Примечание 1: позиции массива размера n индексируются от 0 до $n-1$ включительно.

Примечание 2: диапазон «x..y» включает в себя граничные числа $\{x, x+1, ..., y\}$. Если $x > y$, то диапазон x..y перечисляется по убыванию $\{x, x-1, ..., y\}$

1. Для приведённого ниже алгоритма, где a это массив содержащий n чисел
   (a) (0.5 балла) выразите $T(n)$ – вычислительную сложность алгоритма в RA-модели. Оцените порядок роста времени работы алгоритма асимтотически.
   (b) (0.5 балла) что осуществляет данный алгоритм? Докажите, что он делает «это» корректно для любого входного массива. *Указание: опишите и докажите инвариант внешнего цикла.*
   (c) (1 балл) для любого n постройте перестановку массива первых n натуральных чисел, на которой приведённый алгоритм во всех итерациях внешнего цикла совершает хотя бы один «swap». Сколько всего таких перестановок существует? Почему других нет?

```
1: for i = (n − 1)..1 do
2:     for j = (n − 1)..1 do
3:         if a[j − 1] < a[j] then
4:             swap(a[j − 1], a[j])
```

Решение:

a) T(n)? Это же не рекурсивный алгоритм (в презентациях T(n) обозначалась мастер – задача для рекурсивных алгоритмов). Поэтому просто вычислилю сложность алгоритма:

$$(n-1) + (n-1) + \cdots + (n-1) = (n-1)^2 \Rightarrow O(n^2)$$

b) Данный алгоритм осуществляет обычную сортировку пузырьком, но по убыванию.

Опять-таки, сортировка пузырьком была в презентации, но так как там она была обычная, то в корректности стоит «суффикс» в нашем же случае должен быть «префикс»

Корректность алгоритма следует из того, что на каждой итерации упорядоченный префикс растет хотя бы на один элемент

c) Если мы хотим, чтобы перестановки были во всех итерациях внешнего цикла, то надо взять отсортированный массив по возрастанию.

Пример на Python:

```
def inv_bubble_sort(arr):
 n = len(arr)
 swap_count = 0
 for i in range(n - 1, 0, -1):
  for j in range(n - 1, 0, -1):
   if arr[j - 1] < arr[j]:
    swap_count += 1
    arr[j - 1], arr[j] = arr[j], arr[j - 1]
 return arr, swap_count

arr = [1, 2, 3, 4, 5]
arr, c = inv_bubble_sort(arr)
arr, c  # >> ([5, 4, 3, 2, 1], 10)
```

Всего таких перестановок может быть $\frac{n(n-1)}{2}$

```
(len(arr)*(len(arr)-1))/2
>> 10.0
```

Других не может быть потому, что это максимальное возможное число перестановок. Если брать другой массив, то их просто на просто станет меньше

2. (1 балл) Пусть $p(n) = \sum_{i=0}^{d} a_i n^i$ полином степени d от n, в котором все коэффициенты $a_i > 0$.
   Для фиксированной константы $k \in \mathbb{N}$ докажите следующие утверждения:
   (a) Если $k \geq d$, то $p(n) = O(n^k)$
   (b) Если $k = d$, то $p(n) = \Theta(n^k)$
   (c) Если $k \leq d$, то $p(n) = \Omega(n^k)$
   Указание: воспользуйтесь определениями O, Ω, Θ
3. (1 балл) Докажите, что $\log n! = \Theta(n \log n)$ для любой базы логарифма.
   Указание: найдите такие константы $C_1, C_2 > 0$, и $n_0 \in \mathbb{N}$, что $\forall n > n_0$ выполняется:

$$C_1 \cdot n \log n \leq \log n! \leq C_2 \cdot n \log n$$

Решение:

$$n! = n \cdot (n-1)! \quad \text{для } n = 1, 2, 3, \cdots \Rightarrow n! \leq n^2$$

Следовательно, получается выражение:

$$\log n! \leq C_2 \cdot \log n^n \Rightarrow C_2 = 1$$

Рассмотрим $\frac{n}{2}$ множетелей: $(\frac{n}{2}+1) \cdot (\frac{n}{2}+2) \cdots \geq (\frac{n}{2})^{\frac{n}{2}} \Rightarrow$ необходимо, чтобы выполнялось $\log n! \geq \log((\frac{n}{2})^{\frac{n}{2}})$

$$\log((\frac{n}{2})^{\frac{n}{2}}) = \frac{n}{2}\log\frac{n}{2} = \frac{n}{2}\log n - \frac{n}{2}\log 2 = \frac{n}{2}\log n(1 - \frac{\log 2}{\log n})$$

$$\log n! \geq \frac{n}{2}\log n(1 - \frac{\log 2}{\log n})$$

Найдем n:

$$(1 - \frac{\log 2}{\log n}) \geq \frac{1}{2}$$
$$\log n \geq 2\log 2 = \log 4$$
$$\Rightarrow n \geq 4$$

$$\log n! \geq \frac{1}{4} n \log n$$

Получается: $C_1 = \frac{1}{4}$, $C_2 = 1$, $n = 4$