Indexar seqüències ben parentitzades X80203_ca

Preliminars:
En aquests preliminars expliquem qué és una seqüència ben-parentitzada sobre (,), i quin és el corresponent parèntesis de tancar per a cada parèntesis d'obrir. Si ja teniu clars aquests conceptes, podeu deixar de llegir els preliminars i anar directament a l'exercici en sí.
Una seqüència ben parentitzada és un string s format amb els caràcters d'obrir i tancar parèntesis, és a dir `(` i `)`, que cumpleix les següents condicions:

- Tot prefix de s té més o igual parèntesis d'obrir que de tancar.
- s té en total la mateixa quantitat de parèntesis d'obrir que de tancar.

Sigui s una seqüència ben parentitzada i sigui i una posició de s a on hi trobem un parèntesi d'obrir (és a dir $s[i] == '('$). Sigui j la posició més petita d'entre les que cumpleixen $i < j$ i tals que el substring $s[i..j]$ té tants parentesis d'obrir com de tancar. Resulta que a posició j hi ha d'haver un parèntesis de tancar, i diem que aquest és el corresponent parèntesis de tancar al parèntesis d'obrir que es troba a posició i.
Exercici:
Escriviu un programa que rep seqüències ben-parentitzades d'entrada i les torna a escriure per la sortida, però insertant un número darrera de cada parèntesi de manera que:

- El primer parèntesi d'obrir està seguit d'un 1, el segon parèntesi d'obrir d'un 2, el tercer parèntesi d'obrir d'un 3, i així successivament.
- Per a cada parèntesis d'obrir, el seu corresponent parèntesis de tancar està seguit del mateix número.

Observació: Convé que utilitzeu la classe `stack` per a resoldre aquest exercici de manera eficient.

Entrada

L'entrada conté un nombre arbitrari de casos, un per línia. Cada cas consisteix en un string ben parentitzat.

Sortida

Per a cada cas, escriviu en una línia el mateix string, però afegint darrera de cada parèntesis un número, de manera que els parèntesis d'obrir estan identificats començant des de 1 i creixentment de un en un, i els seus corresponents parèntesis de tancar estan identificats pels mateixos números.

Exemple d'entrada 1

```
()
() ()
() () ()
(())
((()))
```

```
(() ()) (() ())
() (()) (() ()) ((() ()) (() ()))
(() ((())) (() ()) ((() ()) (() ())))
```

Exemple de sortida 1

```
(1)1
(1)1(2)2
(1)1(2)2(3)3
```

Exemple d'entrada 2

```
()()
(()(())())
(())()
()()(())(())()
(((())))(())
(((()))((())))
()()
()(())
()
(()())()
((()))()
(()())
(())()()
(())(())
()
((()))()
((((()))))
(()())()(())
()()
()(()(()))
()
()
()
((((()))()())
()((()))()
((())(((()))))
((()(())
((()))()()
(())
((((()))))()
()(())()
()(())
((()))((()))
((()))
()
(()((()))())()
()
(())
((()))
(()())
```

```
(1(2)2)1
(1(2(3)3)2)1
(1(2)2(3)3)1(4(5)5(6)6)4
(1)1(2(3)3)2(4(5)5(6)6)4(7(8(9)9(10)10)8(11(12)12(13)13
(1(2)2(3(4)4)3(5(6)6(7)7)5(8(9(10)10(11)11)9(12(13)13(1
```

Exemple de sortida 2

```
(1)1(2)2
(1(2)2(3(4)4(5)5)3(6)6)1
(1(2)2)1(3)3
(1)1(2)2(3(4)4)3(5(6)6)5(7)7
(1(2(3)3(4)4)2)1(5(6)6)5
(1(2(3)3(4)4)2(5(6)6)5)1
(1)1(2)2
(1)1(2(3)3)2
(1)1
(1(2)2(3)3)1(4)4
(1(2(3)3)2)1(4)4
(1(2)2(3)3)1
(1(2)2)1(3)3(4)4
(1(2)2)1(3(4)4)3
(1)1
(1(2(3)3)2)1(4)4
(1(2(3(4(5)5)4)3)2)1
(1(2)2(3)3)1(4)4(5(6)6)5
(1)1(2)2
(1)1(2(3)3(4(5)5)4(6)6)2
(1)1
(1)1
(1)1
(1(2(3(4)4)3)2(5)5(6)6)1
(1)1(2(3(4)4)3)2(5)5
(1(2)2)1(3(4(5(6)6)5(7)7)4)3
(1(2)2(3(4)4)3)1
(1(2)2)1(3)3(4)4
(1(2)2)1
(1(2(3(4(5)5)4)3)2)1(6)6
(1)1(2(3)3)2(4)4
(1)1(2(3)3)2
(1(2(3)3)2)1(4(5(6)6)5)4
(1(2(3)3)2)1
(1)1
(1(2)2(3(4)4)3(5)5)1(6)6
(1)1
(1(2)2)1
(1(2(3)3)2)1
(1(2)2(3)3)1
```

Observació

Informació del problema

Autoria: PRO2

Generació: 2026-01-25T16:33:25.128Z