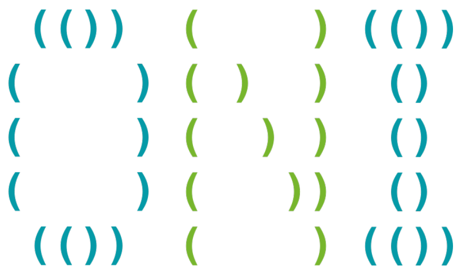
Problem B – Remoção de Parênteses

João is developing a new text processor that uses pairs of parentheses '`(`' and '`)`' to mark italic and bold regions in the text. To stand out from other products on the market, João is trying to add new deletion operations to help users erase text; these operations are specified in the following parts.



You will be given $\boldsymbol{T}$ strings; the i-th string has length $\boldsymbol{N_i}$ and contains only the characters '`(`' and '`)`'. In each part below you must indicate how João can apply his removal operation to delete each string completely.

Part I

In this part you may perform the following operation on a string any number of times:

- Choose a position *between* two consecutive characters of the string.
- Select the first '`(`' to the left of that position.
- Select the first '`)`' to the right of that position.
- Delete both parentheses and every character between them.

The operation can only be applied if there is a '`(`' to the left and a '`)`' to the right of that position.

For each string, decide whether it can be deleted completely using only this operation.

Example

Suppose $\boldsymbol{N} = 4$ and the string is `(())`. Choose the position between the 2nd and 3rd parentheses; that removes the 2nd and 3rd parentheses and leaves `()`. Then choose the position between the remaining parentheses, removing them and leaving the empty string. Thus the answer is SIM.

Constraints

The following limits hold for every test case of this part:

$1 \leq \boldsymbol{T} \leq 20$	Number of strings
$1 \leq \boldsymbol{N_i} \leq 1\,000$	Length of each string

All test cases for this part belong to the single group:

Subtask	Points	Additional Constraints
1	20	No additional constraints

Part II

For this part you must *output* a sequence of positions—if they exist—which when chosen in that order delete the entire string using the operation described in Part I.

Positions are indexed between characters, as in the following example for `()()`:

```
indices between characters
 1    2    3    4    5
    (    )    (    )
    1    2    3    4
   character indices
```

After each removal, the resulting (shorter) string is re-indexed for future operations.

Example

Let $n = 6$ and the string be `(())()`. A valid sequence of positions is 3, 4, 2, because (a vertical bar `|` marks the chosen position):

- position 3: `((|))()` $\rightarrow$ delete interior $\rightarrow$ `()()`;
- position 4: `()(|)` $\rightarrow$ delete right pair $\rightarrow$ `()`;
- position 2: `(|)` $\rightarrow$ delete both $\rightarrow$ empty string.

Constraints

The following limits hold for every test case of this part:

$1 \leq \boldsymbol{T} \leq 20$	Number of strings
$1 \leq \boldsymbol{N_i} \leq 1\,000$	Length of each string

The test cases for this part are organised into two groups:

Subtask	Points	Additional Constraints
2	15	$\boldsymbol{N_i} \leq 100$
3	15	No additional constraints

Part III

For this part you now have a *cursor* that:

- Starts at position 1 (to the left of the entire string).
- Can move one position at a time to the right (positions range from 1 to $\boldsymbol{N}+1$).
- Can delete the character immediately to the left of the cursor.
- Can perform the Part I removal operation at the cursor's current position.

It is always possible to delete the whole string, but João wants to save time. Determine the minimum number of operations (cursor moves and deletions) required.

Example

Let $\boldsymbol{N} = 5$ and the string be `)()()`. The cursor starts at position 1, i.e. `|)()()`

1. Move right (cursor at 2): `)|()()`
2. Delete the character to the left (`)`): `|()()`
3. Move right (cursor at 2): `(|)()`
4. Move right (cursor at 3): `()|()`
5. Perform the Part I operation, deleting the remaining string.

A total of **5** operations.

Constraints

The following limits hold for every test case of this part:

$1 \le \boldsymbol{T} \le 20$	Number of strings
$1 \le \boldsymbol{N_i} \le 1\,000$	Length of each string

The test cases for this part are organised into three groups:

Subtask	Points	Additional Constraints
4	10	$\boldsymbol{N_i} \le 10$
5	20	$\boldsymbol{N_i} \le 100$
6	20	No additional constraints

Summary of Subtasks

All test cases are organised into six groups:

Subtask	Points	Part	Additional Constraints
1	20	Part I	No additional constraints
2	15	Part II	$N_i \leq 100$
3	15	Part II	No additional constraints
4	10	Part III	$N_i \leq 10$
5	20	Part III	$N_i \leq 100$
6	20	Part III	No additional constraints

Input Format

The first line contains an integer P, indicating which part of the problem the test belongs to (1, 2, or 3).

The second line contains an integer T, the number of strings to process.

Then follow T pairs of lines; for the i-th string:

- A line with an integer N_i — the length of the string.
- A line with a string of length N_i containing only '`(`' and '`)`'.

Output Format

The output must contain T lines:

Part I

Print `SIM` (yes in Portuguese) if the string can be deleted completely, or `NAO` (no in Portuguese) otherwise.

Part II

Print one line with an integer K (the number of operations) followed by a second line with K integers — the positions chosen, in the order they are applied.

If the string cannot be deleted completely, print only the single integer -1.

Part III

Print a single integer: the minimum number of operations (cursor moves and deletions) needed to delete the string.

Example 1 Input

```
1
2
4
(())
4
))()
```

Example 1 Output

```
SIM
NAO
```

Example 1 Description

This example corresponds to the example from Part I in the statement.

Example 2 Input

```
2
2
6
(())()
4
((((
```

Example 2 Output

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

Example 2 Description

This example corresponds to the example from Part II in the statement.

Example 3 Input

```
3
1
5
)()()
```

Example 3 Output

```
5
```

Example 3 Description

This example corresponds to the example from Part III in the statement.