

Problem 1. Balanced Parentheses

When programming, we use parentheses `()`, brackets `[]`, and curly braces `{}` a lot. Sometimes, we forget a parenthesis or add an extra one by mistake. Many modern coding environments will automatically check to see if a set of these characters are “balanced” to save time in debugging. A set of parentheses, brackets, and braces is called “balanced” if for every open parenthesis `(`, there is a matching closing parenthesis `)` paired with it and likewise for brackets and braces. Everything between the paired characters must also recursively be a balanced string. For example, these sets of parentheses, brackets, and braces are balanced:

`([{ }])` `(){}[()]` `((()()())())`

While these are not:

`((()()` `[()]` `{ }() [()]}`

Your task is to write a program that will check whether or not a given set of parentheses is balanced.

Input

The input consists of a number n , with $1 \leq n \leq 1000$, followed by n lines. Each line will contain a string containing only the characters `()[]{}.` The length of each string will not exceed 1000 characters.

Output

Write n lines. On each line, write “Balanced” if the corresponding set of parentheses, brackets, and braces is balanced and “Not Balanced” if it is not.

Example

standard input	standard output
6	Balanced
([{ }])	Balanced
(){}[()]	Balanced
((()()())())	Not Balanced
((()()	Not Balanced
[()]	Not Balanced
{ }() [()]}	