

Flamaster

Jasio uwielbia porządek i bardzo irytują go nieposprzątane nawiasy. Ostatnio zobaczył napis „ $()\{([\])$ ”, który go zdenerwował, gdyż nie jest to poprawne nawiasowanie. Szczęśliwie okazało się, że Jasio miał w kieszeni flamaster i mógł dopisać na początku napisu „(”, zaś na końcu „)” i już mógł spać spokojnie, gdyż ciąg „ $()\{([\])\}$ ” jest poprawnym nawiasowaniem.

Jasio postanowił, że misją jego życia będzie naprawianie ciągów nawiasów poprzez dopisywanie nawiasów na początku i na końcu zobaczonego ciągu, tak by uzyskany ciąg nawiasów był poprawny. Jako że jego flamaster ma mało atramentu, Jasio chciałby dopisać możliwie mało nawiasów, aby osiągnąć swój cel. Pomóż mu! Napisz program, który: wczyta ciąg nawiasów, obliczy dla niego najkrótszy poprawiony ciąg nawiasów i wypisze go lub stwierdzi, że takiego ciągu nie ma.

Poprawne nawiasowania są zdefiniowane rekurencyjnie w następujący sposób:

- ciąg pusty jest poprawnym nawiasowaniem;
- jeśli S i T to poprawne nawiasowania, to ich połączenie ST też jest poprawnym nawiasowaniem;
- jeśli S to poprawne nawiasowanie, to poprawnymi nawiasowaniami są też (S) , $[S]$ oraz $\{S\}$.

Wejście

W pierwszym wierszu wejścia znajduje się liczba t odpowiadająca liczbie przypadków testowych. Każdemu przypadkowi testowemu odpowiada dokładnie jeden wiersz zawierający niepusty ciąg składający się ze znaków $(,), [,], \{, \}$ o długości nieprzekraczającej 1000000; jest to ciąg nawiasów, który ma naprawić Jasio.

Wyjście

Wyjście powinno zawierać t wierszy, po jednym dla każdego przypadku testowego, przy czym, oczywiście, kolejność odpowiedzi musi odpowiadać kolejności przypadków testowych wejścia. Dla każdego z przypadków wiersz wyjścia powinien zawierać poprawiony ciąg nawiasów z wejścia. Jeśli istnieje wiele poprawionych nawiasowań o najmniejszej możliwej długości, należy wypisać którekolwiek z nich.

Jeśli naprawienie wejściowego ciągu nawiasów jest niemożliwe, należy wypisać słowo NIE.

Przykład

Wejście	Wyjście
2) { ([([] }	() { ([]) NIE