Some people run along a straight line in the same direction. They start simultaneously at pairwise distinct positions and run with constant speed (which may differ from person to person).

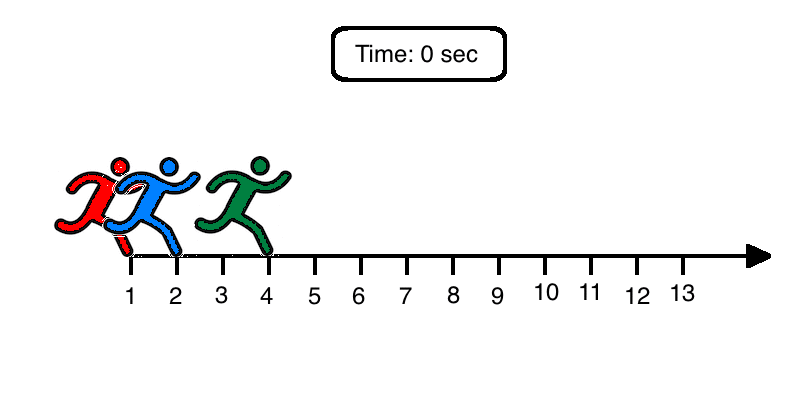
If two or more people are at the same point at some moment we call that a meeting. The number of people gathered at the same point is called meeting cardinality.

For the given starting positions and speeds of runners find the maximum meeting cardinality assuming that people run infinitely long. If there will be no meetings, return -1 instead.

**Example**

For startPosition = [1, 4, 2] and speed = [27, 18, 24], the output should be  
runnersMeetings(startPosition, speed) = 3.

In 20 seconds after the runners start running, they end up at the same point. Check out the gif below for better understanding:



**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] array.integer startPosition**

A non-empty array of integers representing starting positions of runners (in meters).

*Constraints:*  
2 ≤ startPosition.length ≤ 10,  
-1000 ≤ startPosition[i] ≤ 1000.

* **[input] array.integer speed**

Array of positive integers of the same length as startPosition representing speeds of the runners (in meters per minute).

*Constraints:*  
speed.length = startPosition.length,  
1 ≤ speed[i] ≤ 30.

* **[output] integer**

The maximum meeting cardinality or -1 if there will be no meetings.

<https://codefights.com/arcade/code-arcade/spring-of-integration/Hb9Cppx4pCY4d8J5P>

<https://codefightssolver.wordpress.com/2016/11/03/runners-meetings/>

static int runnersMeetings(int[] startPosition, int[] speed)

{

int result = 1;

for (int i = 0; i < startPosition.Length; i++)

{

for (int j = i + 1; j < startPosition.Length; j++)

{

int distDiff = startPosition[j] - startPosition[i],

speedDiff = speed[i] - speed[j],

cnt = 0;

if (speedDiff == 0 && distDiff != 0)

continue;

for (int k = 0; k < startPosition.Length; k++)

{

if (startPosition[k] \* speedDiff + speed[k] \* distDiff

== startPosition[i] \* speedDiff + speed[i] \* distDiff)

{

cnt++;

}

}

if (cnt == 0)

{

continue;

}

if (cnt > result)

{

result = cnt;

}

}

}

if (result < 2)

result = -1;

return result;

}