1. Baseado no array abaixo, desenvolva o seguinte:

```
[2, 7, 4, 9, 3]
```

Somente as alterações de troca no array:

a. Passo a passo do Bubble Sort

```
Após a 1ª iteração
[2, 4, 7, 3, 9]
Trocou 7 por 4; e 9 por 3

Após a 2ª iteração
[2, 4, 3, 7, 9]
Trocou 7 por 3

Após a 3ª iteração
[2, 3, 4, 7, 9]
Trocou 4 por 3
```

b. Passo a passo do Insertion Sort

```
[2, 7, 4, 9, 3] após a 1ª iteração (não houve troca)
[2, 7, 4, 9, 3] após a 2ª iteração (não houve troca)
[2, 4, 7, 9, 3] após a 3ª iteração (trocou 7 por 4)
[2, 4, 7, 9, 3] após a 4ª iteração (não houve troca)
[2, 3, 4, 7, 9] após a 5ª iteração (deslocamento 3 por 4)
```

⇨ Em vermelho é o limite do segmento não ordenado

c. Passo a passo do Selection Sort

Iteração	Array após ordenação	Chave	Permutação	Array Ordenado até a posição
1	[2, 7, 4, 9, 3]	2	2 e 2	0
2	[2, 3, 4, 9, 7]	3	7 e 3	1
3	[2, 3, 4, 9, 7]	4	4 e 4	2
4	[2, 3, 4, 7, 9]	7	9 e 7	3

⇨ Em vermelho é o limite dos elementos ordenados (pa-rede imaginária)

2. Baseado no array abaixo (em negrito) e o processo de ordenação, qual algoritmo produziria tais permutações?

[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

[1, 9, 8, 7, 6, 5, 4, 3, 2, 10]

[1, 2, 8, 7, 6, 5, 4, 3, 9, 10]

[1, 2, 3, 7, 6, 5, 4, 8, 9, 10]

[1, 2, 3, 4, 6, 5, 7, 8, 9, 10]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Dos três métodos utilizados, apenas o Selection Sort possui esse comportamento. Observe o comportamento da segunda linha, onde as chaves 1 e 10 são permutadas.

3. Dado um array com elementos aleatórios, ao utilizar o algoritmo Selection Sort, seria executado em tempo linear, quadrático ou algo entre os dois anteriores? Justifique sua resposta.

   O algoritmo Selection Sort sempre executará em todos os cases (pior, médio e melhor) na ordem quadrática.

4. Com base no método de ordenação a seguir, responda:

```python
def odd_even_sort(a: np.array) -> None:
    for i in range(len(a)):
        if i % 2 != 0:
            for j in range(2, len(a), 2):
                if a[j] < a[j - 1]:
                    a[j], a[j - 1] = a[j - 1], a[j]
        else:
            for j in range(1, len(a), 2):
                if a[j] < a[j - 1]:
                    a[j], a[j - 1] = a[j - 1], a[j]
```

Complexidade de Espaço? (In-place ou Out-place)	In-place
Baseado em comparação? (Sim/Não)	Sim
Recursivo? (Sim/Não)	Não
Estável? (Sim/Não)	Sim
Pior caso? O(n) ou $O(n^2)$	$O(n^2)$