



Avaliação Diagnóstica

Aluno:

Instruções Gerais

Leia atentamente as instruções antes de iniciar a avaliação:

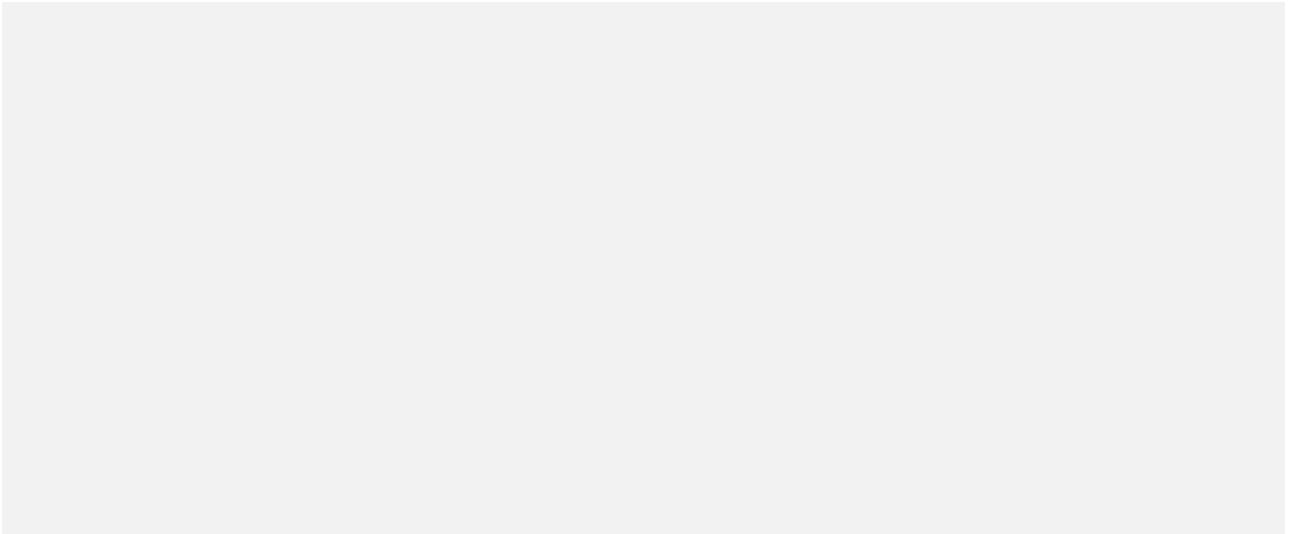
- 1) Não esqueça de colocar o seu nome completo nesta folha.
- 2) A fim de avaliar seu prévio conhecimento e habilidades adquiridas sobre programação em linguagem C, responda às questões propostas, tendo em conta que:
 - a. Esta avaliação visa identificar possíveis conteúdos que devam ser reforçados nesta disciplina;
 - b. Esta avaliação é **diagnóstica** e individual;
 - c. Dado o objetivo diagnóstico desta avaliação, quando for o caso, aponte as dificuldades (apontar os conceitos: funções, ponteiros, recursividade, etc.) que você encontrar.
- 3) Faça uso das boas práticas de programação/codificação e aplique adequadamente os conceitos de modularização que você já conheça.

Autoavaliação (Responder ao final e entregar esta folha ao professor)

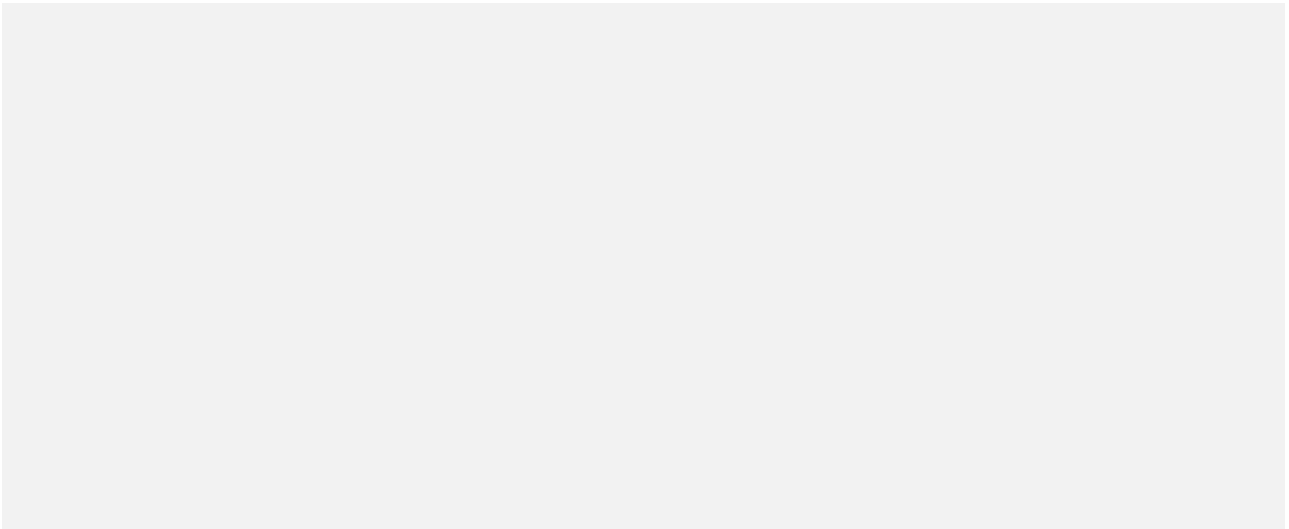
Já cursou IMD0029 Estruturas de Dados Básica I anteriormente? ___ SIM ___ NÃO ___ Aproveitou
--- Caso esteja cursando este semestre, indique a turma ou professor: _____

Indique as dificuldades encontradas **em cada questão** (01-03). Aponte os conceitos que você considera ter deficiência (ex: modularização, passagem de parâmetros, recursão, arquivos, registros, alocação dinâmica, ponteiros, etc).

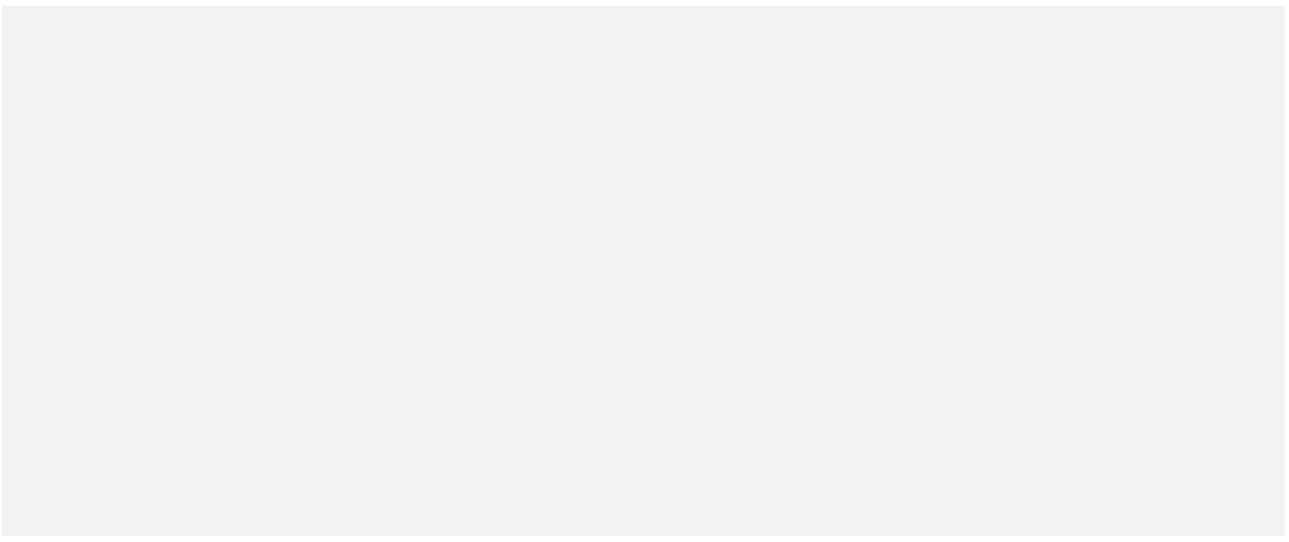
Questão 01.



Questão 02.



Questão 03.



Questão 01

Projete um registro (*struct*) denominado *Aluno* contendo dois campos, um para o nome do aluno e outro para a média do aluno em uma determinada disciplina. Em seguida, implemente um programa que: (1) Leia do arquivo *entrada.dat* um conjunto de alunos (dados por seus nomes e respectivas notas, separadas por ponto-e-virgula); (2) armazene internamente as informações fornecidas pelo usuário em um *array* alocado dinamicamente, e; (3) imprima a média aritmética das notas dos alunos cadastrados.

Exemplo de formato para o arquivo *entrada.dat*:

```
Joao dos Anzois;8.0;
Maria Joaquina;9.0;
```

Exemplo de saída para o programa com o cadastro de dois alunos e suas respectivas notas:

```
$ ./media
$ Lendo registros...
$ Lido: Joao dos Anzois - 8.0
$ Lido: Maria Joaquina - 9.0
$ Total de Registros: 2
$ Calculando...
$ Media das notas dos alunos: 8.5
```

Questão 02

Escreva um programa chamado *anterior* que lê um valor inteiro e retorna o maior número primo inteiro anterior ao valor do fatorial desse número. Você deverá obedecer aos seguintes detalhes de implementação:

- O valor do número inteiro deve ser lido através da linha de comando.
- Implemente o seu programa de forma modular. Crie o conjunto de arquivos *fatorial.h/c* (com a implementação da função de fatorial), *primalidade.h/c* (com a implementação das funções que testam a primalidade de um valor inteiro e que retorna o maior primo inteiro anterior a X) e um arquivo *main.c* (contendo o programa principal). Crie um *makefile* para a compilação e geração do binário/executável.
- Utilize conceitos de recursividade para o cálculo do fatorial e para a obtenção do maior número primo anterior ao valor do fatorial do número em questão.

Você pode utilizar os seguintes casos de teste:

Entrada	Fatorial	Saída
5	120	113
3	6	5
9	362880	362867

Um exemplo de execução do programa seria:

```
$ ./anterior 5
$ Maior numero primo anterior ao fatorial de 5 (120) eh 113.
```

Questão 03

Apresente a saída para o código abaixo, justificando os valores apresentados.

```
#include <stdio.h>

int somaA (int a, int b)
{
    ++a;
    b++;
    int result = a + b;
    return result;
}

int somaB (int* a, int b)
{
    int* x = malloc(sizeof(int));
    (*x) = (*a);
    (*x) *= 2;
    int result = (*x) + b;
    return result;
}

void somaC (int a, int b, int* result)
{
    a++;
    (*result) += a + b;
}

int main(int argc, char* argv[])
{
    int arg1 = 5;
    int arg2 = 6;

    printf("somaA (%d,%d) = %d\n", arg1, arg2, somaA(arg1,arg2));
    printf("somaB (%d,%d) = %d\n", arg1, arg2, somaB(&arg1,arg2));

    int resultado = 0;
    somaC(arg1,arg2,&resultado);
    printf("somaC (%d,%d,resultado) => resultado = %d\n", arg1, arg2, resultado);

    return 0;
}
```