**Universidade Tecnológica Federal do Paraná**

**Departamento acadêmico de eletrônica (DAELN)**

Curso: Engenharia Mecatrônica

Disciplina: Fundamentos de Programação – EL72E / Turmas S15 e S16

Professores: Edenilson José da Silva, Omero Francisco Bertol e Valfredo Pilla Júnior

Data: \_\_/\_\_/\_\_\_\_\_

**Prova 2**

NOTA

**Aluno:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Turma:\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **(Valor: 5,0)** (**Alocação Dinâmica de Memória, Arrays, Arquivos**)

Elaborar um programa com as seguintes características:

a) Ler os dados de um arquivo de texto denominado “numeros.txt” como o seguintes formato:

4 ---> quantidade de números de um conjunto

11 ---> primeiro número do conjunto

32

7

93 ---> quarto número do conjunto

b) Criar usando alocação de memória um vetor para armazenar o conjunto de números

c) Para cada elemento do conjunto gravar em um arquivo de texto denominado “divisores.txt” todos os divisores do número, destacando os números primos. Veja a saída esperada:

**11** possui os seguintes divisores:

1o. divisor = 1

2o. divisor = 11

<<< numero primo >>>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**32** possui os seguintes divisores:

1o. divisor = 1

2o. divisor = 2

3o. divisor = 4

4o. divisor = 8

5o. divisor = 16

6o. divisor = 32

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**7** possui os seguintes divisores:

1o. divisor = 1

2o. divisor = 7

<<< numero primo >>>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**93** possui os seguintes divisores:

1o. divisor = 1

2o. divisor = 3

3o. divisor = 31

4o. divisor = 93

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Conjunto de números avaliados: {11, 32, 7, 93}

1. (**Valor: 5,0**) (**Alocação Dinâmica de Memória, Arrays, Arquivos, Funções**)

Elaborar um programa que realiza a soma de matrizes quadradas de número inteiros. Para tanto, um arquivo ***dados.txt*** contém em sua primeira linha há a dimensão da matriz na forma de apenas um número (por exemplo 4, significa que as matrizes são 4x4). A partir da segunda do arquivo encontramos cada uma das matrizes que serão somadas. No arquivo, há uma linha vazia separando as matrizes.

O programa deverá realizar a operação solicitada e devolver o resultado na forma de um arquivo ***resultado.txt***, no qual a matriz resultante deverá ser armazenada de forma estruturada.

Para tanto, o programa deverá alocar dinamicamente memória para armazenar as matrizes operadas e a resultante da operação.

Função para somar as matrizes:

**void somarMatrizes(int \*\* matriz1, int \*\* matriz2, int \*\* soma\_matriz, int n)**;

onde *n* é o número de elementos das matrizes quadradas deve ser usada para operar a soma das matrizes.

**O programa não deve solicitar o nome dos arquivos ao usuário.**

**dados.txt:**

4 <-- dimensão das matrizes, neste caso, serão matrizes 4 x 4

1 1 1 1 <-- primeira matriz

2 2 2 2

3 3 3 3

4 4 4 4

<-- linha vazia separando as matrizes

1 1 1 1 <-- segunda matriz

1 1 1 1

1 1 1 1

1 1 1 1

**resultado.txt:**

2 2 2 2

3 3 3 3

4 4 4 4

5 5 5 5