UNIVERSIDADE FEDERAL DO MARANHÃO
ESTRUTURAS DE DADOS I – DEPARTAMENTO DE INFORMÁTICA
LISTA DE EXERCÍCIOS – Revisão de Algoritmos de Matrizes e Vetores (Alocaçào dinâmica)

1. Fazer um algoritmo que recebe um número N, aloca um vetor de inteiros dinamicamente com n posições, lê os n números do teclado, coloca nas n posições do vetor, e:

a) Imprime o maior valor do vetor;
Protótipo: void ImprimeMaiorValor (int n);

b) Imprime a soma dos números;
Protótipo: void ImprimeSomaDosNumeros (int n);

c) Imprime o elemento da posição teto (n/2)
Protótipo: void ImprimeElementoDoMeio (int n);

2. Fazer um algoritmo que recebe dois números inteiros N e M, aloca dois vetores de inteiros dinamicamente com n e m posições respectivamente, lê os n números do teclado, coloca nas n posições do vetor, aloca um vetor e retorna o vetor C:
   a) C é o resultado da união do vetor A com o vetor B
      Protótipo: int *UniaoVetores(int *v, int n)

   b) C é o resultado da interseção do vetor A com o vetor B
      Protótipo: int *InterseçãoVetores(int *v, int n)

   c) C é o resultado da intercalação do vetor A com o vetor B;
      Protótipo: int *IntercalaçãoVetores(int *v, int n)

   a) C contém os elementos 1, 3, 5.. de A;
      Protótipo: int *ElemEmPosicaoImparNoVetor(int *v, int n)

   b) C contém os elementos 0, 2, 4.. de B.
      Protótipo: int *ElemEmPosicaoParNoVetor (int *v, int n)

3. Faça um algoritmo que recebe os inteiros n, m, p e q, e as matrizes ma e mb, armazenadas como vetor de vetores (vma e vmb) . O algoritmo retorna a matriz alocada como vetor de vetores mc com o resultado do produto entre as duas matrizes ma e mb se ele for possível, retornando NULL caso contrário.
   Protótipo: int **MultMatrixArmazenadaVetor (int n, int m, int p, int q, float **vma, float **vmb)

4. Faça um algoritmo que recebe os inteiros n, m, p e q, e as matrizes ma e mb, armazenadas como vetor de vetores (vma e vmb) . O algoritmo retorna o vetor vmc com a matriz resultado do produto entre as duas matrizes ma e mb se ele for possível, retornando NULL caso contrário.

Protótipo: int MultMatrixArmazenadaVetor (int n, int m, int p, int q, float *vma, float *vmb, float *vmc)

5. Faça um algoritmo que recebe um vetor de n números reais e uma matriz com m linhas e p colunas, de números reais armazenada no vetor vmat, e imprime em quais posições da matriz (linha e coluna) o valor é igual a um dos valores do vetor (imprime também a posição do vetor).
   Protótipo: int CmpVectorMatrix (int n, int m, int p, float *vet, float *vmat)

6. Faça um algoritmo que recebe os inteiros n, m, p e q, e as matrizes ma e mb, armazenadas nos vetores vma, vmb . O algoritmo retorna o vetor vmc com a matriz resultado do produto entre as duas matrizes ma e mb se ele for possível, retornando NULL caso contrário.
   Protótipo: int MultMatrixArmazenadaVetor (int n, int m, int p, int q, float *vma, float *vmb, float *vmc)

7. Faça algoritmo que recebe uma matriz A_{NxMxP}(volume) armazenado no vetor vola, os valores N, M e P e e retorna um vetor com:

a) Os elementos da linha I da fatia K.
Protótipo: int *DevolveVetorLinhaIFatiaK (int *VolA, int N, int M, int P, int I, int K);

b) Os elementos da coluna J da fatia K.
Protótipo: int *DevolverVetorColunaJFatiaK (int *VolA, int N, int M, int P, int j, int K);

c) Os elementos da diagonal principal da fatia K.
Protótipo: int *DevolverVetorDiagonalfatiaK (int *VolA, int N, int M, int P, int K);

d) Os elementos da linha I de todas as fatias.
Protótipo: int *DevolverVetorLinhaITodasFatias (int *VolA, int N, int M, int P, int I, int K);

e) Os elementos (I, J) de todas as fatias.
Protótipo: int *DevolverVetorElmIJTodasasFatias (int *VolA, int N, int M, int P, int I, int K);

8. Faça um algoritmo para receber a matriz A_{nxm}, armazenada como um vetor de vetores vma, os inteiros n e m, e retornar uma matriz armazenada como vetor de vetores com a transposta da Matriz A.
   Protótipo: int *transposta(int *vma, int n, int m)

9. Faça um algoritmo para receber a matriz A_{nxm}, armazxenada no vetor vma, os inteiros n e m, e retornar o vetor com a transposta da Matriz A.
   Protótipo: int *transposta(int *vma, int n, int m)

10. Faça um algoritmo para receber uma matriz armazenada em um vetor e retornar:
0 – Se for um Matriz Quadrada;

1 – Se for uma Matriz quadrada simétrica;
2 – Se for uma Matriz quadrada Matriz Diagonal;
3 – Se é uma Matriz quadrada Assimétrica;
4 - Se é uma Matriz quadrada Simétrica;

Protótipo: int TipodeMatriz(int *ma, int n, int m)

11. Faça um algoritmo para receber uma matriz armazenada em um vetore retornar:
-1 – Se não for uma Matriz quadrada;
0 – Se for um Matriz Quadrada;
Protótipo: int TipodeMatrizQuadrada(int *ma, int n, int m)

12. Faça um algoritmo para receber uma matriz armazenada em um vetore retornar:
-1– Se não for uma Matriz quadrada;
0 – Se for um Matriz Quadrada;
1 – Se for uma Matriz quadrada simétrica;
Protótipo: int TipodeMatrizSimétrica(int *ma, int n, int m)

13. Faça um algoritmo para receber uma matriz armazenada em um vetore retornar:
    -1– Se não for uma Matriz quadrada;
    0 – Se for um Matriz Quadrada;
    1 – Se for uma Matriz quadrada Matriz Diagonal;
    Protótipo: int TipodeMatrizDiagonal(int *ma, int n, int m)

14. Faça um algoritmo para receber uma matriz armazenada em um vetor e retornar:
    -1– Se não for uma Matriz quadrada;
    0 – Se for um Matriz Quadrada;
    1 – Se é uma Matriz quadrada Assimétrica;
    Protótipo: int TipodeMatrizAssimétrica(int *ma, int n, int m)

15. Faça um algoritmo para receber uma Matriz alocada como um vetor e retornar:
    -1– Se não for uma Matriz quadrada;
    0 – Se for um Matriz Quadrada;
    1 – Se é uma Matriz quadrada Simétrica;
    Protótipo: int TipodeMatrizSimétrica(int *ma, int n, int m)

16. Faça um algoritmo para receber uma Matriz armazenada como um vetor de vetores e retornar uma matriz armazenada como um vetor de vetores com o resultado da multiplicação de A por At (sua transposta) ou NULL caso a multiplicação não seja possível:
    a-usando memória adicional (sem copiar a matriz A para uma outra matriz com sua versão transposta);
    Protótipo: int **multiplicaPelaTranspostaA(int **ma, int n, int m)

    b-sem usar memória adicional (copiando a matriz A para uma outra matriz com sua versão transposta);;
    Protótipo: int **multiplicaPelaTranspostaB(int **ma, int n, int m)

17. Faça um algoritmo para receber uma Matriz armazenada como um vetor e retornar um vetor com o resultado da multiplicação de A por At (sua transposta) ou NULL caso a multiplicação não seja possível:
    a-usando memória adicional (sem copiar a matriz A para uma outra matriz com sua versão transposta);
    Protótipo: int *multiplicaPelaTranspostaA(int *ma, int n, int m)

    b-sem usar memória adicional (copiando a matriz A para uma outra matriz com sua versão transposta);;
    Protótipo: int *multiplicaPelaTranspostaB(int *ma, int n, int m)

18. Faça um algoritmo para receber uma Matriz armazenada como um vetor e retornar um vetor vet com os elementos da linha l da matriz A
    Protótipo: int * linhadaMatriz(int **ma, int n, int m, int l)

19. Faça um algoritmo para receber uma Matriz armazenada como um vetor e retornar um vetor com os elementos da coluna “p” da matriz
    Protótipo: int * colunadaMatriz(int **ma, int n, int m, int p)

20. Faça um algoritmo para receber uma Matriz armazenada como um vetor e retornar um vetor com os elementos da diagonal principal da matriz
    Protótipo: int * diagonaldaMatriz(int **ma, int n, int m)