1195：判断整除

题目描述

一个给定的正整数序列，在每个数之前都插入+号或-号后计算它们的和。比如序列：1、2、4共有8种可能的序列：

(+1) + (+2) + (+4) = 7
(+1) + (+2) + (-4) = -1
(+1) + (-2) + (+4) = 3
(+1) + (-2) + (-4) = -5
(-1) + (+2) + (+4) = 5
(-1) + (+2) + (-4) = -3
(-1) + (-2) + (+4) = 1
(-1) + (-2) + (-4) = -7

所有结果中至少有一个可被整数k整除，我们则称此正整数序列可被k整除。例如上述序列可以被3、5、7整除，而不能被2、4、6、8……整除。注意：0、-3、-6、-9……都可以认为是3的倍数。

输入

输入的第一行包含两个数：N(2<N<10000)和k(2<k<100)，其中N代表一共有N个数，k代表被除数。第二行给出序列中的N个整数，这些整数的取值范围都0到10000之间（可能重复）。

输出

如果此正整数序列可被k整除，则输出YES，否则输出NO。（注意：都是大写字母）

输入样例

3 2
1 2 4

输出样例

NO

解析

观察样例，共有3个数字{1,2,4}，除数为2，则余数的范围即为{0,1}

首先，我们开始计算1%2，很明显答案为1。我们用f[i][j]表示前i个数字能够取到余数j，很明显：

f[1][1] = TRUE
f[1][0] = FALSE

索引	数值	取余结果:j	
		0	1
1	1	FALSE	TRUE
2	2		
3	4		

接下来，我们需要把第二个数字2入队，即在前一个数字对2的取余结果上进行加减2的计算。

1对2取余的结果只有1种，即数字1。因此，我们需要计算：(1+2) % 2和(1-2) % 2。

很明显：(1+2) % 2 = 1，我们标记f[2][1] = TRUE

另一方面，因为(1-2) % 2的结果是负数，为了避免出现负数，根据同余原理，我们可以在括号内加上除数2，再将减数对2取余，即将原式变成(1-2%2+2) % 2，结果仍然是1，我们标记f[2][1] = TRUE，则有下图：

索引	数值	取余结果:j	
		0	1
1	1	FALSE	TRUE
2	2	FALSE	TRUE
3	4		

我们将4入队，继续运算。因为数字2入队后，余数只有等于1这一种情况，因此，我们只需要计算(1+4) % 2 和 (1-4) % 2这两个式子。同样，为了避免出现余数为负的情况，我们需要将第二个式子改变成为(1-4%2+2) % 2。很明显，两个结果都是1，我们记作f[3][1] = TRUE，填入下表。

索引	数值	取余结果:j	
		0	1
1	1	FALSE	TRUE
2	2	FALSE	TRUE
3	4	FALSE	TRUE

其实，算到这里，我们已经看到结果了，因为f[3][0]为假，也就意味着1,2,4这三个数字无论如何计算都不能被2整除。

现在，我们把除数改成3，看看会发生什么？

首先，除数为3时，余数的范围为{0,1,2}，构建表格如下：

索引	数值	取余结果:j		
		0	1	2
1	1			
2	2			
3	4			

1、1%3 = 1，即f[1][1] = TRUE，填入下表

索引	数值	取余结果:j		
		0	1	2
1	1	FALSE	TRUE	FALSE
2	2			
3	4			

2、在前一个余数的基础上继续计算，即计算：
(1 + 2) % 3 = 0，f[2][0] = TRUE
(1 - 2%3 + 3) % 3 = 2，f[2][2] = TRUE
如下图所示：

索引	数值	取余结果:j		
		0	1	2
1	1	FALSE	TRUE	FALSE
2	2	TRUE	FALSE	TRUE
3	4			

3、现在，前面的计算中产生了两个余数，分别是0和2。因此，我们需要在这两个余数的基础上分别加减第三个数字4，即有下面四个式子：

(0 + 4) % 3 = 1，f[3][1] = TRUE

(0 - 4%3 + 3) % 3 = 1，f[3][1] = TRUE

(2+4) % 3 = 0，f[3][0] = TRUE

(2 - 4 % 3 + 3) % 3 = 1，f[3][1] = TRUE

索引	数值	取余结果:j		
		0	1	2
1	1	FALSE	TRUE	FALSE
2	2	TRUE	FALSE	TRUE
3	4	TRUE	TRUE	FALSE

我们可以看到f[3][0]为真，即代表{1,2,4}的组合能够被3整除。

编码

```
#include<bits/stdc++.h>

using namespace std;
const int maxN = 10005;
int n, k;  //n个数字，余数k
bool f[maxN][101];//f[i][j]表示前i个数的和被k整除的余数是否为j,j<k
int a[maxN];

void fn() {
    //遍历数组中的全部数据
    for (int i = 0; i < n; ++i) {
        //遍历K的全部余数
```

```
        for (int j = 0; j < k; ++j) {
            //前i-1个数字中存在余数为j的情况
            if (f[i-1][j]) {
                //标记加上a[i]后对k取余的结果
                f[i][(j + a[i]) % k] = true;
                //标记减去a[i]后对k取余的结果，注意将结果转正
                f[i][(j - a[i] % k + k) % k] = true;
            }
        }
    }
}

int main() {
    memset(f, false, sizeof(f));
    scanf("%d %d", &n, &k);
    for (int i = 0; i < n; ++i) {
        scanf("%d", &a[i]);
    }
    //将首个数字的取余结果标记为真
    f[0][a[0] % k] = true;
    fn();
    //判断是否能够整除
    if (f[n - 1][0]) {
        printf("YES");
    } else {
        printf("NO");
    }
    return 0;
}
```