Universidade Federal de São João del-Rei
Ciência da Computação
Laboratório de Programação II

Instruções para entrega do roteiro:

- Entregue o roteiro apenas como um arquivo compactado no formato *.ZIP* com o nome **Y_roteiroX.zip**, onde **X** é o **número do roteiro** e **Y** é o **número da sua matrícula**. Não serão aceitos outros formatos.
- Envie na raiz do arquivo compactado um arquivo .pdf com as respostas das perguntas sobre sua implementação e os arquivos .c, .h e o Makefile. Não envie executáveis ou coloque subpastas. Serão descontados pontos se vierem incompletos ou incorretos
- Inclua nome e matrícula, e mantenha a resolução dos exercícios ordenada e legível.
- Para perguntas teóricas, sua resposta não pode ser cópia do slide e deve estar escrita em português claro e conciso, apresentando também os cálculos realizados onde necessário.
- **DATA DA ENTREGA: 20/12/2024**

Roteiro 8

Matriz Linearizada

1. A matriz linearizada é uma ferramenta útil para tornar os dados de uma matriz contígua em memória. Utilizando a seguinte estrutura para o nosso TAD Matriz:

```
typedef struct {
     int l, c; //linhas e colunas
     int *data; //dados da matriz
} Matriz;
```

Temos que a posição `data[lin][col]` é data pela posição `data[index]`, onde `int index = lin * m->c + col`.

Com isso, implemente as seguintes funções para a matriz:

TAD Matriz	
`void criaMatriz(Matriz *m);`	Aloca uma nova matriz
`void apagaMatriz(Matriz *m);`	Desaloca a matriz
`void alteraElementoNaMatriz(Matriz *m, int lin, int col, int valor);`	Insere o novo valor na posição indicada
`int consultaElementoNaMatriz(Matriz *m, int lin, int col);`	Retorna o valor da matriz na posição indicada.

`void imprimeMatriz(Matriz *m);`	Imprime a matriz
`Matriz* getTransporta(Matriz *m);`	Retorna a transposta da matriz
`Matriz* somaMatrizes(Matriz *m1, Matriz *m2);`	Retorna a soma das matrizes

2. Implemente também uma função main que teste todas as funções acima da sua Matriz.

Matriz CSR

3. Implemente as mesmas funções acima, mas dessa vez como uma Matriz CSR (Compressed Sparse Rows) vista em sala de aula.

   Considere a seguinte estrutura para a sua implementação:

```
typedef struct {
     int l, c; //quantidade de linhas e colunas da matriz
     int *data; //dados da matriz
     int *jr; //índice das colunas dos elementos
     int *jc; //Índice do começo das linhas
     int Nz; //quantidade de elementos não nulos na matriz
} Matriz;
```

4. Teste seu programa com a mesma função main do exercício 2.