Fair Rations

You are the benevolent ruler of Rankhacker Castle, and today you're distributing bread. Your subjects are in a line, and some of them already have some loaves. Times are hard and your castle's food stocks are dwindling, so you must distribute *as few loaves as possible* according to the following rules:

1. Every time you give a loaf of bread to some person i, you must also give a loaf of bread to the person immediately in front of or behind them in the line (i.e., persons $i+1$ or $i-1$).
2. After all the bread is distributed, each person must have an *even number* of loaves.

Given the number of loaves already held by each citizen, find and print the minimum number of loaves you must distribute to satisfy the two rules above. If this is not possible, print `NO`.

Example

$B = [4, 5, 6, 7]$

- We can first give a loaf to $i = 3$ and $i = 4$ so $B = [4, 5, 7, 8]$.
- Next we give a loaf to $i = 2$ and $i = 3$ and have $B = [4, 6, 8, 8]$ which satisfies our conditions.

All of the counts are now even numbers. We had to distribute 4 loaves.

Function Description

Complete the *fairRations* function in the editor below.

fairRations has the following parameter(s):

- *int B[N]:* the numbers of loaves each persons starts with

Returns

- *string:* the minimum number of loaves required, cast as a string, or 'NO'

Input Format

The first line contains an integer N, the number of subjects in the bread line.

The second line contains N space-separated integers $B[i]$.

Constraints

- $2 \leq N \leq 1000$
- $1 \leq B[i] \leq 10$, where $1 \leq i \leq N$

Output Format

Sample Input 0

```
STDIN       Function
-----       --------
5           B[] size N = 5
2 3 4 5 6   B = [2, 3, 4, 5, 6]
```

Sample Output 0

```
4
```

Explanation 0

The initial distribution is $(2, 3, 4, 5, 6)$. The requirements can be met as follows:

1. Give 1 loaf of bread each to the second and third people so that the distribution becomes $(2, 4, 5, 5, 6)$.
2. Give 1 loaf of bread each to the third and fourth people so that the distribution becomes $(2, 4, 6, 6, 6)$.

Each of the N subjects has an even number of loaves after 4 loaves were distributed.

Sample Input 1

```
2
1 2
```

Sample Output 1

```
NO
```

Explanation 1

The initial distribution is $(1, 2)$. As there are only 2 people in the line, any time you give one person a loaf you must always give the other person a loaf. Because the first person has an odd number of loaves and the second person has an even number of loaves, no amount of distributed loaves will ever result in both subjects having an even number of loaves.

Submissions: 25
Max Score: 25
Difficulty: Easy

Rate This Challenge:

More

Java 7

```java
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.regex.*;

class Result {

    /*
     * Complete the 'fairRations' function below.
     *
     * The function is expected to return a STRING.
     * The function accepts INTEGER_ARRAY B as parameter.
     */

    public static String fairRations(List<Integer> B) {
    // Write your code here
        int n = B.size();
        int count = 0;
        for(int i=0;i<n-1;i++){
            if(B.get(i)%2 != 0){
                B.set(i, B.get(i)+1);
                B.set(i+1, B.get(i+1)+1);
```

```
                count += 2;
            }
        }
        for(int i=0;i<n;i++){
            if(B.get(i)%2 != 0)
                return "NO";
        }
        return String.valueOf(count);
    }

}

public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(System.getenv("OUTPUT_PATH")));

        int N = Integer.parseInt(bufferedReader.readLine().trim());

        String[] BTemp = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

        List<Integer> B = new ArrayList<>();

        for (int i = 0; i < N; i++) {
            int BItem = Integer.parseInt(BTemp[i]);
            B.add(BItem);
        }

        String result = Result.fairRations(B);

        bufferedWriter.write(result);
        bufferedWriter.newLine();

        bufferedReader.close();
        bufferedWriter.close();
    }
}

```

Line: 34 Col: 6

Upload Code as File Test against custom input Run Code Submit Code