1. Astronomy Board Game

In an astronomy board game, N planets in an imaginary universe do not follow the normal law of gravitation. All the planets are positioned in a row.
The planetary system can be in a stable state only if the sum of the mass of all planets at even positions is equal to the sum of the mass of planets at the odd positions.

Initially, the system is not stable, but a player can destroy one planet to make it stable. Find the planet that should be destroyed to make the system stable. If no such planet exists, then return -1. If there are multiple such planets, then destroy the planet with the smallest index and return the index of the destroyed planet.

Example

Let *N=5* and *planets = [2,4,6,3,4]*. Destroying the fourth planet of mass 3 will result in *planets = [2,4,6,4]*, and here, the sum of odd positioned planets is (2+6)=8, and the sum of even positioned planets is (4+4)=8, and both are equal now. Hence, we destroy the fourth planet.

Function Description

Complete the function *getPlanetToDestroy* in the editor below.

getPlanetToDestroy has the following parameter(s):
planets[planets[1],...planets[n]]: An array of integers

Returns

int: the index of the planet to be destroyed.

3. Number of Moves

Given a chess board of n rows (top to bottom) and n columns (left to right).
In each move, a knight moves either:

- *2* column positions and 1 row position
- 2 row positions and 1 column position

In other words, a move is 2 steps along one axis and *1* step along a perpendicular axis.

Given a starting position A and ending position B, calculate the minimum number of moves needed by the knight to move from A to B if it is possible. If it is not possible, return -1. All moves must remain within the chess board.

Example

n = 9

startRow = 4

startCol = 4

endRow = 4

endCol = 8

The chess board has a size of *9 x 9.*

- Starts at the position *(startRow, startCol) = (4, 4).*
- Move 1 step up or down, then 2 steps right to reach either the position (3, 6) or (5,6).
- Move 2 steps right and 1 step down or up as necessary to reach the position (4,8).
- The minimum number of moves to move from the position

ALL

1

2

3

4

5

6

7

8

9

2. Selecting Stocks

An investor has saved some money and wants to invest in the stock market. There are a number of stocks to choose from, and they want to buy at most *1* share in any company. The total invested cannot exceed the funds available. A friend who is a stock market expert has predicted the values of each stock after 1 year. Determine the maximum profit that can be earned at the end of the year assuming the predictions come true.

Example

saving = 250

currentValue = [175, 133, 109, 210, 97]

futureValue = [200, 125, 128, 228, 133]

To maximize profits, the investor should buy stocks at indices *2* and *4* for an investment of *109 + 97 = 206*. At the end of the year the stocks are sold for *128 + 133 = 261*, so total profit is *261 - 206 = 55*.

Function Description

Complete the function *selectStock* in the editor below. The function should return an integer that denotes the maximum profit after one year.

selectStock has the following parameter(s):

int *saving:* amount available for investment

int *currentValue[n]:* the current stock values

int *futureValue[n]:* the values of the stocks after one year

Constraints