AED2 2025 (1S) - AP02 - ALGORITMOS DE ORDENAÇÃO DE ORDEM QUADRÁTICA

Instruções:

1. E/S: tanto a entrada quanto a saída de dados devem ser "secas", ou seja, não devem apresentar frases explicativas. Siga o modelo fornecido e apenas complete as partes informadas (veja o exemplo abaixo);
2. Identificadores de variáveis: escolha nomes apropriados;
3. Documentação: inclua cabeçalho, comentários e indentação no programa;
4. O código-fonte deve ser desenvolvido em Linguagem C;
5. Submeta o programa no sistema Judge. Turma Integral: `https://judge.unifesp.br/AED2IA012025`. Turma Noturna: `https://judge.unifesp.br/AED2N012025`.

Descrição:

Implemente os algoritmos de ordenação quadráticos (*Selection Sort, Bubble Sort e Insertion Sort*) que realizem a ordenação de forma ascendente, com uma modificação: parar o processamento assim que o vetor estiver totalmente ordenado, retornando o número de comparações entre dois elementos da lista, e de trocas de posição ou movimentação de dados executados.

Para comparações, considere todas as vezes, em qualquer trecho do código, em que se comparam dois elementos do vetor/lista a ordenar, sendo verdadeira ou não o resultado dessa comparação (normalmente expresso com uma operação relacional do tipo "maior que"ou "menor que").

Para trocas ou movimentações de dados considere todas as execuções de *swaps* quando os índices dos dados a serem trocados são distintos. Dessa forma, quando os argumentos dos dados da função *swap* indicar dois valores que representam a mesma posição, a função não deve ser executada e consequentemente não computar como uma troca. Para o algoritmo *Insert Sort*, deve-se computar também como movimentações os deslocamentos efetuados nos dados do vetor, bem como a colocação do dado chave (*key*) na posição correta, quando essa nova posição for diferente da original.

Considere as seguintes condições:

1. A complexidade de cada procedimento implementado deve ser a mesma apresentada em aula;
2. O código-fonte **deve** ser escrito em C;
3. **Toda** memória alocada dinamicamente deve ser desalocada;

Solução que violem essas condições poderão ser penalizadas.

E N T R A D A:

- Um inteiro $N(1 \leq N \leq 1000)$, representando o tamanho do vetor.
- N inteiros distintos (cada um entre -10^5 e 10^5), representando os elementos do vetor, separados por um espaço em branco entre sí.

S A Í D A:

Três (3) linhas, cada qual com dois inteiros, representando a quantidade de comparações e de trocas de elementos efetuadas até finalizar a ordenação. Cada linha corresponde a um algoritmo de ordenação, na seguinte ordem: *Selection Sort*, *Bubble Sort* e *Insert Sort*.

Exemplos de entrada e saída:

- *input01*:

Entrada	**Saída**
3 1 2 3	3 0 2 0 2 0

Tabela 1: Exemplos de entrada e saída 01

- *input02*

Entrada	**Saída**
4 4 3 2 1	6 2 6 6 6 9

Tabela 2: Exemplos de entrada e saída 02

- *input03*

Entrada	**Saída**
1000 1000 999 998 ... 1	499500 500 499500 499500 499500 500499

Tabela 3: Exemplos de entrada e saída 03

O caso de teste input03 representa uma entrada com uma lista de 1000 elementos ordenados de forma decrescente. Entretanto, o código-fonte implementado deve aguardar para ler a lista com **todos** os elementos. A forma descrita é apenas para representar a lista de de forma compacta.