E. Multithreading

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given the following concurrent program. There are N processes and the i-th process has the following pseudocode:

```
repeat n_i times
    y_i := y
    y := y_i + 1
end repeat
```

Here y is a shared variable. Everything else is local for the process. All actions on a given row are atomic, i.e. when the process starts executing a row it is never interrupted. Beyond that all interleavings are possible, i.e. every process that has yet work to do can be granted the rights to execute its next row. In the beginning $y = 0$. You will be given an integer W and n_i, for $i = 1, \ldots, N$. Determine if it is possible that after all processes terminate, $y = W$, and if it is possible output an arbitrary schedule that will produce this final value.

Input

In the first line of the input you will be given two space separated integers N ($1 \le N \le 100$) and W ($-10^9 \le W \le 10^9$). In the second line there are N space separated integers n_i ($1 \le n_i \le 1000$).

Output

On the first line of the output write `Yes` if it is possible that at the end $y = W$, or `No` otherwise. If the answer is `No` then there is no second line, but if the answer is `Yes`, then on the second line output a space separated list of integers representing some schedule that leads to the desired result. For more information see note.

Examples

input
1 10 11
output
No

input
2 3 4 4
output
Yes 1 1 2 1 2 2 2 2 2 1 2 1 1 1 1 2

input
3 6 1 2 3
output
Yes 1 1 2 2 2 2 3 3 3 3 3 3

Note

For simplicity, assume that there is no repeat statement in the code of the processes, but the code from the loop is written the correct amount of times. The processes are numbered starting from 1. The list of integers represent which process works on its next instruction at a given step. For example, consider the schedule `1 2 2 1 3`. First process `1` executes its first instruction, then process `2` executes its first two instructions, after that process `1` executes its second instruction, and finally process `3` executes its first instruction. The list must consists of exactly $2 \cdot \Sigma_{i=1 \ldots N} n_i$ numbers.