'''

Mr Ibrahim and Kumar are playing a game of luck.

There is a treasure in front of them, the treasure is on top of N bricks,

in order to win the treasure, the rules are as follows:

- Each turn, the player remove K bricks from the bottom.

- where K is, 0 < K < N.

- and N should be perfectly divisible by K.

The player who cannot remove any more bricks, is called unlucky and the other

player wins the treasure. It is said that Ibrahim will start the game always.

Your task is to print 'true' if Ibrahim wins the treasure, otherwise 'false'.

Input Format:

-------------

An integer N, number of bricks.

Output Format:

--------------

Print a boolean value.

Sample Input-1:

---------------

3

Sample Output-1:

----------------

false

Explanation:

------------

Ibrahim removes 1 brick, Kumar removes 1 brick,

Now, Ibrahim cannot remove any more brick.

Sample Input-2:

---------------

2

Sample Output-2:

----------------

true

Explanation:

------------

Ibrahim removes 1 brick, Kumar cannot removes the brick.

Write your python code below

'''

n=int(input())

if(n%2==0):

print(True)

else:

print(False)

A Train has to go through N railway-stations, the stations are in a cyclic route.

Each railway-station 'k' has coal[k].

The Train has unlimited coal tank and it consumes cost[i] of coal to travel

from railway-station i to its next railway-station (i+1).

The Train starts with an empty coal tank at one of the railway-stations.

Return the starting railway station's index,

If you can travel through all the railway-stations and

reaches the starting-station in the clockwise direction,

otherwise return -1.

Note:

If there exists a solution, it is guaranteed to be unique.

Both input arrays are non-empty and have the same length.

Each element in the input arrays is a non-negative integer.

Input Format:

-------------

Line-1 -> Space separated integers coal[i].

Line-2 -> Space separated integers cost[i].

Output Format:

--------------

Print the station number if possible,

Otherwise print -1.

Sample Input-1:

---------------

1 2 3 4 5

3 4 5 1 2

Sample Output-1:

----------------

3

Explanation:

------------

Start at station 3 (index 3) and fill up with 4 unit of coal. Your tank = 0 + 4 = 4

Travel to station 4. Your tank = 4 - 1 + 5 = 8

Travel to station 0. Your tank = 8 - 2 + 1 = 7

Travel to station 1. Your tank = 7 - 3 + 2 = 6

Travel to station 2. Your tank = 6 - 4 + 3 = 5

Travel to station 3. The cost is 5. The coal is just enough to travel back to station 3.

Therefore, return 3 as the starting index.

Sample Input-1:

---------------

2 3 4

3 4 3

Sample Output-1:

----------------

-1

Explanation:

------------

You can't start at station 0 or 1, as there is not enough coal to travel to the next station.

Let's start at station 2 and fill up with 4 unit of coal. Your tank = 0 + 4 = 4

Travel to station 0. Your tank = 4 - 3 + 2 = 3

Travel to station 1. Your tank = 3 - 3 + 3 = 3

You cannot travel back to station 2, as it requires 4 unit of coal but you only have 3.

Therefore, you can't travel around all the stations and reach the starting staton once

no matter where you start

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String[] coals=sc.nextLine().split(" ");

String[] costs=sc.nextLine().split(" ");

int[] coal=new int[coals.length];

int[] cost=new int[costs.length];

for(int i=0;i<coals.length;i++){

coal[i]=Integer.parseInt(coals[i]);

cost[i]=Integer.parseInt(costs[i]);

}

System.out.println(fun(coal,cost,coal.length-1));

}

public static int fun(int[] coal,int[] cost,int n){

for(int i=0;i<n;i++){

int cos=0;

cos+=coal[i];

if(cos<cost[i]){

continue;

}

else{

int count=0;

int m=i;

cos-=cost[i];

m+=1;

while((m%(n+1))!=i){

cos+=coal[m%(n+1)];

if(cos>=cost[m%(n+1)]){

cos-=cost[m%(n+1)];

count+=1;

}

else{

break;

}

m+=1;

}

if(count==n){

return i;

}

}

}

return -1;

}

}

