ATIVIDADE 2

PARTE 1 - ORDENAÇÃO C/ VETORES

Desenvolva um programa capaz de ler um array de n valores (a ser definido pelo usuário). Após a leitura, o programa deve solicitar ao usuário que escolha o algoritmo de ordenação a ser utilizado para ordenar o vetor, assim como se a ordenação será feita em ordem crescente ou decrescente. A seguir, o programa deve exibir o vetor original e o vetor ordenado.

É **obrigatório** o desenvolvimento das seguintes subrotinas:

- **le_vetor -** Função responsável pela leitura dos valores de um vetor até que o valor -100 seja lido OU o vetor chegue no tamanho máximo definido pela constante `MAX_ARR`. Deve receber uma referência ao vetor a ser preenchido e retornar a quantidade de valores preenchidos pelo usuário.

- **troca_elementos -** Procedimento que recebe como parâmetros um vetor e dois índices (i e j). A subrotina deve trocar os elementos i e j do vetor de posição.

- **selection_sort -** Procedimento responsável pela implementação do algoritmo selection sort. Deve receber uma referência ao vetor, seu tamanho e um valor inteiro que define se a ordenação será feita em ordem crescente (1) ou não (0).

- **bubble_sort -** Procedimento responsável pela implementação do algoritmo bubble sort. Deve receber uma referência ao vetor, seu tamanho e um valor inteiro que define se a ordenação será feita em ordem crescente (1) ou não (0).

Exemplos:

```
Digite os valores do vetor: 1 2 100 3 500 7 189 2 3 -100
[1, 2, 100, 3, 500, 7, 189, 2, 3]
Selecione o método de ordenação (1 - Selection, 2 - Bubble): 1
Ordenação crescente (1) ou decrescente (0): 0
[500, 189, 100, 7, 3, 3, 2, 2, 1]

Digite os valores do vetor: 3 400 -300 -2 -2 1 1 400 -100
[3, 400, -300, -2, -2, 1, 1, 400]
Selecione o método de ordenação (1 - Selection, 2 - Bubble): 2
Ordenação crescente (1) ou decrescente (0): 1
[-300, -2, -2, 1, 1, 3, 400, 400]
```

PARTE 2 - BUSCA BINÁRIA C/ VETORES

Desenvolva um programa capaz de ler um array de 10 valores (definido pela constante `ARR_MAX`). Após a leitura, o usuário deve ser capaz de informar uma chave a ser buscada dentro do array utilizando o algoritmo de **busca binária**. Para garantir que a busca binária funcione, deve-se implementar o método de ordenação **bubble sort** sempre que a busca é chamada. Ao final do programa, o índice original (de antes da ordenação) onde se encontra a chave escolhida deve ser exibido. Caso a chave não seja encontrada, o programa deve informar ao usuário que a chave não foi encontrada.

Dica: utilize uma matriz com 10 linhas e 2 colunas, onde o par de valores de cada um dos vetores linha atende ao seguinte padrão:
`[valor_original, índice_original]`

Não é obrigatório implementar nenhuma subrotina em específico.

Exemplos:

```
Digite os valores do vetor: 2 3 1 4 5 10 7 8 9 10
Digite uma chave a ser buscada: 7
[2, 3, 1, 4, 5, 10, 7, 8, 9, 10]
Valor 7 encontrado no índice 6.

Digite os valores do vetor: 1 2 3 4 5 6 7 8 9 10
Digite uma chave a ser buscada: 6
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Valor 6 encontrado no índice 5.

Digite os valores do vetor: 1 2 3 4 5 6 7 8 9 10
Digite uma chave a ser buscada: 11
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Valor 11 não foi encontrado!
```

Por quê você está aqui? O enunciado tá lá em cima, seu vagabundo!