Balance the Board

You place n^2 indistinguishable pieces on an $n \times n$ chessboard, where $n = 2^{90} \approx 1.27 \times 10^{27}$. Of these pieces, n of them are slightly lighter than usual, while the rest are all the same standard weight, but you are unable to discern this simply by feeling the pieces.

However, beneath each row and column of the chessboard, you have set up a scale, which, when turned on, will tell you *only* whether the average weight of all the pieces on that row or column is the standard weight, or lighter than standard. On a given step, you are allowed to rearrange every piece on the chessboard, and then turn on all the scales simultaneously, and record their outputs, before turning them all off again. (Note that you can only turn on the scales if all n^2 pieces are placed in different squares on the board.)

Find an algorithm that, in at most k steps, will always allow you to rearrange the pieces in such a way that every row and column measures lighter than standard on the final step.

Scoring

An algorithm that completes in at most k steps will be awarded:

- 1 pt for $k > 10^{55}$
- 10 pts for $k = 10^{55}$
- 30 pts for $k = 10^{30}$
- 50 pts for $k = 10^{28}$
- 65 pts for $k = 10^{20}$
- 80 pts for $k = 10^5$
- 90 pts for $k = 2021$
- 100 pts for $k = 500$

You are allowed to prove multiple bounds, and will receive points for the best bound that is correctly proven. **Please state which bound you are trying to prove above any proof, or you may not be awarded points for that bound.**

Partial points may be awarded for progress towards the next bound, and partial points may be deducted for logical flaws or lack of rigor in proofs. To get full points, you must demonstrate that your algorithm always produces a correct result, and always runs in at most the stated number of moves.

Multiply to the Moon

You are initially given the number $n = 1$. Each turn, you may choose any positive divisor $d \mid n$, and multiply n by $d + 1$. For instance, on the first turn, you must select $d = 1$, giving $n = 1 \cdot (1 + 1) = 2$ as your new value of n. On the next turn, you can select either $d = 1$ or 2, giving $n = 2 \cdot (1 + 1) = 4$ or $n = 2 \cdot (2 + 1) = 6$, respectively, and so on.

Find an algorithm that, in at most k steps, results in n being divisible by the number $2021^{2021^{2021}} - 1$.

Scoring

An algorithm that completes in at most k steps will be awarded:

- 1 pt for $k > 2021^{2021^{2021}}$
- 20 pts for $k = 2021^{2021^{2021}}$
- 50 pts for $k = 10^{10^4}$
- 75 pts for $k = 10^{10}$
- 90 pts for $k = 10^5$
- 95 pts for $k = 6 \cdot 10^4$
- 100 pts for $k = 5 \cdot 10^4$

You are allowed to prove multiple bounds, and will receive points for the best bound that is correctly proven. **Please state which bound you are trying to prove above any proof, or you may not be awarded points for that bound.**

Partial points may be awarded for progress towards the next bound, and partial points may be deducted for logical flaws or lack of rigor in proofs. To get full points, you must demonstrate that your algorithm always produces a correct result, and always runs in at most the stated number of moves.

Topologist's Trap the Tiger

There is a tiger (which is treated as a point) in the plane that is trying to escape. It starts at the origin at time $t = 0$, and moves continuously at some speed k. At every positive integer time t, you can place one closed unit disk anywhere in the plane, so long as the disk does not intersect the tiger's current position. The tiger is not allowed to move into any previously placed disks (i.e. the disks block the tiger from moving). Note that when you place the disks, you can 'see' the tiger, (i.e. know where the tiger currently is).

Your goal is to prevent the tiger from escaping to infinity. In other words, you must show there is some radius $R(k)$ such that, using your algorithm, it is impossible for a tiger with speed k to reach a distance larger than $R(k)$ from the origin (where it started).

Find an algorithm for placing disks such that there exists some finite real $R(k)$ such that the tiger will never be a distance more than $R(k)$ away from the origin.

Scoring

An algorithm that can trap a tiger of speed k will be awarded:

- 1 pt for $k < 0.05$
- 10 pts for $k = 0.05$
- 20 pts for $k = 0.2$
- 30 pts for $k = 0.3$
- 50 pts for $k = 1$
- 70 pts for $k = 2$
- 80 pts for $k = 2.3$
- 85 pts for $k = 2.6$
- 90 pts for $k = 2.9$
- 100 pts for $k = 3.9$

You are allowed to prove multiple bounds, and will receive points for the best bound that is correctly proven. **Please state which bound you are trying to prove above any proof, or you may not be awarded points for that bound.**

Partial points may be awarded for progress towards the next bound, and partial points may be deducted for logical flaws or lack of rigor in proofs. To get full points, you must demonstrate that your algorithm always produces a correct result, and always runs in at most the stated number of moves.