Instituto Federal de Educação, Ciência e Tecnologia

Câmpus Campinas
ED1 – Estruturas de Dados I

Professores: José Américo (jose.americo@ifsp.edu.br)
Samuel Martins (samuel.martins@ifsp.edu.br)

Simulado Prova

Instruções

- Você deverá resolver o laboratório no arquivo **simulado.c**, que se encontra na página do laboratório no run.codes. Este arquivo já contém as definições (estruturas), algumas funções e o Programa Principal implementados por questões práticas;
- Você deverá usar tais implementações para a resolução do Laboratório, **NÃO PODENDO ALTERÁ-LAS;**
- A submissão deverá ser feita pelo sistema run.codes;
- Será considerada apenas a **última submissão** enviada ao sistema;
- A nota do lab será baseada nos acertos dos casos de teste e na correção das questões;
- A **NÃO** utilização ou alteração das implementações já definidas, bem como a criação de outras funções, CORRESPONDERÁ EM **NOTA ZERO**.
- Qualquer tentativa de fraude ou plágio também corresponderá em **NOTA ZERO** na prova.

1) [8.0] Para organizar as notas finais dos alunos de Estruturas de Dados 1, o prof. Dexter resolveu usar um **vetor dinâmico.** A nota de cada aluno é armazenada, exclusivamente, em um dado índice do vetor. Por exemplo, o índice [1] guarda a nota do aluno 1.

A fim de guardar a quantidade de notas, o professor propôs a seguinte struct:

```
typedef struct _vetor {
    int capacidade;
    int tamanho;
    float *data;
} Vetor;
```

onde ***capacidade*** é a capacidade máxima do vetor, ***tamanho*** é o tamanho atual do vetor (número de elementos inseridos) e ***data*** é o vetor de valores (notas).

O arquivo **simulado.c** já possui as definições (estruturas), algumas funções implementadas e o programa principal. Você deverá usar tal arquivo, NÃO ALTERANDO NADA JÁ IMPLEMENTADO. Seu objetivo é implementar apenas **APENAS** as seguintes funções:

a) [2.0] ***void adiciona_nota(Vetor *vet_notas, float nota);***
Adiciona uma nova nota no final do vetor. Caso o vetor esteja cheio, não faça nada.

b) [2.0] ***void adiciona_pontos_extras(Vetor *vet_notas, int indice_aluno, float pontos_extras);***
Adiciona um valor de pontos extras (**pontos_extras**) para o aluno de índice **indice_aluno**. Caso o índice passado seja menor que zero, os pontos extras são adicionados para todos os alunos. A nota final máxima é 10.0.

c) [2.0] ***float nota_media_turma(const Vetor *vet_notas);***
Retorna a nota média da turma.

d) [2.0] ***float desvio_padrao_turma(const Vetor *vet_notas);***
Retorna o desvio padrão das notas da turma.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}}$$

onde **N** é a quantidade de notas do vetor, x_i é a iésima nota e $\bar{x}$ é a nota média.

Entrada

A *primeira linha* da entra consiste de um **número inteiro** informando a capacidade do vetor de notas.

As próximas linhas consistem de uma sequência de comandos, sendo que o programa só termina com o comando ***para***

O Programa Principal (main), bem como as funções de impressão, já estão implementados corretamente. VOCÊ NÃO DEVERÁ ALTERAR TAIS FUNÇÕES.

Os comandos são:

- ***adiciona_nota*** **NOTA**
  - Adiciona a nota **NOTA** no final do vetor.
  - Caso o vetor esteja cheio, não faça nada.
  - Se a nota não estiver no intervalo [0, 10], não faça nada.
- ***adiciona_pontos_extras*** **INDICE_ALUNO PONTOS_EXTRAS**
  - Adiciona a quantidade de pontos extras (**PONTOS_EXTRAS**) para o aluno de índice **INDICE_ALUNO**.
  - Caso o índice passado seja menor que zero, os pontos extras são adicionados para todos os alunos.
  - Se o índice for acima da capacidade do vetor, não faça nada.
  - A nota final máxima é 10.0.

- ***imprime_notas***
  - Imprime as notas dos alunos.
- ***imprime_media_turma***
  - Computa e imprime a nota média da turma.
- ***imprime_desvio_padrao_turma***
  - Computa e imprime o desvio padrão das notas da turma.
- ***para***
  - Termina a execução do programa.

Saída

A saída consiste na impressão dos elementos da lista por meio das funções ***imprime_notas, imprime_media_turma*** e ***imprime_desvio_padrao_turma***. Essas funções já estão implementadas.

2) Questões teóricas

Responda as questões pelo link:
https://moodle.cmp.ifsp.edu.br/mod/quiz/view.php?id=38180

2.1) [1.0] Explique para que serve as funções *malloc* e *calloc* e como elas diferem entre si.

2.2) [1.0] Qual o problema do trecho de código abaixo?

```
float *v;

for (int i = 0; i < 10000000; i++) {
    v = (float *) calloc(50000 * sizeof(float));
}

free(v);
```