Домашнее задание

Основы синтакиса

Всего баллов: 10

Бонусные баллы: 1

Эта домашка на «общее владение синтаксисом». Будет хорошо, если вы освоетесь с языком с уже знакомыми задачами, попадаете на какие-то грабли с *BorrowChecker*-ом, набьете шишек. Дальше подобных заданий практически не будет.

1. Максимум

Баллы: 1

Напишите функцию `max(a: u32, b: u32) -> u32`, возвращающую максимум двух чисел.

2. Сортировка

Баллы: 5

Напишите функцию, принимающую на вход `Vec<f64>`, сортирующую его по возрастанию. Сортировка может быть любая, даже работающая за $O(n^2)$. Встроенными методами сортировки пользоваться нельзя — да и просто так не получится, **Rust** знает, что среди `float`-ов бывают `NaN`(Not a Number), а они очень плохо сортируются. Но это часть спецификации `float`-ов во всех языках, ничего не поделать.

Ваша сортировка должна перемещать `NaN`-ы в конец массива.

P.S. Разумеется, встроенными (точнее, *обобщёнными* и *абстрактными*) методами воспользоваться можно, и мы это потом сделаем.

Мы дадим *1 бонусный балл* за $O(n \log n)$. Напишите это в комментарии, чтобы мы не пропустили.

3. Большой распил

Баллы: 4

Эту задачу я стырил с Codeforces, с одного из давних школьных контестов. Она совсем несложная, но может потребоваться чуть-чуть подумать. Думаю, простенькие «олимпиадные» задачи здесь решали все.

Вам дана последовательность целых чисел (высот глаз агентов), содержащая одинаковое количество чётных и нечётных чисел. Требуется в условиях ограниченного бюджета построить максимальное число перегородок, которые «распилят» последовательность (пирог) на непустые отрезки, на каждом из которых количество чётных чисел равно количеству нечётных чисел.

Перегородки разделяют последовательность на непрерывные подряд идущие отрезки, например:
`[4,1,2,3,4,5,4,4,5,5] -> построили две перегородки -> [4,1|2,3,4,5|4,4,5,5]`.

После разрезов на каждом отрезке количество четных элементов должно быть равно количеству нечётных.

Стоимость построения перегородки между агентами i и $i+1$ составляет $|h_i - h_{i+1}|$ у.е. Найдите максимальное количество перегородок, которые можно построить, потратив при этом не более b у.е.

Напишите функцию `saw(elements: Vec<u64>, b: u64) → usize`

Где целое число b ($1 \le b \le 100$) — объём бюджета, которым вы располагаете.

Вектор `elements` содержит n чисел $h_1, h_2, ..., h_n (1 \le h_i \le 100, 2 \le n \le 100)$ — уровни . Последовательность *содержит одинаковое количество чётных и нечётных элементов*.

Функция должна возвращать одно число — максимальное количество разрезов, которые можно провести, потратив не более b биткоинов.

Примеры:

```
b: 4
elements: 1 2 5 10 15 20
→ 1
```

```
b: 10
elements: 1 3 2 4
→ 0
```

```
b: 100
elements: 1 2 3 4 5 6
→ 2
```