Estrutura de Dados Avançada – Lista 00

Engenharia da Computação – 3º Período – 2018/2 – Prof.º Philippe Leal

ATENÇÃO: Todos os vetores e matrizes utilizados têm que ser declarados na função *main*.

1) Considere a matriz $A=[a_{ij}]_{n\times m}$, onde $n = 4$ e $m = 5$, com números inteiros positivos gerados aleatoriamente de 1 até 20. Faça um algoritmo para gerar a matriz *A* e verificar se ela satisfaz a seguinte condição:

$$\underset{0\le j\le m-1}{Min}\sum_{i=0}^{n-1} a_{ij} \;\le\; \underset{0\le i\le n-1}{Max}\prod_{j=0}^{m-1} a_{ij}$$

Crie um **procedimento** para gerar a matriz e uma **função** para realizar a verificação. De acordo com o retorno da função de verificação, deve-se imprimir na função *main*: "**Condicao Satisfeita**" ou "**Condicao Nao Satisfeita**".

2) Considere uma matriz *M* de ordem 3 de números inteiros armazenada no arquivo "**MatrizM.txt**". Faça um algoritmo para ler esta matriz do arquivo e imprimir na tela se ela é ou não uma **Matriz Ortogonal**.

Utilize três **procedimentos**: um para gerar a matriz *M*, outro para calcular a sua matriz Transposta (M^T) e o terceiro para calcular a multiplicação $M.M^T$.

Utilize também uma **função** para retornar se a matriz *M* é Ortogonal ou não. A impressão dessa informação tem que ser na função *main*.

Obs.: Se uma matriz quadrada *M* é uma **matriz ortogonal**, então $\boldsymbol{M.M^T = I}$, onde M^T é a **matriz transposta** de *M* e *I* a **matriz identidade**.

3) Considere um vetor que armazena 10 números inteiros pares e 10 números inteiros ímpares todos embaralhados, ou seja, sem qualquer ordem preestabelecida. Faça um algoritmo para ler esse vetor do arquivo "**Vetor.txt**" e depois organizá-lo de modo que os **números pares** fiquem nas **posições ímpares** do vetor e os **números ímpares** fiquem nas **posições pares** do vetor. Crie dois **procedimentos**: um para preencher o vetor com os números do arquivo e o outro para organizá-lo.

Obs.: Não é permitido utilizar qualquer outro vetor/matriz para auxiliar a organização.

4) Considere o arquivo "**Numeros.txt**" com 30 números inteiros. Faça um algoritmo para armazenar esses números em um vetor e depois ordenar este mesmo vetor de maneira **não-decrescente**. Utilize três **procedimentos**: um para preencher o vetor, outro para ordenar o vetor e um terceiro para imprimir o vetor antes e depois da ordenação.

Obs.: Não é permitido utilizar qualquer outro vetor/matriz para auxiliar a ordenação.

5) Considere dois números inteiros a e b ($b \geq 0$) lidos pelo teclado. Faça um algoritmo **recursivo** para calcular o valor de a^b.

6) Considere um vetor com 20 números naturais maiores do que 1 lidos pelo teclado. Faça um algoritmo **recursivo** que organize esse vetor de modo que os números **compostos** fiquem nas **primeiras** posições e os números que **não são compostos** nas **últimas** posições. Essa organização deve ser realizada **sem utilizar qualquer estrutura de dados auxiliar**.

Crie e utilize dois **procedimentos**: um para preencher o vetor e outro **recursivo** para realizar a organização do vetor. Crie e utilize também uma **função** para retornar 1, se um número natural for composto, ou retornar 0, caso contrário.

Obs. **1**: Um número natural C é **composto** se ele tem mais de dois divisores naturais distintos.

Obs. **2**: Não é permitido utilizar qualquer outro vetor/matriz para auxiliar a ordenação.

7) Considere o arquivo "**Numeros.txt**" com 20 números inteiros que devem ser armazenados em um vetor. Faça um algoritmo **recursivo** para imprimir o **maior** valor deste vetor.

Obs.: Não é permitido utilizar qualquer outro vetor/matriz para auxiliar.