

Swapping Brackets (bracketswap)


Do you remember Baby Bob? His older brother, Balanced Benjamin would like to give a gift to his brother. He has a string S of length N , where N is even. The string consists of $\frac{N}{2}$ opening brackets (“(”) and $\frac{N}{2}$ closing brackets (“)”). As Bob likes *valid bracket sequences*¹, Benjamin would like to turn S into a valid bracket sequence by swapping (not necessarily adjacent) characters in S .



Figure 1: Benjamin constructing the gift.

Since he’s less mathematically inclined than his younger brother, he asks for *your* help.

Write a program that turns S into a balanced bracket sequence by swapping characters in S using the minimum number of swaps. If there are more than one way to do this, you can output any of them.

 Among the attachments of this task you may find a template file `bracketswap.*` with a sample incomplete implementation.

Input

The input file consists of:

- a line containing integer N , the length of S .
- a line containing string S , a bracket sequence of length N .

Output

The output file consists of:

- a line containing integer R , the minimum number of swaps necessary.
- R lines, each line contains the indices (0-based) of the two characters to be swapped.





¹A bracket sequence is valid if “1” and “+” characters can be inserted into it so that it becomes a valid mathematical expression. For example, “()()()())” is a valid bracket sequence.

Constraints

- $1 \leq N \leq 1\,000\,000$.
- N is even.
- S has length N and only contains characters “(” and “)”.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (33 points) $N \leq 16$.

- **Subtask 3** (44 points) $N \leq 5678$.

- **Subtask 4** (23 points) No additional limitations.


Examples

input	output
4) () (1 0 3
8))) ((((2 0 5 7 1
10 () (() (())	0

Explanation

In the **first sample case** the string after the swap looks like this: “(())”.

In the **second sample case** the swaps change string S the following way:

$$)))((((\Rightarrow ())(())(\Rightarrow (((((((((($$

In the **third sample case** the bracket sequence is already valid.