1. Son Goku wants to try new food

Son Goku wants to discover new kind of gastronomy and will choose a circular path to travel and discover new recipes. He needs to eat enough food to take him from one place to another. He knows that after traveling some distance, he can find a food source that will motive him to travel a further distance.

There are *n* food sources along the circular path numbered from *0* to *n-1*. Initially, Son Goku hasn't eat and can teleport to any food source and start his journey. Son Goku consumes units of food equal to the units of distance travelled. He needs to choose a point to start his journey that will allow him to complete his journey. Determine the lowest index of the starting points from which Son Goku can start his journey and visit all of the places in the circular path in order. If there is no solution, return -1.

For example, there are $n = 4$ sources of food along his route: $food = [3, 2, 5, 4]$ and $dist = [2, 3, 4, 2]$. The first attempt is starting at the first source, $food[0] = 3$. He teleports there without cost and eat *3* units of food. The distance to the next point is $dist[0] = 2$. It will need *2* units of food to get there and he eat $food[1] = 2$ units upon arrival, so he has consume $3 - 2 + 2 = 3$ units of food after making his first stop. Continuing along the journey:

- *3 - dist[1] + food[2] = 3 - 3 + 5 = 5*
- *5 - dist[2] + food[3] = 5 - 4 + 4 = 5*
- *5 - dist[3] = 5 - 2 = 3*

At this point, he is back to the first place. Because he can complete his journey starting at source *food[0]*, there is no reason to continue with the analysis so its index, *0,* is returned. To illustrate a point from the same example, if he starts at position *2*, where *food[1] = 2* and *dist[1] = 3*, he will not be able to proceed to the next point because the distance is greater than his food units. Note that the list is circular, so from *food[3]* in this example, the next source on the path is *food[0].*

Function Description

Complete the function *optimalPosition* in the editor below. The function must return an integer that denotes the minimum index of *food* from which he can start a successful journey. If no such starting point exists, return *-1.*

optimalPosition has the following parameter(s):

food[food[0],...food[n-1]]: an array of integers where *food[i]* denotes the amount of *food* in the i^{th} source.

dist[dist[0],...dist[n-1]]: an array of integers where *dist[i]* denotes the distance to the next *food* source.

Constraints

- $1 \le n \le 100000$
- $0 \le food[i] \le 10000$
- $0 \le dist[i] \le 10000$

▼ Input Format For Custom Testing

The first line contains an integer, *n*, that denotes the number of elements in *food*.

Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains an integer that describes *food[i]*.

The next line again contains the integer, *n*, that denotes the number of elements in *dist*.

Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains an integer that describes *dist[i]*.

▼ Sample Case 0

Sample Input For Custom Testing

```
4
2
4
5
2
4
4
3
1
3
```

Sample Output

```
1
```

Explanation

Here *food = [2, 4, 5, 2]* and *dist = [4, 3, 1, 3].* If Son Goku starts at the second *food* source, his *food* levels are:

- *food[1] = 4*
- *4 - dist[1] + food[2] = 4 - 3 + 5 = 6*
- *6 - dist[2] + food[3] = 6 - 1 + 2 = 7*
- *7 - dist[3] + food[0] = 7 - 3 + 2 = 6*
- *6 - dist[0] = 6 - 4 = 2.*

The first point from where Son Goku can start his journey is the 2^{nd} *food* source. The output should be *1*, the index of the 2^{nd} location.

▼ Sample Case 1

Sample Input For Custom Testing

```
4
8
4
1
9
4
10
9
3
5
```

Sample Output

```
-1
```

Explanation

Here *food = [8, 4, 1, 9]* and *dist = [10, 9, 3, 5].* In each case except *food[3] = 9*, the distance to the next source is greater than the amount of *food* at the current source.

food[3] = 9

9 - dist[3] + food[0] = 9 - 5 + 8 = 12

12 - dist[0] + food[1] = 12 - 10 + 4 = 6

6 - dist[1] + food[2] = 6 - 9 + 1 = -2

No matter where Son Goku starts, he will not be able to finish his travel.