Задача A. Счастливый день

Для начала напишем функцию, определяющую, является ли поступающее ей на вход число x красивым. Для этого воспользуемся алгоритмом перевода в k-ичную систему счисления при $k = 10$.

```
function isBeautiful(x: integer): boolean;
var
  cnt: integer;
  sum: integer;
begin
  cnt:= 0;
  sum:= 0;
  while (x > 0) do begin
    inc(cnt);
    sum:= sum + (x mod 10);
    x:= x div 10;
  end;
  result:= (sum mod cnt = 0);
end;
```

Далее, напишем программу, последовательно проверяющую все натуральные числа, и находящую таким образом n-ое красивое число.

```
readln(n);
i:= 1;
  while (true) do begin
    if (isBeautiful(i)) then begin
      n:= n - 1;
    end;
    if (n = 0) then begin
      writeln(i);
      break;
    end;
  i:= i + 1;
end;
```

Задача B. Игровое поле

Самый простой алгоритм состоит в том, чтобы перебрать все элементы матрицы и для каждого из них проверить верно ли, что он не больше остальных элементов своей строки и не меньше остальных элементов своего столбца. Однако время работы этого алгоритма есть $O(nm(n+m))$, поэтому при заданных в условии задачи ограничениях он не укладывается в ограничения по времени.

Поэтому необходимо придумать более быстрый алгоритм. Заметим, что для того, чтобы проверить, является ли элемент матрицы минимальным в

своей строке, не требуется его сравнивать со всеми элементами строки, а достаточно сравнить только с минимальным из них (на равенство). Для столбцов можно действовать аналогично.

Задача C. Дюны

Порывы ветра:

- было n порывов ветра
- i-ый порыв менял высоту участков дюны с l_i-го по r_i-й
- изменение высоты: $+x_i$, $-x_i$, $+x_i$, $-x_i$, ...

Запросы:

- m запросов: «какая высота в итоге у q_i-го участка?»

Количество участков от 1 до миллиарда, поэтому не надо моделировать миллиард участков дюны!

Для каждого запроса – участка номер q_i – вычислим его высоту. Для этого просмотрим заново все n порывов ветра, про каждый посчитаем, менял ли он высоту участка q_i и как именно.

Задача D. Игра

Мальчикам выгодно начать кидать на столбики, которые есть у обоих.

Считаем both — количество столбиков, на которые оба могут закинуть. NPetya и NVasya — количества столбиков, на которые может закинуть только Петя и на которые может закинуть только Вася, соответственно.

Далее моделируем ходы по очереди.

- both > 0, то обязательно закидываем на общие.
- both = 0 и NPlayer > 0, то закидываем на соответствующий столбик.
- both = 0 и NPlayer = 0, то больше не закинуть.