



PROBLEM 9 - COLOURS

A manager for a toy company wants to reduce the cost of manufacturing their line of toys. Briefly, the toys are created by robots that operate on assembly tracks by adding and linking track components into modules and by merging existing modules into more complex modules. Components can be either *active* or inactive. At any moment there is exactly one *active component* per each *module* and *track*; and this is the only component that can be linked or merged on that track for that module.

The new budget for this company will only support components which consist of *three colours* and *adjacent components* must have *different colours*. Your job is to decide which of the current toys in the inventory can be produced with this colouring limitation.

The company uses the following BNF formalism to describe more precisely the blueprints of its toys:

1. $\langle \text{toy} \rangle ::= \langle \text{last-track} \rangle \langle \text{module} \rangle$

The current $\langle \text{toy} \rangle$ consists of a main $\langle \text{module} \rangle$. $\langle \text{last-track} \rangle$ gives the number of the last *track* used to build the current $\langle \text{toy} \rangle$; the *tracks* are numbered from 0 to $\langle \text{last-track} \rangle$.

2. $\langle \text{module} \rangle ::= (' \langle \text{operator-sequence} \rangle ')$

$\langle \text{module} \rangle$ represents a *simple module* that only contains *operators*.

The *operators* given by the $\langle \text{operator-sequence} \rangle$ are processed in a left-to-right order.

This $\langle \text{module} \rangle$ starts with empty tracks and then automatically adds one *active component* on each of the available *tracks*.

3. $\langle \text{merged-module} \rangle ::= (' \langle \text{module} \rangle_1 \langle \text{module} \rangle_2 ')$

This is a *merge* operation that builds a complex $\langle \text{merged-module} \rangle$ by merging *the active components* of $\langle \text{module} \rangle_1$ and $\langle \text{module} \rangle_2$, after both modules are completely built.

4. $\langle \text{module} \rangle ::= (' \langle \text{merged-module} \rangle \langle \text{operator-sequence} \rangle ')$

The *components* of the $\langle \text{merged-module} \rangle$ remain on the *tracks* and its *active components* are further worked upon by the *operators* given by $\langle \text{operator-sequence} \rangle$



5. $\langle \text{operator-sequence} \rangle ::= \lambda \mid \langle \text{operator} \rangle \langle \text{operator-sequence} \rangle$
6. $\langle \text{operator} \rangle ::= \langle \text{node-operator} \rangle \mid \langle \text{edge-operator} \rangle$
7. $\langle \text{node-operator} \rangle ::= \langle \text{track-number} \rangle$

A $\langle \text{node-operator} \rangle$ adds an *active component* on the specified $\langle \text{track-number} \rangle$ for the current module and the previously *active component* on that track becomes inactive.

8. $\langle \text{edge-operator} \rangle ::= \langle \text{track-number-pair} \rangle$

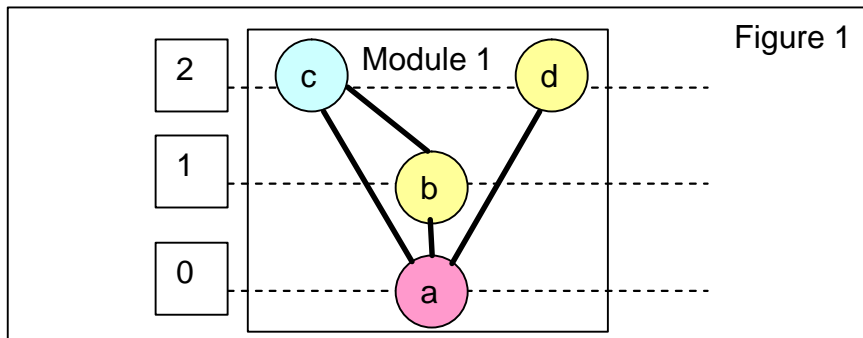
An $\langle \text{edge-operator} \rangle$ *links* the *active components* of the two *track-numbers*.

The following examples show several simple toy blueprints; in the figures tracks are represented as horizontal dotted lines, components as circles, links as full lines, and the time axis flows from left to right.

EXAMPLE 1

Figure 1 depicts a 3-colourable toy that can be built using the following blueprint:

2 (20 10 21 2 20)



There are 3 tracks numbered 0, 1, 2. The toy consists of a single module containing 4 components labelled a, b, c, d , linked by the lines $c-a, b-a, c-b, d-a$. To build this toy the robot will execute in order the following operations:

1. Add a on track 0, b on track 1, c on track 2. At this stage a, b, c are the active components.
2. Make the links $c-a, b-a, c-b$.
3. Add d on track 2, which makes d active and inactivates c .
4. Make the link $d-a$. At this stage a, b, d are the active components.

Note that the same toy can also be built using several other blueprints, such as the following two:

2 (10 20 21 2 20)

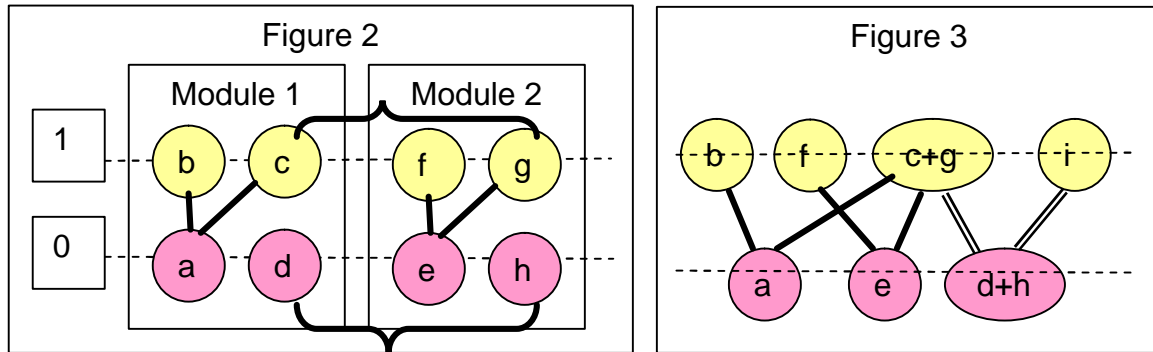
2 (((20 10) (21)) 2 20)



EXAMPLE 2

Figures 2 and 3 illustrate a sequence of operations involving a *merge*, for another 3-colourable (in fact even 2-colourable) toy that can be built as specified by the following blueprint:

1 (((1 0 1 1 0 0) (1 0 1 1 0 0)) 1 0 1 1 0)



1. First, module 1 is built:
 - a. Add a on track 0 and b on track 1.
 - b. Link $b-a$.
 - c. Add c on track 1.
 - d. Link $c-a$.
 - e. Add d on track 0. c, d are now the active components of module 1.
2. Secondly, module 2 is built using similar operations. g, h are now the active components of module 2.
3. Thirdly, modules 1 and 2 are *merged* together, which means that *active* components of each track are identified, i.e.,

$c=g, d=h$.

The snapshot of Figure 2 illustrates this moment, with braces showing component identification.
4. Fourthly, the just merged components are linked, i.e.,

$(c+g)-(d+h)$

$(c+g)-(d+h)$ are now the active components of the merged module 1+2.
5. Lastly, i is added to track 1 and linked with $(d+h)$.

The final result is depicted in Figure 3 and links made after the merging are depicted with double lines. At the end, i and $(d+h)$ are the active components of the main module.



INPUT FORMAT

The input will consist of a sequence of toy blueprints, one per line of at most 250 characters. Each toy blueprint contains a sequence of tokens separated by single spaces, and conforming to the BNF rules stated earlier.

The first token is a positive integer t , $0 \leq t \leq 6$, denoting the maximum track number for the robot's arms to grab (i.e., there are $t+1$ current components for the robot).

The interpretations for the remaining tokens are given in the next table.

Token	Meaning of the Robot's Instruction
i	Add a new component on track i , where i is a decimal digit, $0 \leq i \leq t$. Note that this component becomes the <i>active</i> component on this track
ij	Link the <i>active</i> components on tracks i and j , where i and j are decimal digits, $t \geq i > j \geq 0$. Note this token consists of two track numbers, with no intervening space.
(Begin marker for a new module. Note that, according to the BNF description two tokens ')' and '(' adjacent in sequence denote a <i>merge</i> operation.
)	End marker for a new module.

The input will be terminated by a toy description with $t=0$, which is not processed.

SAMPLE INPUT:

```
2 ( 20 10 21 2 20 )
1 ( ( ( 10 1 10 0 ) ( 10 1 10 0 ) ) 10 1 10 )
3 ( 32 31 20 21 10 0 10 30 20 )
2 ( ( ( 10 1 10 21 1 21 10 ) ( 21 ) ) 0 10 20 )
0 ( )
```

OUTPUT FORMAT

The required output is a line of the form "Toy #: ?", where # denotes the toy sequence number starting at 1 and ? is either Yes or No depending whether the toy can be properly built using at most 3 colours.

SAMPLE OUTPUT:

```
Toy 1: Yes
Toy 2: Yes
Toy 3: No
Toy 4: Yes
```