Teste 1 – Vetor de Ponteiros para Estruturas

Considere que um laboratório de exames clínicos identifica os seus clientes por números naturais (0, 1, 2, ...) e os organiza na forma de um vetor de ponteiros para uma **estrutura** que, para cada cliente, contém o número de exames realizados e um vetor de resultados de cada um destes exames. Ver Fig.1, onde as setas são ponteiros. Você deve montar este vetor a partir dos dados de um **arquivo texto** que tem o formato do exemplo da Fig. 2 (onde a primeira linha informa o número total de clientes precedido obrigatoriamente por "**`Clientes:`**" e a ordem dos clientes é qualquer). Todas as memórias alocadas devem ser **dinâmicas** (dos vetores e das estruturas).

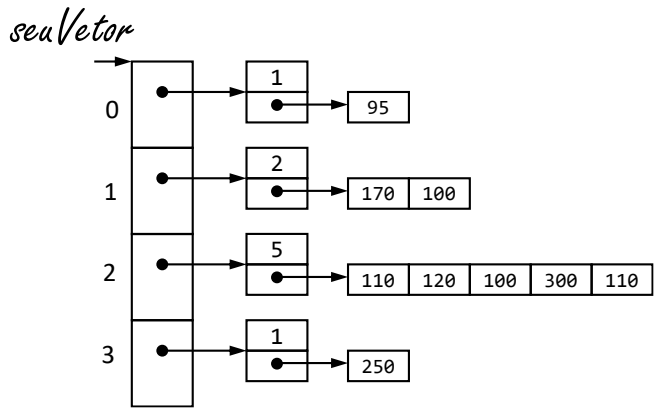

Fig.1

```
Clientes: 4
2 110 120 100 300 110
3 250
1 170 100
0 95
```

Fig. 2 Arquivo texto com dados

```
Cliente 0: 95
Cliente 1: 170 100
Cliente 2: 110 120 100 300 110
Cliente 3: 250

Clientes com o menor numero de exames:
Cliente 0: 95
Cliente 3: 250

Memoria Liberada!
```

Fig. 3 Exemplo de *output*

Na *main*, você deve ler os dados e montar o seu vetor principal usando uma **função auxiliar** que recebe, o vetor com os dados dos resultados dos exames do cliente em questão e o seu tamanho e retorna um ponteiro para a estrutura. As alocações de memória para as estruturas e para os vetores com os resultados de cada cliente devem ser do tamanho exato necessário e realizadas dentro desta função auxiliar, e.g. `seuVetor[i] = suaFunção(...);` (a função retorna NULL se houver problemas de memória). Considere que o número de exames nunca é maior do que um determinado valor (que você deve definir com **`#define`**) (use um valor compatível com os seus testes).

Durante a leitura dos dados identifique qual é o **menor número de exames realizados** por um cliente (no exemplo acima é 1 exame, o que ocorreu para os clientes 3 e 0). Não calcule este valor percorrendo o vetor montado (faça-o durante a leitura do arquivo texto). Também desenvolva e use três outras funções: uma para imprimir o vetor, uma para imprimir apenas os clientes que fizeram o menor número de exames e uma para liberar todas as memórias. Gerencie memória e interrupções (nestes casos basta *exit*() sem mensagens). Feche o arquivo.

Deposite um arquivo **.c** e um arquivo **.txt** com os dados utilizados. Nomeie os arquivos com o seguinte formato: *nome_ultimoSobrenome_matricula*_Teste01_G2. A falta do arquivo .txt anula o teste. Use obrigatoriamente dados completamente diferentes dos usados nos exemplos acima (número de clientes e resultados). Os dados NÃO podem estar ordenados por identificação de cliente. Considere que não há zero clientes e zero exames.

Dica: *fscanf* não lida com *end of line* (\n) (exceto para *strings*). Use a função *fgetc*(*stream*), que lê um caractere no *stream*, para ir testando se encontrou *newline* ou *end of file* (EOF) após cada leitura de um inteiro. Atenção especial para a última linha do arquivo.

DISCLAIMER:

Por favor coloque as seguintes linhas de comentário no início do seu código (contendo os seus dados), como sendo a sua identificação e a sua declaração:

```
/*****************************************************************
NOME COMPLETO:
MATRICULA PUC-Rio:
DATA:
DISCIPLINA: INF1007    TURMA (33A, 33B, 33C, 33D):
DECLARACAO DE AUTORIA:
   Declaro que este documento foi produzido em sua totalidade por mim,
   sem consultas a outros alunos, professores ou qualquer outra pessoa.
*****************************************************************/
```