Instituto Federal de Educação, Ciência e Tecnologia

Câmpus Campinas
ED1 – Estrutura de Dados I

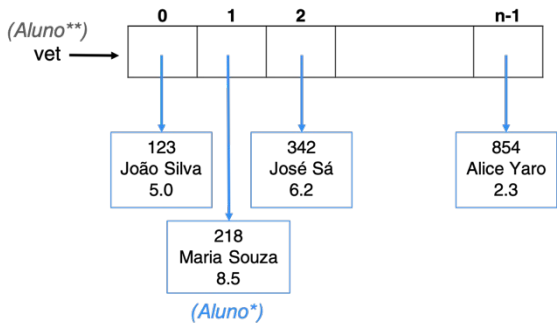
Professores: José Américo (jose.americo@ifsp.edu.br)
Samuel Martins (samuel.martins@ifsp.edu.br)

Avaliação P1

Instruções

- Você deverá resolver o laboratório no arquivo **prova1.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) [7.0] Para organizar as notas finais dos alunos de ED1, os profs. Sasá e Zezé resolveram utilizar um **vetor de alunos, alocado dinamicamente.** Cada aluno possui 3 informações: RA, equivalente ao prontuário (***único***), nome e nota final. A figura abaixo ilustra tal vetor:



onde **n** é o número de alunos da turma. A *struct* abaixo foi codificada par armazenar os dados de um aluno:

```c
typedef struct _aluno {
    int RA;
    char nome[64];
    float nota_final;
} Aluno;
```

O arquivo **prova1.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_pontos_extras(Aluno **vet, int n, int indice_aluno, float pontos_extras);***

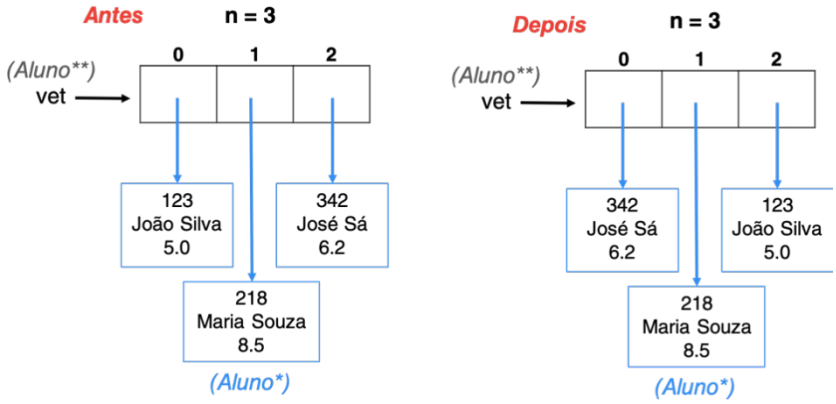
- Adiciona um valor de pontos extras (**pontos_extras**) para o aluno de índice **indice_aluno**;
- Assuma que o vetor possui exatamente **n alunos** inseridos;
- Se o índice for negativo, os pontos extras são adicionados para todos os alunos;
- Se o índice for inválido, não faça nada;
- Os pontos extras também **podem ser negativos;**
- A nota final mínima é 0.0 e a máxima é 10.0;

b) [2.0] ***Aluno *aluno_com_maior_nota(Aluno **vet, int n);***

- Retorna **uma cópia** do aluno com a maior nota;
- Em caso de empate, retorne o aluno com **menor RA**;
- Assuma que o vetor possui exatamente **n alunos** inseridos;

d) [3.0] ***void inverte_alunos(Aluno **vet, int n);***

- Inverte o vetor de alunos;
- Assuma que o vetor possui exatamente **n alunos** inseridos;



Entrada

A *primeira linha* da entra consiste de um **número inteiro n** informando o número de alunos do curso.
As próximas **n** linhas contém os dados dos alunos da turma (um aluno por linha).
As linhas subsequentes consistem de uma sequência de comandos, sendo que o programa só termina com o comando ***para***

Os comandos são:

- ***adiciona_pontos_extras*** **INDICE_ALUNO PONTOS_EXTRAS**
  - Adiciona a quantidade de pontos extras (**PONTOS_EXTRAS**) para o aluno de índice **INDICE_ALUNO**.
- ***inverte_alunos***
  - Inverte o vetor de alunos da turma.
- ***imprime_alunos***
  - Imprime todos os alunos da turma (armazenados no vetor).
- ***imprime_aluno_maior_nota***
  - Procura e imprime os dados do aluno com maior nota da turma.
- ***para***
  - Termina a execução do programa.

Saída

A saída consiste na impressão dos elementos da lista por meio das funções ***imprime_alunos*** e ***imprime_aluno_maior_nota***. Essas funções já estão implementadas.

2) Questões teóricas

Responda as questões pelo link:
https://moodle.cmp.ifsp.edu.br/mod/quiz/view.php?id=38665

2.1) [0.75] Considerando um vetor 'v' em linguagem C, indique a expressão que NÃO é verdadeira:

a) v == &v

b) *v == v[0]

c) &v==&v[0]

d) *(v+1) == &v[1]

2.2) [0.75] Com relação a matrizes alocadas dinamicamente em C, a seguinte afirmação NÃO é verdadeira:

a) Matrizes dinâmicas podem ser alocadas usando comandos ‘malloc’ ou ‘calloc’;

b) Matrizes alocadas dinamicamente nem sempre ocupam posições contíguas na memória Heap

c) Matrizes dinâmicas podem ser percorridas em qualquer ordem, sem perda de eficiência.

d) Alocações dinâmicas de uma matriz m são desfeitas usando comandos free para a parte da memória heap e m=NULL

2.3) [0.75] A função 'destroi_vetor_alunos' possui o comando:

*vet_ref = NULL;

Informe se o efeito desse comando ocorre na memória Stack ou memória Heap. Comente.

2.4) [0.75] Considere que a variável **mat** é uma matriz de inteiros, de tamanho N x N, alocada dinamicamente. Comente se há algum problema com o trecho de código abaixo.

```
// ...
// N = 30000
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        mat[j][i] += 10;
    }
}
// ...
```