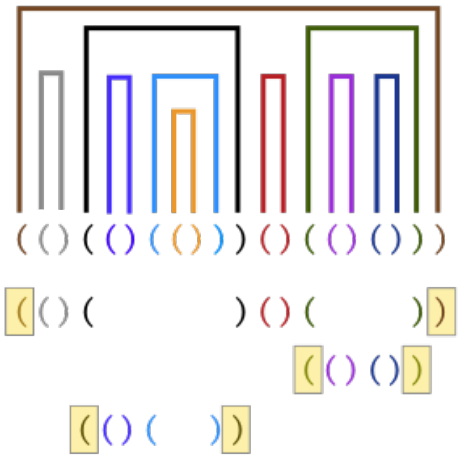
# Balanced parentheses

Python requires all parentheses to be balanced. This means that every parenthesis that is opened in the source code must be closed further downstream, and that every closed parenthesis must have been opened upstream. Unbalanced parentheses result in a compiler error. The figure below shows the corresponding parentheses in the balanced code snippet (()(() (()))()(()())).



Example indicating the matching pairs of parenthesis in the balanced code snippet (()(()(()))()(()())).

Write a program that can decide whether or not the parentheses in a given text fragment are balanced.

## Input

The first line of the input contains a number $t \in \mathbb{N}$ that indicates the number of text fragments that will follow. This is followed by another $t$ lines, each containing a text fragment. You may assume that each text fragment only contains a single type of parentheses. The types of parentheses that should be taken into account are ( and ), { and }, < and > and [ and ].

## Output

Determine for each given text fragment whether or not the parenthesis used are balanced. Use the terms balanced and unbalanced to write out the evaluation.

## Algorithm

If a given text fragment only contains a single type of parentheses, the following simple algorithm can be used to determine whether or not these parentheses are balanced:

1. initialize a counter to the value 0
2. traverse the characters of the text fragment left to right, and
    a. increment the counter by one if the character is an opened parenthesis
    b. decrement the counter by one if the character is a closed parenthesis; if this gives a negative counter, it means that there are more closed than opened parentheses at this point in the text fragment
3. if the counter is not equal to zero after the traversal, it means that there is an unequal

number of opened and closed parentheses

## Example

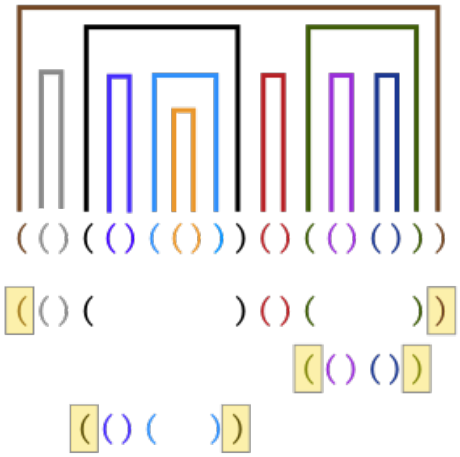
### Input:

```
4
(()()(()))
[[[[[ ]]]]
one (two (three) four) five
)(
```

### Output:

```
balanced
unbalanced
balanced
unbalanced
```

In Python moeten alle haakjes gebalanceerd zijn. Dat betekent dat elk geopend haakje verderop in de broncode ook moet afgesloten worden en dat elk haakje dat gesloten wordt, eerder in de broncode reeds moet geopend zijn. Als de haakjes niet gebalanceerd zijn, dan resulteert dit in een compileerfout. Onderstaande illustratie geeft bijvoorbeeld aan welk haakjes met elkaar corresponderen in het gebalanceerde tekstfragment (()(()(()))()(()())).



Voorbeeld dat aangeeft welke haakjes met elkaar corresponderen in het gebalanceerde tekstfragment (()(()(()))()(()())).

Schrijf een programma dat voor een gegeven tekstfragment kan beslissen of de gebruikte haakjes al dan niet gebalanceerd zijn.

## Invoer

De eerste regel van de invoer bevat een getal $t \in \mathbb{N}$ dat aangeeft hoeveel tekstfragmenten er volgen. Daarna volgen $t$ regels die elk een een tekstfragment bevatten. Je mag ervan uitgaan dat in elk tekstfragment slechts één soort haakjes gebruikt wordt. De mogelijke haakjes waarmee je rekening moet houden zijn ( en ), { en }, < en > en [ en ].

## Uitvoer

Bepaal voor elk gegeven tekstfragment of de gebruikte haakjes al dan niet gebalanceerd zijn. Gebruik hiervoor de termen gebalanceerd and ongebalanceerd.

## Algoritme

Als een gegeven tekstfragment maar één soort haakjes gebruikt, dan kan je volgend eenvoudig algoritme gebruiken om na te gaan of die haakjes gebalanceerd zijn:

1. initialiseer een teller met waarde 0
2. doorloop de karakters van het tekstfragment van links naar rechts, en
    a. verhoog de teller met één indien het karakter een geopend haakje is
    b. verlaag de teller met één indien het karakter een gesloten haakje is; als de teller hierna negatief geworden is, dan betekent het dat er vóór dit punt in het tekstfragment meer gesloten dan geopende haakjes zijn
3. als de teller na het doorlopen niet gelijk is aan nul, dan betekent dit dat het aantal geopende haakjes niet gelijk is aan het aantal gesloten haakjes

## Voorbeeld

### Invoer:

```
4
(()()(()))
[[[[[ ]]]]
een (twee (drie) vier) vijf
)(
```

### Uitvoer:

```
gebalanceerd
ongebalanceerd
gebalanceerd
ongebalanceerd
```