TAM_OVERFLOW posições). Neste exercício, contudo, o overflow será desconsiderado.*

Entrada

O arquivo de entrada consiste de vários números inteiros em uma única linha:

- Os dois primeiros valores são: **QTDE_CONTAINERS TAM_CONTAINER** onde QTDE_CONTAINERS é a quantidade de *containers e* TAM_CONTAINER é o tamanho de cada *container.* Sua lista deverá possuir o tamanho total conforme segue:
  *n = QTDE_CONTAINERS * TAM_CONTAINER*;
- O terceiro valor da entrada é **QTDE_INSERCOES** que corresponde à quantidade de chaves a serem inseridas na tabela hash. **Em sequencia, a entrada conterá vários números inteiros** (cuja quantidade é **QTDE_INSERCOES**) **a serem inseridos na tabela hash** (use um *loop* para percorrer estes QTDE_INSERCOES elementos);
- Por fim, com os valores restantes da entrada, você deverá realizar consultas à tabela. Cada consulta será computada individualmente e terá um resultado na saída.

Todos os valores da entrada serão separados por espaços.

Saída

A saída deve informar, para cada consulta à tabela, qual é a **quantidade de comparações de chaves que foram realizadas na pesquisa da respectiva chave** (também deve retornar resultado se o elemento não for encontrado). Portanto, cada valor consultado deverá gerar um número inteiro como saída que é a

quantidade de comparações de chaves que foram realizadas para buscá-lo. Todas as saídas serão separadas por um espaço.

Exemplo:

Entrada – run.codes:
11 3 9 **10 21 31 32 9 4 3 13 14** 9 10 13 32 54 23 25

Entrada comentada para melhor visualização:
QTDE_CONTAINERS 11
TAM_CONTAINER 3
QTDE_INSERCOES 9
9 ELEMENTOS INSERIDOS: 10 21 31 32 9 4 3 13 14
ELEMENTOS QUE SOBRARAM PARA A BUSCA: 9 10 13 32 54 23 25

Saída:
2 1 1 3 3 1 3