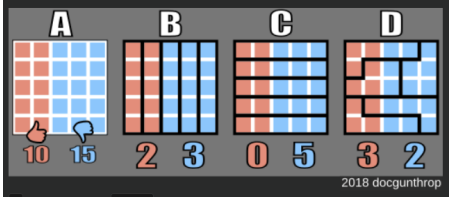
gerrymander — *noun.* the dividing of a state, county, etc., into election districts so as to give one political party a majority in many districts while concentrating the voting strength of the other party into as few districts as possible.

Objective

Given a `5 x 5` region populated by `25` citizens, your task is to write a function that divides the region into `5` districts given the following conditions:

- `10` citizens will vote for your candidate, while the other `15` will vote for the opponent
- Your candidate must win the popular vote for `3` of the `5` districts
- Each district must have an equal number of voters
- Each district must be one contiguous cluster of voters (i.e. each voter has one or more orthogonally adjacent neighbors from the same district)

Concept Overview



`A` : You're given a `5 x 5` square matrix representing the layout of the region occupied by eligible voters. The following panels show different ways to set boundaries for `5` districts.

- `B` : Proportionate outcome — blue and red win in proportion to their voting
- `C` : Disproportionate outcome — blue wins all
- `D` : Disproportionate outcome — red wins majority despite having fewer total supporters

Your function must solve the challenge presented in panel `D`

Input

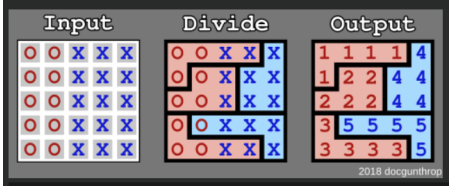
Your function will receive a newline-separated string consisting of `X` and `O` characters. The `O`s represent the voters in support of your candidate, and the `X`s represent those in support of the opponent.

Output

Your function should return a `5x5` newline-separated string comprised of the digits `1` through `5` where each group of identical digits represents its own unique district.

If a solution does not exist, return `null`, `None`, or `nil`

Test Example




```
region = [
    'OOXXX',
    'OOXXX',
    'OOXXX',
    'OOXXX',
    'OOXXX'
]

# one possible solution where regions 1,2, and 3 are won
gerrymander('\n'.join(region)) # '11114\n12244\n22244\n35555\n33335'
```

Testing Constraints

- Full Test Suite: `10` fixed tests and `10` randomly-generated tests
- Zero or more valid solutions will exist for each test
- Inputs will always be valid