MC202 — ESTRUTURAS DE DADOS

Laboratório 12 — Serviço da Alfândega

Problema

O Serviço de Alfândega do Brasil está encarregado do controle dos alimentos que entraram no país, por isso deve fazer uma revisão de todos os alimentos que serão trazidos, dado que a quantidade de alimentos que entra é muito grande e os recursos são limitados, a alfândega deve escolher aleatoriamente um grupo de caixas de lote, que será revisado, se alguma destas caixas for rejeitada o lote todo será rejeitado.

Para escolher as caixas que serão revisadas, é necessário criar um sistema que faça a escolha usando uma técnica específica. Alfândega decidiu tentar muitas técnicas diferentes, para o qual você foi atribuído a tarefa de escolher as caixas usando a mediana estatística.

Este método consiste em obter a caixa do meio de cada grupo de caixas à medida que são retiradas do lote e adicionadas ao grupo, de modo que cada vez que uma nova caixa é adicionada ao grupo, será necessário calcular a mediana e esta será selecionada para realizar a revisão. Coisas a considerar:

1. Uma caixa pode ser selecionada mais de uma vez para o conjunto de revisão.
2. No caso de que o número de caixas no grupo seja par, as duas caixas no meio devem ser selecionadas.

Entrada

Cada linha consiste em uma nova caixa retirada do lote, esta linha é composta por **<nome do produto>** e **<peso do produto>**, e a última linha é o símbolo **#**.

Saída

Para cada nova caixa retirada do lote, a mediana da lista previamente extraída deve ser obtida, de modo que cada linha da saída seja composta do nome ou nomes dos produtos que estão no meio da lista.

Exemplo 1

Entrada

```
mango 2
apple 3
#
```

Saída

```
mango: 2
mango: 2
apple: 3
```

Exemplo 2

Entrada

```
pear 3
orange 7
strawberry 1
blackberry 5
cucumber 8
avocado 2
apricot 6
#
```

Saída

```
pear: 3
pear: 3
orange: 7
pear: 3
pear: 3
blackberry: 5
blackberry: 5
pear: 3
blackberry: 5
blackberry: 5
```

Dicas:

- O método para obter a mediana deve ser menor que **O(n)**.
- Usar um **max-heap** e um **min-heap** é uma boa ideia
- Os pesos não podem ser repetidos no mesmo lote, os produtos podem.

Critérios específicos

- O uso da estrutura de dados do **heap** será obrigatório.
- Para as turmas E e F, este laboratório tem peso 3.
- Para as turmas G e H, este laboratório tem peso 2.
- Deverão ser submetidos os seguintes arquivos:
    - **heap.h**: Interface do heap.
    - **heap.c**: Implementação do heap.
    - **lab12.c**: Programa principal.
- Tempo máximo de execução: 1 segundos.

Observações gerais

No SuSy, haverá 3 tipos de tarefas com siglas diferentes para cada laboratório de programação. Todas possuirão os mesmos casos de teste. As siglas são:

1. **SANDBOX:** Esta tarefa serve para testar o programa no SuSy antes de submeter a versão final. Nessa tarefa, tanto o prazo quanto o número de submissões são ilimitados, porém arquivos submetidos aqui **não serão corrigidos**.
2. **ENTREGA:** Esta tarefa tem limite de **uma única submissão** e serve para entregar a versão final dentro do prazo estabelecido para o laboratório. Não use essa tarefa para testar o seu programa: submeta aqui quando não for mais fazer alterações no seu programa.
3. **FORAPRAZO:** Esta tarefa tem limite de **uma única submissão** e serve para entregar a versão final após o prazo estabelecido para o laboratório, mas com nota reduzida (conforme a ementa). O envio nesta tarefa irá substituir a nota obtida na tarefa ENTREGA apenas se o aluno tiver realizado as correções sugeridas no feedback ou caso não tenha enviado anteriormente em ENTREGA.

Observações sobre SuSy:

- Versão do GCC: C-ANSI 4.8.2 20140120 (Red Hat 4.8.2-15).
- Flags de compilação:
  `-ansi -Wall -pedantic-errors -Werror -g -lm`
- Utilize comentários do tipo `/* comentário */`
  comentários do tipo `//` serão tratados como erros pelo SuSy.

Além das observações acima, esse laboratório será avaliado pelos critérios gerais:

- Indentação de código e outras boas práticas, tais como:
    - uso de comentários (apenas quando forem relevantes);
    - código simples e fácil de entender;
    - sem duplicidade (partes que fazem a mesma coisa).
- Organização do código:
    - tipos de dados criados pelo usuário e funções bem definidas e tão independentes quanto possível.
- Corretude do programa:
    - programa correto e implementado conforme solicitado no enunciado;
    - inicialização de variáveis sempre que for necessário;
    - dentre outros critérios.