Supermarket Maze

Instructions

While shopping for groceries you find that the supermarket is like a maze filled with blocked walkways that hinder your shopping sprees.

Given the floor plan of the supermarket, find the number of steps made within the shortest path from the entrance to the checkout counter.

To make things interesting some mazes are unsolvable, in that case return -1 as number of steps to solve the maze!

Expose a `POST` endpoint `/supermarket` for us to verify!

Input

```
{
    "tests": {
        "0":{
            "maze": [[1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1],
                     [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
                     [1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1],
                     [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1],
                     [1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1],
                     [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1],
                     [1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1],
                     [1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1],
                     [1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1],
                     [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1],
                     [1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]],
            "start": [3, 0],
            "end": [1, 10]
        },
        "1": {
            "maze": [[1, 1, 1, 0, 1, 1, 1, 1, 1],
                     [1, 1, 1, 0, 0, 0, 0, 0, 1],
                     [1, 1, 1, 0, 1, 1, 1, 0, 1],
```

```
                    [1, 1, 1, 0, 0, 0, 1, 0, 1],
                    [1, 1, 1, 1, 1, 0, 1, 0, 1],
                    [1, 0, 0, 0, 1, 0, 1, 0, 1],
                    [1, 0, 1, 1, 1, 0, 1, 0, 1],
                    [1, 0, 0, 0, 0, 0, 1, 0, 1],
                    [1, 1, 1, 1, 1, 1, 1, 0, 1]],
            "start": [3, 0],
            "end": [7, 8]
        },
        "2": {
          "maze": [[1, 1, 1, 0, 1, 1, 1, 1, 1],
                   [1, 1, 1, 0, 0, 0, 0, 0, 1],
                   [1, 1, 1, 0, 1, 1, 1, 0, 1],
                   [1, 1, 1, 0, 0, 0, 1, 0, 1],
                   [1, 1, 1, 1, 1, 0, 1, 1, 1],
                   [1, 0, 0, 0, 1, 0, 1, 0, 1],
                   [1, 0, 1, 1, 1, 0, 1, 0, 1],
                   [1, 0, 0, 0, 0, 0, 1, 0, 1],
                   [1, 1, 1, 1, 1, 1, 1, 0, 1]],
          "start": [3, 0],
          "end": [7, 8]
        },
        ...
    }
}
```

Output Expected

```
{
    "answers": {
            "0": 29,
            "1": 13,
            "2": -1,
            ...
    }
}
```

Limitations

1. Assume that the start point is `(X,0)` where `X can be the x-coordinate within the length of the maze`
2. No diagonal movement
3. The floor plan
    - A wall is represented by `1`
    - A walkable space is represented by `0`

Example

Example Input

```
[1, 1, 1, 0, 1, 1, 1]
[1, 0, 1, 0, 0, 0, 1]
[1, 0, 0, 0, 1, 0, 1]
[1, 1, 0, 1, 1, 1, 1]
[1, 0, 0, 0, 0, 1, 1]
[1, 1, 1, 1, 0, 1, 1]

Coordinates will be given in as [x-axis, y-axis]

start: [3, 0]
end: [4, 5]
```

Example Output

```
answer: 9
```

Visualization

[1, 1, 1, **0**, 1, 1, 1]
[1, 0, 1, **0**, 0, 0, 1]
[1, 0, **0**, **0**, 1, 0, 1]
[1, 1, **0**, 1, 1, 1, 1]
[1, 0, **0**, **0**, **0**, 1, 1]
[1, 1, 1, 1, **0**, 1, 1]

Actual number of test cases given may change.

Inventory Management

Gotham Corporation is way past its prime and has been leaking money from all the fronts. Bruce Wayne has decided to plug the leakages and wanted to make Gotham great again and appoints you as the inventory manager for the corporation. The current inventory management system search looks for exact matches, ignoring spelling mistakes in the database. This results in redundant inventory being purchased.

Task Objective

You need to build a search algorithm which returns possible matches in case of no exact match by following the instructions given below.

Only the following operations are allowed on the search item characters to find the possible matches.

- Insertion
- Deletion
- Substitution

You should return top 10 matches (case insensitive) in the increasing order of minimum number of operations performed on the search item. You need to provide the operations performed on the search item in your result. Any possible set of operations on the search item are allowed as long as the target result is obtained by applying the operations on the search item.

If two items have same number of operations, order them alphabetically.

We will do a POST request on your team URL with the endpoint /inventory-management to evaluate the solution

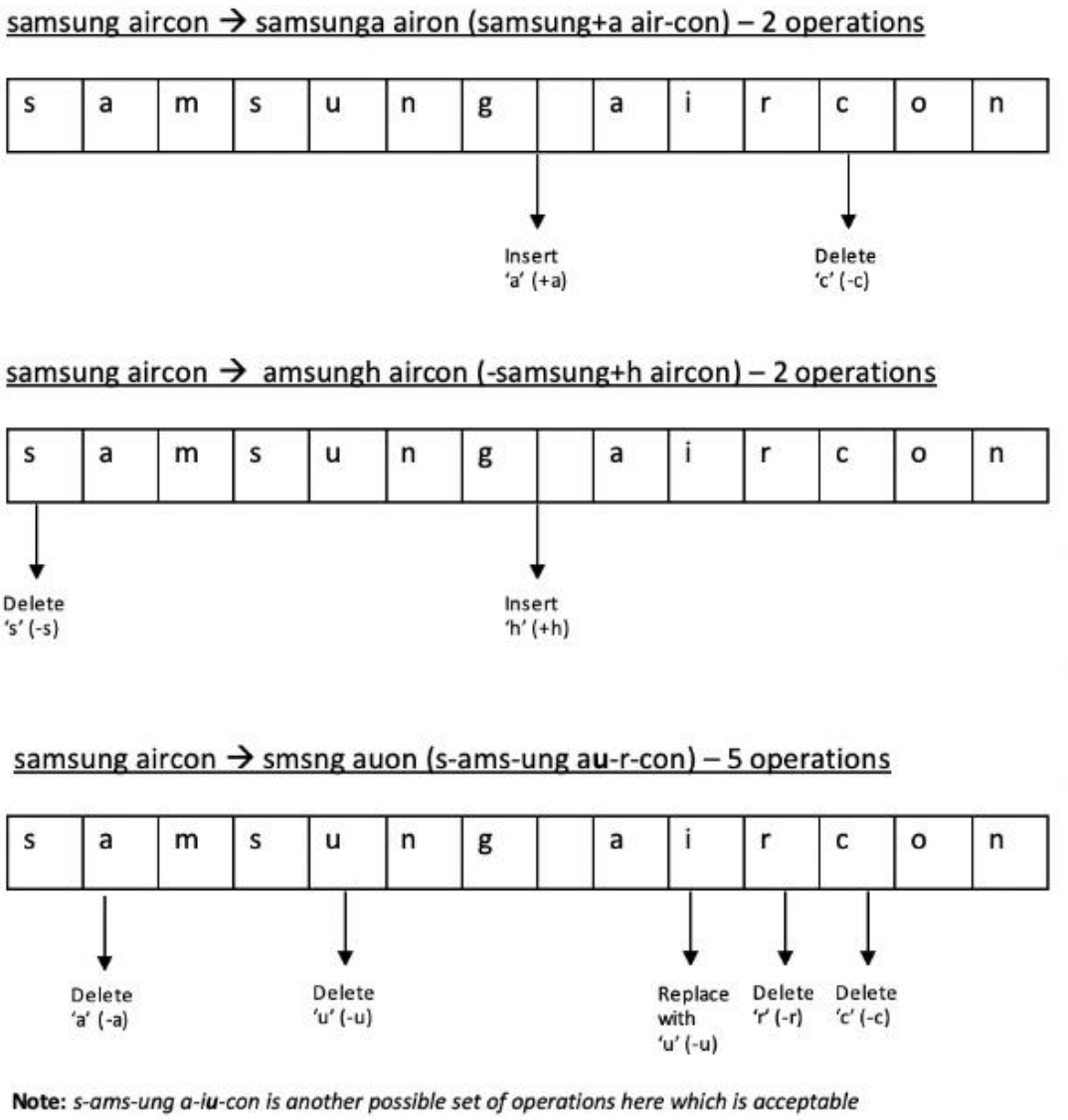
Sample Input

```
[{"searchItemName":"Samsung Aircon","items":["Smsng
Auon","Amsungh Aircon","Samsunga Airon"]}]
```

Sample Output

```
[{"searchItemName":"Samsung Aircon","searchResult":["-
Samsung+h Aircon","Samsung+a Air-con","S-ams-ung Au-r-con"]}]
```

Explanation



Note: *s-ams-ung a-iu-con is another possible set of operations here which is acceptable*

Revisit Geometry

You are given coordinates of a shape and straight line. Your task is to find out all the possible points of intersection.

Requirements

- Expose a POST method with endpoint `/revisitgeometry`
- Return output result as json

Constraints

- Input will be a JSON object of shape and line coordinates.
- Join the coordinates in the same order of input.
- Use straight lines to form the shape.
- Input line will not overlap with any of the lines of the shape resulting in infite points of intersection.
- n is an integer specifying the number of coordinates. 1 <= n <= 100.
- The challenge is calibrated to a precision of .01

Input description

Use straight line to join the co-ordinates and form the shape.
Please join the coordinates in the same order of input and ensure that the last and first co-ordinate are joined by a line to complete the shape.
Note: Shape must remain fixed, but you can extrapolate the straight line in any direction for infinite length to find the points of intersection. If the shape and straight line do not intersect, return an empty array [].
The HTTP POST request will come with a body of Content-Type: application/json

Scenario 1

Sample input JSON

```
{
  "shapeCoordinates": [
    { "x": 21, "y": 70 },
    { "x": 72, "y": 70 },
    { "x": 72, "y": 127 }
  ],
  "lineCoordinates": [
    { "x": -58, "y": 56 },
    { "x": -28, "y": 68 }
  ]
}
```

Sample output JSON

```
[
```

```
  { "x": 72, "y": 108 },
  { "x": 45.52, "y": 97.41 }
]
```

Scenario 2

Sample input JSON

```
{
  "shapeCoordinates": [
    { "x": -21, "y": -18 },
    { "x": 71, "y": -18 },
    { "x": 71, "y": 71 },
    { "x": -21, "y": 71 }
  ],
  "lineCoordinates": [
    { "x": 68, "y": -8 },
    { "x": 108, "y": 42 }
  ]
}
```

Sample output JSON

```
[
  { "x": 60, "y": -18 },
  { "x": 71, "y": -4.25 }
]
```

Scenario 3

Sample input JSON

```
{
      "shapeCoordinates":[
         {
            "x":63,
            "y":26
         },
```

```
            {
               "x":115,
               "y":26
            },
            {
               "x":115,
               "y":54
            },
            {
               "x":63,
               "y":54
            }
         ],
         "lineCoordinates":[
            {
               "x":-88,
               "y":85
            },
            {
               "x":-58,
               "y":97
            }
         ]
      }
```


Sample output JSON


```
[]
```


Salad Spree

Background

Meet Tom, software developer by profession, but only because noone has figured out how to make a living out of his favourite hobbies, consuming salad and being lazy.
Tom will go to the greatest lengths to not move a muscle in his body, rumour has it that he went a month without eating because the dishes weren't clean.

While he wears his laziness like a badge of honour, he has recently discovered that it's possible to be lazy and healthy at the same time.
Tom has discovered that the miracle food - 'Salad' is the panacea to all his problems. He can consume nearly infinite amounts of it while putting on no weight.

Problem

Alas, there still is a thorn in his plan, he needs to walk to stores to buy the amounts of salads he requires, and due to recent food shortages, each store that sells salads is only selling one salad per customer.
Tom is also incredibly miser and wants to make sure that he spends the least amount of money possible procuring his salads.
He has crafted an intelligent plan and decided that he will only buy salad from stores that are next to each other, that way he minimises the walking that he needs to do.
However, all this thinking has rendered Tom too tired and therefore he has hired you to write a program which will tell him the least amount of money he would need to spend to buy the number of salads he needs from consecutive stores.
Tom is too lazy and if there aren't enough consecutive stores selling the number of salads he needs, he will not buy any salads, therefore your solution should return 0 in that case.

Endpoint

Provide a `POST` endpoint `/salad-spree` that given 1 set of input will return 1 set of output

Input

The `HTTP POST` request will come with a body of `Content-Type: application/json` .

```
{

    "number_of_salads" : n,
```

```
    "salad_prices_street_map" : S (S is an array of arrays where each array indicates the price of salads in each store on a certain street. A price of "X" indicates that the store doesn't sell salads)

}
```

Output

The expected `HTTP` response will come with a body of `Content-Type: application/json` containing

```
{

    result : (The minimum amount of money that Tom needs to spend to buy n salads from n consecutive stores that sell salad)

}
```

Constraints

- 1 ≤ ***n*** ≤ ***size of each array inside S*** ≤ 100
- HTTP Request Timeout: 500 milliseconds

i.e. You should be expecting the number of required salads to be smaller or equal to the number of shops per street. Both of these numbers should not exceed 100.

Examples

Sample Input 1

```
{

    "number_of_salads" : 3,

    "salad_prices_street_map" : [["12", "12", "3", "X", "3"], ["23", "X", "X", "X", "3"], ["33", "21", "X", "X", "X"], ["9", "12", "3", "X", "X"], ["X", "X", "X", "4", "5"]]

}
```

Sample Output 1

```
{
```

```
    "result": 24

}
```

Explanation 1

The solution in this case is on the 4th street, with 9, 12 and 3 being the lowest possible combination of three consecutive shops to buy salad from.

Sample Input 2

```
{

    "number_of_salads" : 3,

    "salad_prices_street_map" : [["X", "X", "2"], ["2", "3", "X"], ["X", "3", "2"]]

}
```

Sample Output 2

```
{

    "result": 0

}
```

Explanation 2

Your solution should return 0, since no street has three consecutive stores that sell salads.

Sample Input 3

```
{

    "number_of_salads" : 2,

    "salad_prices_street_map" : [["2", "3", "X", "2"], ["4", "X", "X", "4"], ["3", "2", "X", "X"], ["X", "X", "X", "5"]]

}
```

Sample Output 3

```
{

    "result": 5

}
```

Explanation 3

Even though there is more than one solution, the correct answer is still 5.

Contact Tracing

A recent pandemic outbreak is spreading fast. CodeIT-Suisse is one of the very few countries where the number of infected cases is only a handful. The administration of CodeIT-Suisse wants to contain the spread of the outbreak within the country. As part of the containment plan, it wants to trace how each infected case in the country has contracted the infection.

A Health Organisation says that the infected can be traced back to the origin by analysing the genome patterns of the infected against the available infected genome data set of the pandemic.

When the virus mutates and spreads, its genome code can be altered by one or two characters in its genome string of three character instructions. The minimal the alterations between the two genome strings, the more likely that they both are related.

Also, if the genome string has more than one instruction with instruction's first character altered, the mutation is said to be non-silent. In case of non-silent mutation, the virus exhibits change in its effectiveness.

Can you write a program that could output the possible tracing paths of the infected and whether there is a non silent mutation of the virus in the trace?

Input and Output

Input

- You will be given a JSON string with Infected, Origin, and cluster of genomes information.

Output

- You should return a String array of possible trace paths from Infected to Origin.

Rules and Constraints

- Expose a post endpoint "/contact_trace" that takes JSON input and returns a string array of trace paths.
- Your output should contain all possible trace paths.
- All genome strings in the input are of same and fixed length.
- "name" field in the input json is unique.
- In case of non-silent mutation indicate it with an "*" suffixed to the name in the trace path

Trace String format

```
(genome-name)(* [if non-silent mutation])(white-space)->(genome-name)(* [if non-silent mutation])(white-space)->...
```

Sample Input/Output

Post Endpoint

`/contact_trace`

Input #1

```
{
    "infected": {
        "name":"orange",
        "genome":"acg-gcu-uca-gca-acu-ccc-gua-acg-gcu-uca-gca-acu-cac-gaa"
    },
    "origin": {
        "name":"turquoise",
        "genome":"acg-gcu-uca-gca-acu-ccc-gua-acg-gcu-uca-gca-acu-cac-gaa"
```

```
    },
    "cluster":[
        {
            "name":"magenta",
            "genome":"acg-gcu-uca-gca-acu-ccc-gua-acg-gcu-uca-gca-acu-cac-
gaa"
        }
    ]
}
```

Output #1

```
[
    "orange -> magenta",
    "orange -> turquoise"
]
```

Explanation

- All genomes of infected, origin, and cluster ones in the above test case are all same which means the infected could be contacted the virus either from "magenta" or from "turquoise". Hence the output.

Input #2

```
{
    "infected": {
        "name":"plastic",
        "genome":"acg-gcu-uca-gca-acu-ccc-gua-acg-gcu-uca-gca-acu-cac-gaa"
    },
    "origin": {
        "name":"metal",
        "genome":"acg-acu-uca-gca-acu-ccc-gua-acg-ccu-uca-gca-acu-cac-gac"
    },
    "cluster":[
        {
            "name":"thread",
            "genome":"acg-acu-uca-gca-acu-ccc-gua-acg-ccu-uca-gca-acu-cac-
gaa"
```

```
        }
    ]
}
```

Output #2

```
["plastic* -> thread -> metal"]
```

Explanation

Snakes & Ladders, Smoke & Mirrors

How To Play

Snakes & Ladders, Smoke & Mirrors is an extension of regular Snakes & Ladders game.

It is played with a single six-sided die.

Beside the jumps Snakes and Ladders, there are two additional jumps, Smoke and Mirror, on the board.

- Smoke
  - Roll the die again, and move *backwards* by that value (like how Snake moves backwards).
- Mirror
  - Roll the die again, and move *forwards* by that value (like how Ladders moves forwards).

The game ends when a player reaches the end.

If the player overshoots, the player will move backwards by the extra steps.

For example, if the board is 10x10, the player is on square `99` and rolled `2`, the final square will still be `99` (given `(100 - (99 + 2 - 100))`).

This version of the game *does not* grant an additional die roll when a six (6) is rolled.

Objective

Given a board and number of players, you will return a list of die rolls such that the game ends with the *last* player winning.

Upon a successful play with the die rolls, a score of `(board size) / (number of squares landed)` will be given.

Otherwise, the score is 0.

We will do a `POST` request on your team URL with the endpoint `/slsm`.

Scoring

Your solution will be evaluated (i.e. played) twelve times with varying difficulty levels, which are added to give a raw score.

If the last player wins all games, the raw score will be divided by two and be at least 20, else the raw score will be divided by four, to arrive at the final score.

The maximum point is 100, before factoring in challenge difficulty weightage.

Constraints

- Board size, `boardSize`, is a product of width and height, which will each be between `8` and `20`, inclusive.
- Number of players, `players`, will be between `2` and `8`, inclusive.
- Squares are `1`-based, so the first square is `1` and the last square will be `width * height`.
- Both the first and last squares will not be the start or end of any jump.

- A Smoke will not appear before the sixth square (so that the second die roll will not go beyond the first square).
- A Mirror will not appear within the last six squares (so that the second die roll will not go beyond the last square).
- None of the jumps' squares will conflict with one another.
- There will be at least one of each jump.

Sample input

```
{"boardSize":100,"players":4,"jumps":["59:39","9:37","42:0","0:3"
```

```
]}
```

Explanation:

- Board size: 100
- Players: 4
- Jumps (directions are important!):
  - Snake: From `59` to `39`
  - Ladder: From `9` to `37`
  - Smoke: From `42` to `-<0>`, `<0>` being a die roll placeholder (direction `42:0` resembling a Snake)
  - Mirror: From `3` to `+<0>`, `<0>` being a die roll placeholder (direction `0:3` resembling a Ladder)
  - Usage of `0` as a die roll placeholder is to facilitate parsing of values as integers

Sample output

```
[1, 2, 3, 4, 5, 6]
```

A list of die rolls.

Live example

Click here.

The input example is in `"input"` and the output (i.e. your endpoint result) is in `"output"`.

Cluster

Given an area (2D array of '1's, '0s' and '*'s) at a specific time snapshot, you are tasked to find the number of clusters.

'1' represents an infected individual who has the ability to transmit the virus.

'0' represents an uninfected individual who can be infected by an infected person . If infected, this individual has the ability of an infected individual, and transmit the virus.

'*' represents a space where there are no individuals.

The virus can be transmitted horizontally, vertically and diagonally by a distance of 1 in the area.

A cluster is formed if there are infected individuals, or if there are virus transmissions between infected individuals and non-infected individuals.

You can assume the area is always made up by a constant length and breadth and can disregard anything beyond the given area. Individuals are also static and cannot move.

Constraints

3 <= length of area <= 1000

3 <= breadth of area <= 1000

Sample Input 1

```
  [
    ["*", "*", "*", "*"],
    ["*", "1", "*", "*"],
    ["*", "*", "*", "*"],
    ["*", "*", "*", "1"],
    ["*", "*", "*", "*"]
  ]
```

Sample Output 1

```
{
  "answer": 2
}
```

Sample Input 2

```
  [
    ["*", "*", "*", "*"],
    ["*", "1", "0", "*"],
    ["*", "*", "*", "*"],
    ["*", "*", "0", "0"],
    ["*", "*", "*", "*"]
  ]
```

Sample Output 2

```
{
  "answer": 1
}
```

Sample Input 3

```

```

```
  [
    ["*", "*", "*", "*"],
    ["*", "1", "0", "*"],
    ["*", "*", "*", "1"],
    ["*", "*", "0", "0"],
    ["*", "*", "*", "*"]
  ]
```

Sample Output 3

```
{
  "answer": 1
}
```

Sample Input 4

```
  [
    ["1", "0", "*", "*"],
    ["0", "0", "*", "0"],
    ["*", "*", "0", "*"],
    ["*", "*", "*", "*"],
    ["*", "*", "*", "*"],
    ["*", "0", "0", "*"]
  ]
```

Sample Output 4

```
{
  "answer": 1
}
```

Sample Input 5

```
  [
    ["*", "*", "*", "*", "*", "*", "*", "*", "*"],
    ["*", "0", "0", "0", "*", "*", "*", "*", "*"],
```

```
    ["*", "*", "1", "*", "*", "*", "*", "*", "*"],
    ["*", "0", "0", "0", "*", "*", "*", "*", "*"],
    ["*", "*", "*", "*", "0", "*", "*", "*", "*"],
    ["*", "*", "*", "*", "*", "0", "0", "*", "*"],
    ["*", "*", "*", "*", "1", "*", "*", "*", "0"],
    ["*", "*", "*", "*", "0", "*", "*", "0", "0"],
    ["*", "*", "*", "*", "*", "*", "*", "*", "*"],
    ["*", "*", "*", "*", "*", "*", "*", "*", "*"],
    ["*", "*", "*", "*", "*", "*", "*", "*", "*"],
    ["*", "*", "1", "0", "0", "*", "*", "*", "*"],
    ["*", "*", "*", "*", "*", "*", "*", "*", "*"]
  ]
```

Sample Output 5

```
{
  "answer": 2
}
```

Evaluation

Your solution will be run against multiple random test cases. Be sure to include corner cases.

Requirements

Expose 1 POST endpoint /cluster for evaluation

Clean the Floor

Instructions

You find that cleaning the floor with the least amount of movement helps to calm your nerves. A value higher than 0 represents the dirtiness of the floor. Find out what is the least number of moves to clean the floor.

Expose a `POST` endpoint `/clean_floor` for us to verify!

Input

```
{
  "tests": {
    "0": {
      "floor":[0, 1]
    },
    "1": {
      "floor":[1, 1]
    },
    ...
  }
}
```

Output Expected

```
{
  "answers": {
    "0": 1,
    "1": 2,
    ...
  }
}
```

Itty Bitty Details

1. The start point is always at position 0 of the array
2. Each move traverses 1 position in the array and performs 1 action
3. There are only 2 possible actions per move:
    - If the position has **dirt level > 0** then the dirt level decreases by 1 *(Spend some time and effort to clean the floor)*

    - If the position has **dirt level = 0** then the dirt level increases by 1 *(You walked on a clean floor so you dirty it)*
4. When all values in the floor have a **dirt level = 0**, then the floor is clean!

Examples

Input 1

```
[0, 1]
```

Output 1

```
1 Move to clean
Move 0: [0, 1]
Move 1: [0, 0] // You're done cleaning!
```

Input 2

```
[1, 1]
```

Output 2

```
2 Moves to clean
Move 0: [1, 1]
Move 1: [1, 0]
Move 2: [0, 0] // You're done cleaning!
```

Actual number of test cases given may change.

GMO Engineering for a Sustainable Food Supply

The world is facing a food supply crisis!

In order to ensure sustainable and self-sufficient food supply, you are tasked to lead the national food council to increase the yield of crops by increasing its drought-resistance index (DRI) through genome sequencing.

A higher DRI would indicate higher survivability of the crop, increasing the yield of the crops.

A genome sequence is made up of exactly four alphabetical characters: A, C, G, T.

Task

Re-arrange the input sequences given in a way such that the DRI is maximized, to improve the crops' DRI.

To calculate the DRI of the crop, we will be calculating according to the following rules:

1. A triple of A (AAA) will deduct the DRI by 10 points. So AAAAAA will reduce DRI by 20 points.
2. Every contiguous set of ACGT will increase the DRI by 15 points
3. A pair of CC will increase the DRI by 25 points.

   So CCCC will increase DRI by 50 points.
4. Combination pairs of overlapping sequences will not be awarded double points i.e. AAACGT will not qualify for -10 (AAA) and +15 (ACGT)

   Take note, in this case, AAA will be first used to qualify for -10 and it cannnot be used in the computation of ACGT. Thus, the score for AAACGT will be -10 instead of a total of +5 (-10 + 15)

The score of DRI starts from 0, and can be negative/positive. Of course, the more positive the DRI score, the better the crop.

Examples

Please note that the following examples are **NOT** optimal. They are meant to show DRI score calculations only.

Example 1

```
Input:  AAACCCAAAGGTTACTGAAAAG
Output: AAGAAGAAGAAGAATATTCCCC
```

Explanation for example 1 This output will give us a DRI score of 50.

How is the DRI score tabluated:

- There is no occurence of AAA in the output, therefore, the DRI score is not deducted.
- There is no ACGT in the output, therefore the DRI score is not increased.
- There are 2 occurences of CC in the output, therefore, the DRI score is increased by 50 points.

```
Output DRI Score : 0 x -20 + 0 x 15 + 2 x 25 = 50
```

Example 2

```
Input:  AAAAAACCTTTGGGGGGGTTTT
Output: AGAGAGAGAGAGCCTTTTTTTG
```

Explanation for example 2 This output will give us a DRI score of 25.

How is the DRI score tabluated:

- There is no occurence of AAA in the output, therefore, the DRI score is not deducted.
- There is no ACGT in the output, therefore the DRI score is not increased.
- There is 1 occurence of CC in the output, therefore, the DRI score is increased by 25 points.

```
Output DRI Score : 0 x -20 + 0 x 15 + 1 x 25 = 25
```

API

Please expose a **POST** endpoint /intelligent-farming for us to verify.

Additionally, please ensure that your response sets the HTTP header **Content-Type** to **application/json**. If this is not done, the evaluation call will fail.

Input and Output Schema

The input and output JSON schemas are exactly the same. Simply put, mutate only the gene sequence content and return the same data structure.

- Do **NOT** change the values of the *runId* and the *id* shown in the schema below, doing so will result in submission failure.
- Do **NOT** change the length of the *geneSequence*, doing so will result in submission failure.
- Do **NOT** change the proportion of genes in the gene sequences, doing so will result in submission failure.

  E.g. Input gene sequence = AAAA, and output gene sequence = ACGT
- There will be exactly 10 gene sequences given to you. You must submit back 10 gene sequences, while not changing the value of *id* as mentioned in point 1. Any fewer or any more gene sequences returned will result in submission failure.

```
{
    "runId": String <- Do NOT Change
    "list": [
        {
            "id": Integer, <- Do NOT Change
            "geneSequence": String <- Mutate this
        },
        ...
```

```
    ]
}
```

Sample Input and Output

```
{
    "runId": "a8098c1a-f86e-11da-bd1a-00112444be1e"
    "list":[
        { "id": 1, "geneSequence": "ACGT" },
        { "id": 2, "geneSequence": "ACGT" },
        { "id": 3, "geneSequence": "ACGT" },
        { "id": 4, "geneSequence": "ACGT" },
        { "id": 5, "geneSequence": "ACGT" },
        { "id": 6, "geneSequence": "ACGT" },
        { "id": 7, "geneSequence": "ACGT" },
        { "id": 8, "geneSequence": "ACGT" },
        { "id": 9, "geneSequence": "ACGT" },
        { "id": 10, "geneSequence": "ACGT" }
    ]
}
```

Contacts

- Tan Chee Wei
- Jerald Gan
- Louis Lim

Don't forget to save the Earth!

Magical Fruit Basket

Mafrba, the magical fruits seller is holding a competition to find the best estimator in town.

The challenge is to guess the exact weight for a given basket of fruits.

The closer you are, the more points you get!

In a magical fruit basket, there can be up to 3 kinds of fruits where it can contain 1 to 100 of each kind.

At the start of every hour, the weight for each kind of fruit is randomly generated, set between 1 to 100 grams, inclusive.

For example, for a magical fruit basket of 1 maApple, 2 maWatermelons, and 3 maBanana, the sample input is given below.

Sample Input

```
{ "maApple": 1, "maWatermelon": 2, "maBanana": 3 }
```

The expected output would be the guessed weight in grams, no decimal places.

So, if your guess is that maApple is 10g, maWatermelon is 20g, and maBanana is 30g.

Your final guess for the basket of fruits should be 1*10 + 2*20 + 3*30 = 140g

Sample Expected Output

```
140
```

After you have made your guess, Mafrba will place the basket on the magic weighing scale.

Then, you will be updated through telepathy(your group's dashboard), how far your guess was from the actual weight of the magical fruit basket rounded to the nearest 100 grams.

Your final guess for the basket of fruits should be 1*10 + 2*20 + 3*30 = 140g

For example:

If the actual weight of the magical fruit basket is 190g:

- Your guess is 140g, you will see that your guess is 100g away, rounded up from 50g.

- Your guess is 200g, you will see that your guess is 0g away, rounded down from 10g.

Do note that you will only get the full score if you guess the exact weight(whole number)

Requirements

Expose 1 POST endpoint /fruitbasket for evaluation

Social Distancing

Instructions

The Singapore government has trouble deciding the limit for the number of people per train in the circuit breaker! Help them to resolve this conundrum by solving this question:

*How many ways can **X number of people** sit in an MRT cabin with **Y number of seats** given that there is a **required safety distance of Z seats**?*

Note: The order of the people do not matter.

Expose a `POST` endpoint `/social_distancing` for us to verify!

Input

```
{
    "tests": {
        "0": {
```

```
            "seats": 8,
            "people": 3,
            "spaces": 1
        },
        "1": {
            "seats": 7,
            "people": 3,
            "spaces": 1
        },
        "2": {
            "seats": 6,
            "people": 2,
            "spaces": 2
        },
        ...
    }
}
```

Output Expected

```
{
    "answers": {
        "0": 20,
        "1": 10,
        "2": 6,
        ...
    }
}
```

Example

```
Given that seats = 6, people = 3, spaces = 1
Expected Answer: 4
```

Explanation

S denotes an empty seat and P denotes a person sitting on a seat

Note: The order of the people do not matter.

Ways	S1	S2	S3	S4
1	**P**	S	**P**	S
2	**P**	S	**P**	S
3	**P**	S	S	**P**
4	S	**P**	S	**P**

Actual number of test cases given may change.

Optimized Portfolio

Portfolio Hedging Optimization

Problem Statement

You would be given Portfolio Data and a set of Index Futures Data to determine, which Index Future contract should be used to hedge the Portfolio and reduce the risk in an optimized manner. Index Future with the lowest optimal hedge ratio should be used as Optimized hedged position for the portfolio. If there are multiple future positions with the same Optimal Hedge Ratio, it should further be filtered on the lowest Futures Prc volatility. If still there are multiple records, filter based on lowest number of futures contract, to find the single fit.

Formulae to be used:

1. Optimal Hedge Ratio = ρ X (σs / σp)

Where:

ρ = The correlation coefficient of the changes in the spot and futures prices

σs = Standard deviation of changes in the spot price ‘s’

σp = Standard deviation of changes in the futures price ‘f’

An optimal hedge ratio is an investment risk management ratio that determines the percentage of a hedging instrument, i.e., a hedging asset or liability that an investor should hedge. As the hedge ratio approaches a value closer to 1, the established position is said to be “fully hedged.”
On the other hand, as the hedge ratio approaches a value closer to 0, it is said to be an “unhedged” position. So, a value of 0.4 indicates that only 40% of Portfolio Value needs hedging.

1. Number of Futures Contract = Optimal Hedge Ratio * (Portfolio value)/ (Futures contract size)

where, Futures contract size = Futures Price * Notional Value of Portfolio

Input Description:

1. For the Portfolio, following will be given:
   Portfolio Name
   Value
   Underlying Spot Price volatility
2. For Index Future, following will be given:
   Index Future name
   Correlation Coefficients between Portfolio and the Index Future
   Volatility of the Index Future Price
   Price of the Index Future
   Notional Value of the portfolio per Futures Contract

Output Description:

- For the Given input Portfolio, find the Index Future, Optimal Hedge Ratio and Number of Futures Contract, which provides best hedging and lowest risk.
- Number of futures contract should use ROUND to the nearest whole number.
- Optimal Hedge ratio should use ROUND to 3 decimal places.

Requirements:

- Expose a POST method with endpoint /optimizedportfolio
- Return output result as json in the required format
- Algorithm used should be optimized and sufficiently fast to enumerate results in split seconds

Constraints:

- Write an optimized solution as total timeout for all the test cases in single evaluation is up to 30 secs.

Sample Input:

```
{
    "inputs": [
            {

"Portfolio":{"Name":"Portfolio_Unhedged","Value":200000000,"SpotPrcVo
l":0.75},
                "IndexFutures":
                [

{"Name":"Index_Fut_A","CoRelationCoefficient":0.75,
"FuturePrcVol":0.95, "IndexFuturePrice":20.5, "Notional": 100000},

{"Name":"Index_Fut_B","CoRelationCoefficient":0.25,
"FuturePrcVol":0.85, "IndexFuturePrice":15.5, "Notional": 100000}
                ]
            }
    ]
}
```

Sample Output:

```
{
    "outputs": [

{"HedgePositionName":"Index_Fut_B","OptimalHedgeRatio":0.221,
"NumFuturesContract":29}
    ]
}
```

NOTE:
This is a sample, input could be large data having multiple records. Accordingly, output too will have multiple values.

Pre-Tick

A trader is trying to predict accurately the future price of an instrument such as a share or forex in this period of volatility. To start off, the trader does not need to predict the far future but just if the instrument will increase or decrease in price for the next tick. For that to happen, the trader will need your help to build the model.

Objective

Build an endpoint */pre-tick* that will receive 2000 records of an instrument in chronological order.

The data provided will be in csv format with ',' as separator and '\n' as line delimiter.

There is no limit on the type of algorithm allowed for the model. Feel free to use machine learning if desired.

Predict the close price of the instrument in the next tick.

Sample Input

"Open,High,Low,Close,Volume

867.4,873.1,862.9,873.1,761.0

914.9,924.6,911.1,917.9,741.0"

Expected Output

"920.5"

The Great Olympiad of Babylon

Created By Moin and Ryan

Problem Statement

The city of Babylon holds a International Olympiad each year inviting intellectuals from all over the world. Arsenios hails from a small village in Macedonia and has dreamt since early childhood to be able to participate in the Olympiad, which consists of questions from every book known to mankind.

In the year 1300BC, Arsenios is finally able to make the journey to Babylon but he only has a limited number of days to prepare for the Olympiad. He would like to ask for your help in scheduling his daily reading in order to maximize his chances of winning.

As such Arsenios has figured out the amount of time he has each day up until the day of the Olympiad. He also knows the amount of time in minutes it will take him to read a particular book. He will provide you this information and would be grateful if you can help him design the optimal schedule. Since the Olympiad guarantees an equal number of questions from each book, Arsenios will be able to maximize his score simply by reading the most books he can read given the constraints. He would like to ask you to tell him how many books he would be able to read with the optimal schedule given that he has to read a book to completion on the same day that he starts reading it.

Problem Specifications

The number of books (`numberOfBooks`) would always be greater than the number of days available for preparation (`numberOfDays`). The time required to read book i where `0 <= i < numberOfBooks` to completion is going to be provided as $books_i$ where books is a list. The time available for reading on day j where `0 <= j < numberOfDays` is going to be provided as $days_j$ where days is a list.

The result should be the number of books Arsenios can read with the optimal schedule (`optimalNumberOfBooks`).

Endpoint: /olympiad-of-babylon

Input Limits

```
3 < numberOfDays <= 10
5 < numberOfBooks <= 50
80 <= days[i] <= 120 | For all i
20 <= books[i] <= 80 | For all i
```

Input Schema

```
{
    numberOfBooks: Integer,
    numberOfDays: Integer,
    books: [Integer],
    days: [Integer]
}
```

Output Schema

```
{
    optimalNumberOfBooks: Integer
}
```

Examples

Input

```
{
    numberOfBooks: 5,
    numberOfDays: 3,
    books: [114, 111, 41, 62, 64],
    days: [157, 136, 130]
}
```

Ouput

```
{
    optimalNumberOfBooks: 5
}
```

Potential Optimal Schedule

```
Day 1: Book 2 and Book 3 (111 minutes + 41 minutes = 152 minutes < 157 minutes)
Day 2: Book 4 and Book 5 (62 minutes + 64 minutes = 126 minutes < 136 minutes)
Day 3: Book 1 (114 minutes < 130 minutes)
```

Driverless Car

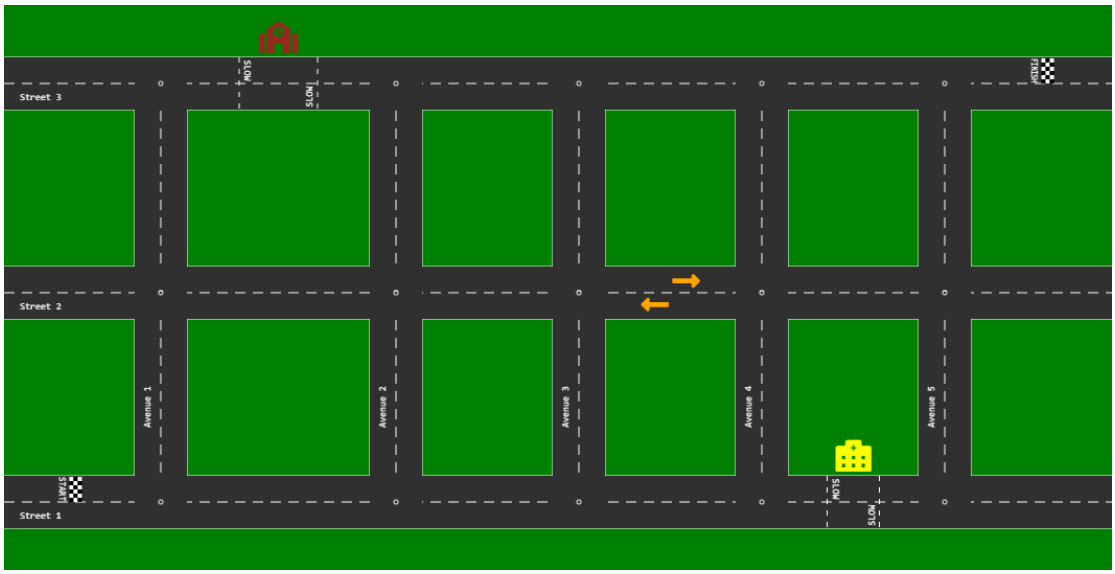
A small part of the town has been selected for testing of driverless vehicles. You will be in-charge of driving and keeping track of one of these vehicles.

Your goal is to navigate and drive your car from start to finish, following traffic rules, in the least amount of time.

There will be no other vehicles, pedestrians or obstacles on the roads. To navigate you need to give instructions to the vehicle. These instructions will be in the form of the direction, distance and acceleration/deceleration.

Map Example

The picture below is a representation of the town area map (not to scale). Start and Finish are indicated.



Semantics

Rules

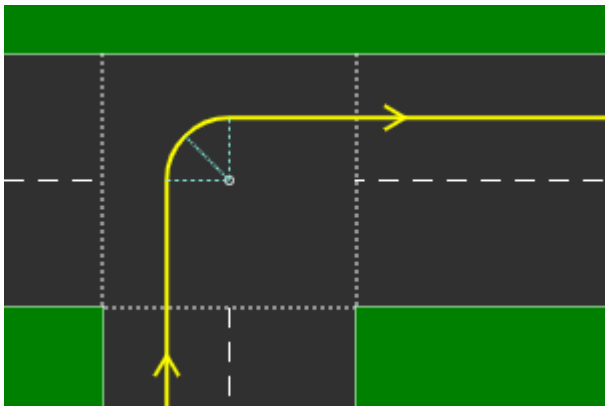
Standard traffic rules need to be followed at all times.

- Vehicles are driven on the left side of the road
- Vehicles need to maintain max speed allowed on the road or in that zone
- Vehicles can only move in the forward direction
- Vehicles are to be driven exactly in the middle of the lane
- Vehicles can't cross the road dividers
- Vehicles shouldn't slow down or stop at the finish line

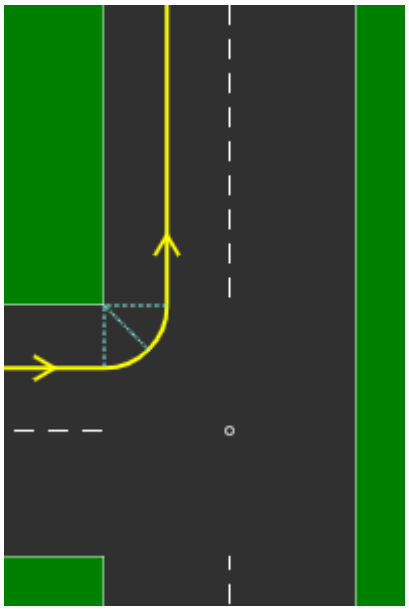
Intersection Rules

- To take a turn, vehicle needs to be in the middle of the circular path that will turn
- Vehicles can pass through intersections at a **top speed of 5m/s**

Right turn at intersection



Left turn at intersection



Location

Start and finish locations will be provided to you. Location is represented in co-ordinates (x, y) where x refers to the horizontal direction in meters and y refers to vertical direction in meters. Looking at the map example, x co-ordinate increases as you move from left to right, while y co-ordinate increases from bottom to top.

```
{ "x": 100, "y": 100 }
```

Roads

Roads are classified as streets and avenues. A road can be of type *street* or *avenue*. Streets run perpendicular to avenues. As seen on the map example, streets are horizontal and avenues are vertical.
A road can have zones (which can restrict top speed). Roads have exactly one lane to go in each direction (splitting road width equally). Vehicles drive on the left side of the road. All **roads are 10 meters** wide and each **lane is 5 meters**.

```
{
  "name": "S1",
  "type": "street",
  "from": {
    "x": 0,
    "y": 100
  },
  "to": {
    "x": 500,
    "y": 100
  },
  "maxSpeed": 20,
  "zones": [
    {
      "maxSpeed": 10,
      "from": {
        "x": 250,
        "y": 100
      },
      "to": {
        "x": 300,
        "y": 100
      }
    }
  ]
}
```

Vehicles

Vehicles have specifications like top speed, acceleration and deceleration (braking). The vehicle can be assumed to be at the location where its front is (meaning you can ignore vehicle's length, consider it as a point).

```
{
  "topSpeed": 25,
  "acceleration": 10,
  "deceleration": 15
}
```

Instruction

Instruction to the vehicle includes direction, distance and acceleration/deceleration. Acceleration should be +ve to increase speed, -ve to brake (decelerate), and zero to maintain speed. Direction can be *straight*, *left*, or *right*. Distance should be **truncated to 3 decimal places** - for example, distance of 5.12395 meters should be sent as *5.123*.

```
{
  "acceleration": 8,
  "distance": 5.925,
  "direction": "straight"
}
```

In order to turn, you would have to say something like *turn left for x meters at constant speed*. You would have to calculate the distance of the turn (remember to truncate to 3 decimal places).

```
{
  "acceleration": 0,
  "distance": 1.234,
  "direction": "left"
}
```

Units

- Location is specified in horizontal & vertical co-ordinates in m (meters)
- Speed is specified in m/s (meters per seconds)
- Acceleration is in m/s^2 (meters per second square)

- Deceleration is in m/s^2 (meters per second square)
- Distance is in m (meters)

Sample Input


```
{
  "gameId": "G495690bb",
  "roads": [
    {
      "name": "S1",
      "type": "street",
      "from": {
        "x": 0,
        "y": 100
      },
      "to": {
        "x": 5000,
        "y": 100
      },
      "maxSpeed": 20,
      "zones": [
      ]
    }
  ],
  "finish": {
    "x": 4800,
    "y": 102.5
  },
  "start": {
    "x": 0,
    "y": 102.5
  },
  "vehicle": {
    "acceleration": 10,
    "deceleration": 15,
    "topSpeed": 25
  }
}
```

Sample Output

```
{
  "gameId": "G495690bb",
  "instructions": [
    {
      "acceleration": 10,
      "distance": 20,
      "direction": "straight"
    },
    {
      "acceleration": 0,
      "distance": 4780,
      "direction": "straight"
    }
  ]
}
```

How

- Expose a REST endpoint "driverless-car" that accepts HTTP POST request, for example *example.com/driverless-car*
- The endpoint will be called with input json, it should accept content-type *application/json*
- There will be 10 games presented to you
- You have 30 seconds to respond to each call

Yin Yang

Background

Unbeknownst to most mortals, our world exists and thrives only because of the presence of cosmically generated Yin and Yang elements which powers its core.

While most physics teachers will have you believe that the Earth spins and

revolves around the Sun because of some ridiculous concept like Gravity, in reality, the Gods release an arbitrary number of Yin and Yang elements into the atmosphere which are captured by the Guardian of the Earth, Sage Ryan, who then uses his Qi to release these elements into the core. However, the balance of these elements is incredibly important, if one kind of the elements exceeds the other by even a slight amount, the world would collapse. (Dinosaurs found this out the hard way when Sage Ryan partied a bit too hard one night.)

Problem

Every morning, Sage Ryan's ancient machine, "Element Altar" absorbs some elements from the atmosphere.
Then, depending on the current balance of elements inside the core, Sage Ryan operates the Altar and picks the elements of the desired kind and releases them to the Earth's core.

However, there are some strange, divinely ordained rules which Sage Ryan must obey while operating the Altar and hence making this elements release work incredibly arduous:

- Once ***n*** elements of Yin (displayed as 'y') or Yang (displayed as 'Y') are absorbed, they will form a single row on the Elements altar. e.g. "yYyyY" -> [yin, Yang, yin, yin, Yang]
- Elements Altar then provides an integer ***k*** which is the number of elements Ryan must pick today.
- For each picking process ***i*** where $1 \leq i \leq k$
    i. Elements Altar will generate (uniformly and independently) a random integer ***x*** between 1 and ***n*** - ***i*** + 1 (inclusive)
    ii. Elements Altar will then let Ryan pick the x^{th} element **either from the left or right end of the row**, which will then reduce the number of elements in the row by one.

Suddenly, one day in the middle of the Yin season, the sky became dimmer, the weather became colder. Sage Ryan realized that the number of Yin elements in the core had catastrophically increased.
Therefore, starting from today, **he needs to release as many Yang elements**

as he possibly can to the Earth's core until this seasonal Yin crisis is resolved.

However, before the picking process can commence, for every combination of elements, Sage Ryan needs to know roughly how much Qi he would need to accumulate.
Unfortunately, Sage Ryan was not a very disciplined student and slept through most of his Statistics classes. Therefore, he is hiring you to calculate the **expected number of Yang elements** he will release given that he always makes the most optimal choice of picking elements which would maximise the total number of Yang elements picked after k steps.

Hint 1: Ryan's goal is to maximise the total number of Yang elements released to the core and hence, sometimes he would need to pick a Yin element (rather than Yang) and throw it away if he thinks it would give him a higher probability to pick more Yang elements later on. Therefore, while designing a solution, always picking a Yang element may not be the most accurate strategy.
Hint 2: Look at Expected Value

Endpoint

Provide a `POST` endpoint `/yin-yang` that given 1 set of input will return 1 set of output

Input

The `HTTP POST` request will come with a body of `Content-Type: application/json` .

```
{

    "number_of_elements" : n,

    "number_of_operations" : k,

    "elements" :  E,

}
```

Output

The expected `HTTP` response will come with a body of `Content-Type: application/json` containing

```
{

    result : (The expected number of Yang elements picked)

}
```

Constraints

- $1 \leq \boldsymbol{k} \leq \boldsymbol{n} \leq 30$
- HTTP Request Timeout: 5 seconds

Examples

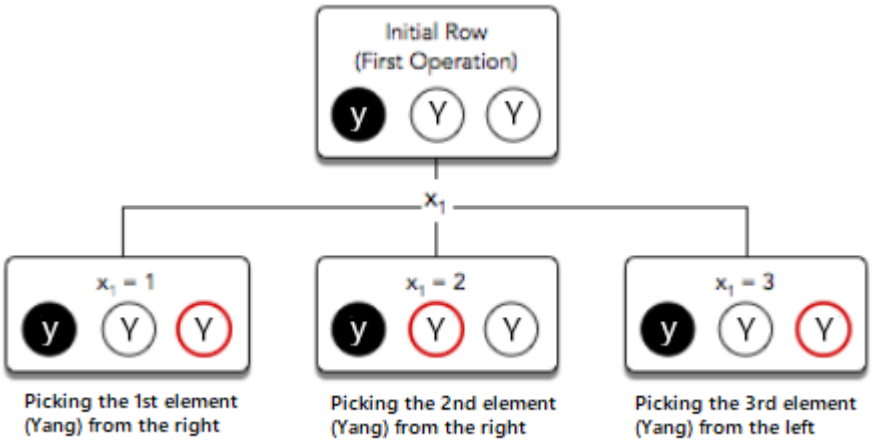
Sample Input 1

```
{

    "number_of_elements" : 3,

    "number_of_operations" : 1,

    "elements" :  "yYY"

}
```

Sample Output 1

```
1.0000000000
```

Explanation 1



Independent of ***x***, a Yang element will always be picked by Ryan, so, the expected number of Yang elements chosen after the ***k*** = 1 operation is 1. Since Ryan is only required to do one operation, the answer to this problem is 1.

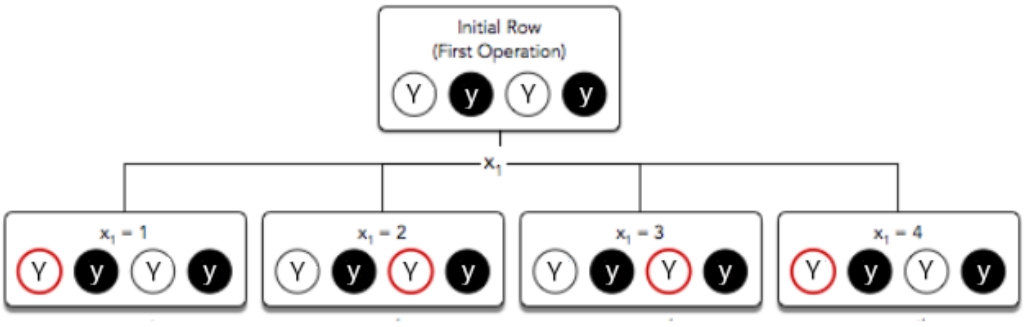
Sample Input 2

```
{

    "number_of_elements" : 4,

    "number_of_operations" : 2,

    "elements" :  "YyYy"

}
```

Sample Output 2

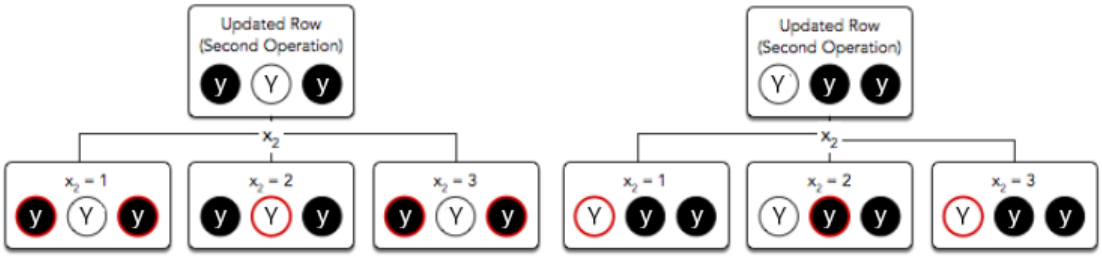
```
1.5000000000
```

Explanation 2



Independent of ***x***, a Yang element will always be picked by Ryan, so, the expected number of Yang elements chosen after the ***k*** = 1 operation is 1.

For the second operation, there are two possible orderings of Yin and Yang elements (depending on which Yang element was picked in the first operation):



In the first possible elements ordering, the probability of picking a Yang Element is ⅓. In the second possible elements ordering, the probability of picking a Yang element is ⅔. This means the expected number of Yang elements chosen in the second operation is ½ * ⅓ + ½ * ⅔ = ½. After performing both the operations, the total expected number of Yang elements is 1 + ½ = 1.5.

Bucket Fill

You are given a 2D graph of some water sources, buckets and pipes in SVG format. Water will flow vertically downwards from the water source(s) to the bottom of the SVG. Along its way down, it fills up the buckets or gets redirected by the pipes.

Your task is to calculate the total area of buckets that will be filled with water.

Requirement

- Expose a POST method with endpoint `/bucket-fill`.
- This endpoint should accept `Content-Type: image/svg+xml`.
- It should return a body with `Content-Type: application/json`.

Input

Note: There will be varying difficulties, the numbers indicated below is for the maximum difficulty.

- SVG will be in a maximum size of 2048x2048.
- Co-ordinates will always be in integers.
- Water is denoted as `<circle/>` in the SVG.
- A pipe or bucket is denoted as `<polyline/>` in the SVG.
- 2 ≤ Number of Buckets ≤ 64
- 1 ≤ Number of Pipes ≤ 64

Sample Input:

```
    <svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" width="1024" height="1024">
        <circle cx="500" cy="0" r="1" stroke="blue" fill="none" />
        <polyline fill="none" stroke="black" points="360,175 360,225
400,225 400,175" />
        <polyline fill="none" stroke="black" points="480,5 480,40 520,40
520,5" />
        <polyline fill="none" stroke="black" points="500,50 400,150" />
    </svg>
```

You may view above svg here.

Output

Expected json result should be in this format: `{ result: <THE_NUMERIC_ANSWER_HERE> }`.

Sample Output:

```
{
    result: 3315
}
```

Explanation:

- There is one water source at `(500, 0)`.
- There is a bucket bounded by the top left corner `(480, 5)` and bottom right corner `(520, 40)`, which has an area of `1365`.
- There is another bucket bounded by the top left corner `(360, 175)` and bottom right corner `(400, 225)`, which has an area of `1950`.
- Finally, there is a pipe that starts at `(500, 50)` and ends at `(400, 150)`.
- Since the water source will fill the first bucket, and then the second one with the help of the pipe, the answer for this 2D graph is `3315`.

Scoring Guideline

Your result will only be scored if it is within 50% or 200% of the expected answer. In other words, you will get some score if you get close enough.

Assumptions

- All elements in SVG will have a line thickness of 1px.
- There is at least one water source and either a bucket or pipe given in the SVG file. From the SVG perspective, that means there is at least one `<circle/>` and one `<polyline/>` element.
- Water sources have an infinite amount of water.
- Pipes are diagonal bounded by a square, and will always have a tangent of 1 or -1.
- For a pipe to redirect, the water stream must either hit along the length or the starting top.
- Water that enters from any segment of a pipe will come out below the last unit.
    - For example, a pipe ending at `(50, 50)` will redirect water to the point `(50, 51)`.
- A bucket may be inside another bucket such that they are either aligned at the top or one enclosing the other completely.
- No bucket will be partially contained in another bucket.
- Overflowing of buckets will happen when the bucket is filled up, and is only possible if its corner's above and diagonals are unobstructed.

    - Using the ASCII art below as an illustration, the middle bucket with water `~` will overflow as the points marked `@` are unobstructed, with water `.` flowing downwards.

```
x x     x x
xxx     xxx
  @@@   @@@
  .x~~~~x.
  .xxxxxx.
```

- If a stream hits a bucket's corner, the stream will fill the bucket and overflow it.
- No shapes will cross another.
    - A bucket may encroach into the square boundaries of a pipe, but not block the path.
    - A pipe will not enter the inside of a bucket, so it may only start or end at the top of the bucket.
- Do not include water that look like they might be trapped in between buckets or pipes, we only need the area of buckets that will be filled.
    - For example, the area marked `!` below should not be included, and should only count those marked in `~`.

```
 .     .
..... .....
.x~x!!!x~x.
.xxx!!!xxx.
. x~~~~~x .
. xxxxxxx .
```

Introduction to "swaps"

In investment banks we sell a lot of financial products. One type of financial product we sell is called "swaps". A swap allows a client to bet on an asset (e.g. stocks) without actually having to buy them. This is sometimes attractive to the client because the commission rates of buying a swap can be lower than buying the actual assets from an exchange.

Let's look at some examples.

If a client buys one unit of *long* swap on HSBC stock (0005.HK), after an agreed date, if HSBC's stock goes up $1, we need to pay our client $1. If the stock goes down $1, our client will need to pay us $1. This works as if the client is holding 1 unit of HSBC stock.

There are also *short* swaps, which is the opposite of long swaps. If a client buys one unit of *short* swap on HSBC stock (0005.HK), after an agreed date, if HSBC stock goes up $1, our client needs to pay us $1. If HSBC stock goes down $1, we need to pay our client $1. This works as if the client is holding -1 one unit of HSBC stock.

Background

When we sell swaps, we are exposed to market risks since we can potentially lose money to our clients depending on the market. As an investment bank, we only want to make money from commissions and we don't want to be affected by market volatility. As a result, we need to *hedge* our risk (i.e. to decrease our risk).

One way of hedging our risk is whenever a client buys HSBC swaps from us, we go to the market and buy actual HSBC stocks. In this way, no matter if the price goes up or down, we don't make/lose money. Instead, we just earn money from the commissions of selling swaps to our clients.

So an ideal case would be: Client buy or sell X amount of Swap -> We go to market to buy or sell the same amount of underlying stock -> we earn commission :D But then there are also some cases when a client buy a swap, and another client short a swap, then the client are flat and we are not in risk anyway. Or a client buy some swap, and sell them back in a very short time. There's no point of us buying stock and sell them at such short time, we can save the cost in the transaction and earn more money! So here we need to figure out given the current market situation and client behavior, when we should hedge? We will do some simulation for your strategy. The criteria will include the total Profit and Loss (PnL) and the variance of it as an estimation of total market risk exposure. Here you are the trader and will be given the client order, and the current bid and ask of the underlying, and finally respond with how much you want to hedge at this time.

Here are some setting:

Please expose an endpoint `/swaphedge` so that the server can ping and give incoming ticks. All input data and assumed return data are all in JSON format.

Every day has 30 time unit, so you will get 31 http request including the initialization. At the end of the day (EOD), both your position and client's position need to be zero, and the balance will be used to calculate PnL. Note that for each trade you make, you need to pay 0.1% of stamp duty, so frequent trading might not be the most ideal solution!

Evaluation:

As a sell side, we hope to be risk neutral, which mean regardless of the market condition, we hope to make constant profit from this business. Therefore the algorithm will be evaluated by the PnL compared to the client. The metrics will include both average and standard deviation of PnL difference. An algorithm that gives great PnL but having a huge volatility means it bring a lot of risks to the bank and hence might not be regarded as the optimal solution.

if through out the day, the client PnL is : [0, 1, 3] while our hedging PnL is [0, 1, 2], the mean and std wold be computed from the net PnL dataset : [0-0, 1-1, 2-3], which is equal to [0, 0, -1]. The mean and std of this array would be compared to that from the benchmark algorithm. Therefore, you not only want to make as much money as your client, but also avoid taking too much risk so that your portfolio is too volitile.

The request input format:

```
{
    'time': current unit time
    'bid': price to SELL a unit of the product
    'ask': price to BUY a unit of the product
    'accu_order': accumulated order from the client
    'our_position': our banks current holding
    'balance': our trader's cash balance
    'client_balance'': our client's temporary cash balance, to be
settled at EOD
    'order': the order the client put at this current time
```

```
    'run_id': the run_id of the evaluation assigned by the central
server
    'tradedate_id': used to identify the same evaluation round
}
```

Your respond format:

```
{
    'order': Int, the number of shares you want to buy, negative
number indicate a sell.
}
```

Case 1:

Case 1:

- The Start of the day initialization:

Evaluator input

```
{
    "time": 0,
    "bid": 100,
    "ask": 101,
    "accu_order": 0,
    "our_position": 0,
    "balance": 0,
    "client_balance": 0,
    "order": 0,
    "run_id": "test",
    "tradedate_id": "ff153900fbb511ea96e834cff6dfab7d"
}
```

User respond

```
{
    'order': 0
}
```

- The first tick coming

Evaluator input


```
{
    "time": 1,
    "bid": 100.0,
    "ask": 101.0,
    "accu_order": 33,
    "our_position": 0,
    "balance": 0,
    "client_balance": -3333.0,
    "order": 33,
    "run_id": "test",
    "tradedate_id": "ff153900fbb511ea96e834cff6dfab7d"
}
```


trader's potential respond


```
{
    'order': 33
}
```


Here the user hedge instanly, and we buy the stock at the same time our client has buy in. There's no risk left. Notice that you and client will buy at ask and sell at bid, so a frequent trading is prob not the optimal solution! Also you will pay 0.1% stamp duty for each orders.

- The second tick coming

Evaluator input


```
{
    "time": 2,
    "bid": 100.0,
    "ask": 101.0,
    "accu_order": 21,
    "our_position": 33,
    "balance": -3336.333,
    "client_balance": -2133.0,
    "order": -12,
    "run_id": "test",
```

```
    "tradedate_id": "ff153900fbb511ea96e834cff6dfab7d"
}
```


trader's potential respond


```
{
    'order': -12
}
```


We are selling since our client are flattening the position. Also note that you will pay 0.1% stamp duty when selling as well.

- The Third tick coming

Evaluator input


```
{
    "time": 3,
    "bid": 100.0,
    "ask": 101.0,
    "accu_order": -10,
    "our_position": 21,
    "balance": -2137.533,
    "client_balance": 967.0,
    "order": -31,
    "run_id": "test",
    "tradedate_id": "ff153900fbb511ea96e834cff6dfab7d"
}
```


trader's potential respond


```
{
    'order': -31
}
```


- The forth tick coming

Evaluator input


```
{
    "time": 4,
```

```
    "bid": 101.0,
    "ask": 102.0,
    "accu_order": -6.0,
    "our_position": -10,
    "balance": 959.3670000000001,
    "client_balance": 559.0,
    "order": 4.0,
    "run_id": "test",
    "tradedate_id": "ff153900fbb511ea96e834cff6dfab7d"
}
```


trader's potential respond


```
{
    'order': 4
}
```


- The fifth tick coming

Evaluator input


```
{
    "time": 5,
    "bid": 101.0,
    "ask": 102.0,
    "accu_order": -3.0,
    "our_position": -6,
    "balance": 550.9590000000001,
    "client_balance": 253.0,
    "order": 3.0,
    "run_id": "test",
    "tradedate_id": "ff153900fbb511ea96e834cff6dfab7d"
}
```


trader's potential respond


```
{
    'order': 3
}
```

Assuming there's no other trades happen in the day, at EOD everything need to be flattened to zero, so both of the 3 short shares of your's and client's would be buy back at ask price at EOD. Then the result will be compared to benchmark algorithm's result.