'''

Bablu is working in a construction field.

He has N number of building blocks, where the height and width of all the blocks are same.

And the length of each block is given in an array, blocks[].

Bablu is planned to build a wall in the form of a square.

The rules to cunstruct the wall are as follows:

- He should use all the building blocks.

- He should not break any building block, but you can attach them with other.

- Each building-block must be used only once.

Your task is to check whether Bablu can cunstruct the wall as a square

with the given rules or not. If possible, print true. Otherwise, print false.

Input Format:

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

Line-1: An integer N, number of BuildingBlocks.

Line-2: N space separated integers, length of each block.

Output Format:

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

Print a boolean value.

Sample Input-1:

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

6

1 2 6 4 5 6

Sample Output-1:

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

true

Sample Input-2:

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

6

5 3 2 5 5 6

Sample Output-2:

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

false

'''

n=int(input())

l=list(map(int,input().split()))

s=0

for i in l:

s+=i

if(n<4):

print("false")

else:

if(s%4!=0):

print("false")

else:

print("true")

Hulk and Thor become best friends now.

Hulk lives in the P-th house while Thor lives in the N-th house.

Hulk has decided to meet Thor every day. After going for many days,

he notices that there are few streets such that he passes every time

he goes to meet Thor, no matter which path he takes.

You have been given a grid of N houses in the form of undirected graph.

Houses are numbered from 1 to N.

For each I-th query, you have been given P (Hulk's house).

You have to say how many streets are there which he encounters in every path

he takes to go from P to N. If there is no such street, print "-1".

Also, note P can be equal to N.

Input Format:

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

Line-1: Two space separated integers N and M, Number of houses, number of streets.

Next M lines: contain two space-separated integers u and v,

denoting a street between house-u and house-v.

Next line: contains an integer P.

Output Format:

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

Print an integer, number of streets in that path.

Sample Input-1:

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

4 3

1 2

1 3

2 4

1

Sample Output-1:

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

2

Explanation:

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

There is only one path 1->2->4. so answer = 2

Sample Input-2:

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

4 3

1 2

1 3

2 4

4

Sample Output-2:

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

-1

Explanation:

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

There is no path as they both live in the same house 4.

So answer = -1.

In a game, there are some questions in a row, and each question has two options

'A' or 'B'. The organizer wants to confuse the participants by maximizing the

number of consecutive questions with the same option as valid answer.

(multiple As or multiple Bs in a row).

The organizer is given a word 'options' and an integer P, where options[i] is

the valid answer to the ith question and P is the maximum number of times

you may perform the following operation:

- Change the options value for any question to 'A' or 'B'

(i.e., set options[i] to 'A' or 'B').

Your task is to return the maximum number of consecutive As or Bs in the options,

after performing the operation at most P times.

Input Format:

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

Line-1 -> A word, the options

Line-2 -> An integer P, P number of operations.

Output Format:

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

Print an integer as output.

Sample Input-1:

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

ABBABA

2

Sample Output-1:

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

5

Explanation:

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

We can replace any two 'A's at with 'B's. There are five consecutive 'B's.

Sample Input-2:

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

BABBABABAA

1

Sample Output-2:

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

4

Sample Input-3:

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

BABBABABAA

2

Sample Output-3:

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

6

#include<bits/stdc++.h>

using namespace std;

int main(){

string s;

int n;

cin>>s>>n;

int i=0,i1=0,j=0,j1=0,k=0,k1=0;

int r=0,r1=0;

while(j<s.length()){

if(s[j]=='A'){

k++;

}

else{

k1++;

}

while(min(k,k1)>n){

if(s[i]=='A'){

k--;

}

else{

k1--;

}

i++;

}

r=max(r,j-i+1);

j++;

}

// ---------------

// BABBABABAA

// while(j1<s.length()){

// if(s[j1]=='B'){

// k1++;

// }

// while(k1>n){

// if(s[i1]!='B'){k1--;

// }

// i1++;

// }

// r1=max(r1,j1-i1+1);

// j1++;

// }

cout<<r;

}

In an online magazine, every day they will ask their readers a puzzle.

The puzzle has to be solved using the following rule set:

- The puzzle is an 9\*9 matrix, and consist of 9 -> 3\*3 sub-matrices.

- Each row in the puzzle should be filled with 1-9digits with out repetition

- Each column in the puzzle should be filled with 1-9digits with out repetition.

- Each 3\*3 sub-matrix should be filled with 1-9 digits with out repetition.

You will be given the puzzle in the form of 9 strings, each string length is 9.

Your task is check whether the puzzle can be solved or not.

If it is possible to solve, print "true". Otherwise print "false".

Input Format:

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

9 Lines: Each line consists of a string length-9, consist of digits[1-9] and '.' only

Output Format:

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

Print a boolean value.

Sample Input-1:

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

53..7....

6..195...

.98....6.

8...6...3

4..8.3..2

7...2...6

.6....28.

...419..5

....8..79

Sample Output-1:

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

true

Sample Input-2:

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

83..7....

6..195...

.98....1.

8...6...3

4..8.3..1

7...2...6

.6....28.

...419..5

....8..79

Sample Output-2:

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

false

#include <iostream>

#include <vector>

using namespace std;

bool isValid(vector < vector < char >> & board, int row, int col, char c) {

for (int i = 0; i < 9; i++) {

if (board[i][col] == c)

return false;

if (board[row][i] == c)

return false;

if (board[3 \* (row / 3) + i / 3][3 \* (col / 3) + i % 3] == c)

return false;

}

return true;

}

bool solveSudoku(vector < vector < char >> & board) {

for (int i = 0; i < board.size(); i++) {

for (int j = 0; j < board[0].size(); j++) {

if (board[i][j] == '.') {

for (char c = '1'; c <= '9'; c++) {

if (isValid(board, i, j, c)) {

board[i][j] = c;

if (solveSudoku(board))

return true;

else

board[i][j] = '.';

}

}

return false;

}

}

}

return true;

}

int main(){

vector<vector<char>> l(9,vector<char>(9));

for(int i=0;i<9;i++){

for(int j=0;j<9;j++){

cin>>l[i][j];

}

}

if(solveSudoku(l)){

cout<<"true";

}

else{

cout<<"false";

}

// cout<<solveSudoku(l);

}