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

1) Considere uma struct que armazena os seguintes dados de um aluno: **matrícula**, **primeiro nome**, **nota da P1**, **nota da P2** e **média**. Faça um algoritmo para ler e armazenar os dados de 30 alunos em um vetor, sendo que a **média** será calculada pelo algoritmo como a média das outras duas notas. Uma vez armazenados os dados, o algoritmo tem que dividir esse vetor em dois novos vetores: o vetor dos alunos **Aprovados** e o vetor dos alunos **Reprovados**, considerando a média para aprovação maior ou igual a 6.0.

Crie dois **procedimentos**: um para preencher o vetor com os dados dos alunos e outro para dividi-lo, imprimindo os dois novos vetores ao final.

2) 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:

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

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**".

3) Considere uma matriz M de ordem 3 de números inteiros gerados aleatoriamente de 1 até 30. 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**.

4) Faça um algoritmo para preencher um vetor de tamanho 20 com números inteiros gerados aleatoriamente de 1 até 50 e depois organizar esse vetor de modo que os números **pares** fiquem nas **primeiras** posições e os números **ímpares** nas **últimas**. Essa organização deve ser realizada **sem utilizar um vetor auxiliar**.

Utilize três **procedimentos**: um para preencher o vetor, outro para realizar a organização dos números e um terceiro para imprimir o vetor (que deve ser utilizado antes e depois da organização dos números).

Exemplo com um vetor qualquer de tamanho 10:

Antes:	5	8	1	6	7	4	10	13	2	12
Depois:	12	2	10	4	6	8	5	1	7	13

5) 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 pelo teclado 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 para auxiliar a organização.

6) 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 para auxiliar a ordenação.

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

8) 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.