E. Arthur and Brackets

time limit per test: 2 seconds

memory limit per test: 128 megabytes

input: standard input

output: standard output

Notice that the memory limit is non-standard.

Recently Arthur and Sasha have studied correct bracket sequences. Arthur understood this topic perfectly and become so amazed about correct bracket sequences, so he even got himself a favorite correct bracket sequence of length $2n$. Unlike Arthur, Sasha understood the topic very badly, and broke Arthur's favorite correct bracket sequence just to spite him.

All Arthur remembers about his favorite sequence is for each opening parenthesis ('`(`') the approximate distance to the corresponding closing one ('`)`'). For the i-th opening bracket he remembers the segment $[l_i, r_i]$, containing the distance to the corresponding closing bracket.

Formally speaking, for the i-th opening bracket (in order from left to right) we know that the difference of its position and the position of the corresponding closing bracket belongs to the segment $[l_i, r_i]$.

Help Arthur restore his favorite correct bracket sequence!

Input

The first line contains integer n ($1 \le n \le 600$), the number of opening brackets in Arthur's favorite correct bracket sequence.

Next n lines contain numbers l_i and r_i ($1 \le l_i \le r_i < 2n$), representing the segment where lies the distance from the i-th opening bracket and the corresponding closing one.

The descriptions of the segments are given in the order in which the opening brackets occur in Arthur's favorite sequence if we list them from left to right.

Output

If it is possible to restore the correct bracket sequence by the given data, print any possible choice.

If Arthur got something wrong, and there are no sequences corresponding to the given information, print a single line "`IMPOSSIBLE`" (without the quotes).

Examples

input

```
4
1 1
1 1
1 1
1 1
```

output

```
()()()()
```

input

```
3
5 5
3 3
1 1
```

output

```
((()))
```

input

```
3
5 5
3 3
2 2
```

output

```
IMPOSSIBLE
```

input

```
3
2 3
1 4
1 4
```

output

```
(())()
```