You are given a string s that consists of English letters, punctuation marks, whitespace characters and brackets. It is guaranteed that the brackets in s form a [regular bracket sequence](keyword://regular-bracket-sequence).

Your task is to reverse the strings in each pair of matching parenthesis, starting from the innermost one.

**Example**

For string "s = a(bc)de" the output should be  
reverseParentheses(s) = "acbde".

**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] string s**

A string consisting of English letters, punctuation marks, whitespace characters and brackets. It is guaranteed that parenthesis form a *regular bracket sequence*.

*Constraints:*  
5 ≤ x.length ≤ 55.

* **[output] string**

<https://codefights.com/challenge/Wg7u4KrHsYbZqvDar>

--------------SOLUCION MAS COMPACTA--------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static int[] arrayNiveles2(string s)

{

int[] niveles = new int[s.Length];

int nivel = 0;

for (int i = 0; i < s.Length; i++)

{

if (s[i] == '(')

{

nivel++;

niveles[i] = nivel;

}

else if (s[i] == ')')

{

niveles[i] = nivel;

nivel--;

}

}

return niveles;

}

static string reverseParentheses(string s)

{

char[] c = s.ToCharArray();

int[] niveles = arrayNiveles2(new string(c));

int i;

int max = niveles.Max();

while (max >= 1)

{

i = Array.IndexOf(niveles, max);

while (i < s.Length)

{

int inicio = i;

while (i < s.Length)

{

if (niveles[i] == max && s[i] == '(')

{

inicio = i;

break;

}

i++;

}

int fin = i + 1;

while (i < s.Length)

{

if (niveles[i] == max && s[i] == ')')

{

fin = i;

break;

}

i++;

}

if (i < niveles.Length && i >= 0)

{ Array.Reverse(c, inicio, fin - inicio + 1); }

i++;

}

max--;

}

return String.Join("", new string(c).Split('(', ')'));

}

static void Main(string[] args)

{

//string s = "co(de(fight)s)";

//Console.WriteLine(reverseParentheses(s));

//s = "Code(Cha(lle)nge)";

//Console.WriteLine(reverseParentheses(s));

//s = "a(bc)de";

//Console.WriteLine(reverseParentheses(s));

//s = "abcde";

//Console.WriteLine(reverseParentheses(s));

string s = "co(de(fight)s)";

Console.WriteLine(reverseParentheses(s));

//string s = "(())";

//int[] niv1 = arrayNiveles1(s);

//foreach (int elem in niv1)

//{

// Console.Write(elem + " ");

//}

//Console.WriteLine();

//int[] niv2 = arrayNiveles2(s);

//foreach (int elem in niv2)

//{

// Console.Write(elem + " ");

//}

Console.ReadLine();

}

}

}

--------------SOLUCION PRESENTADA----------------------

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

//static void revertir(int[] arr)

//{

// for (int i = 0; i < arr.Length / 2; i++)

// {

// int temp = arr[i];

// arr[i] = arr[arr.Length - i - 1];

// arr[arr.Length - i - 1] = temp;

// }

//}

static int[] arrayNiveles(string expresion)

{

int maxNivel = 0;

int[] niveles = new int[expresion.Length];

Stack<char> pila = new Stack<char>();

bool balanceado = true;

for (int i = 0; i < expresion.Length && balanceado; i++)

{

char actual = expresion[i];

if (actual == '(')

{

//nivel++;

pila.Push(actual);

maxNivel = Math.Max(pila.Count, maxNivel);

niveles[i] = pila.Count;

}

else if (actual == ')')

{

niveles[i] = pila.Count;

balanceado = (pila.Count > 0) && pila.Pop() == '(';

}

}

return niveles;

}

static void revertir(char[] arr, int inicio, int fin)

{

for (int i = inicio; i < Math.Ceiling( (double)(inicio + fin)/2) ; i++)

{

char temp = arr[i];

arr[i] = arr[fin - (i-inicio) ];

arr[fin - (i-inicio) ] = temp;

}

}

static string reverseParentheses(string s)

{

char[] c = s.ToCharArray();

int[] niveles = arrayNiveles(new string(c));

int i;

for ( i = 0; i < niveles.Length; i++)

{

Console.Write(niveles[i] + "");

}

Console.WriteLine();

int max = niveles.Max();

int indexOfMax = Array.IndexOf(niveles, max);

i = indexOfMax + 1;

while (i < niveles.Length)

{

if (niveles[i] == max && s[i] == ')')

{

if (i < niveles.Length && i >= 0)

{

revertir(c, indexOfMax , i );

}

break;

}

i++;

}

//Console.WriteLine();

while (max >= 1)

{

while (i < s.Length)

{

int inicio = i;

while (i < s.Length)

{

if (niveles[i] == max && s[i] == '(')

{

inicio = i;

break;

}

i++;

}

int fin = i + 1;

while (i < s.Length)

{

if (niveles[i] == max && s[i] == ')')

{

fin = i;

break;

}

i++;

}

if (i < niveles.Length && i >= 0)

{

revertir(c, inicio , fin );

}

i++;

}

i = 0;

max--;

// i = Array.IndexOf(niveles, max);

}

string[] res = new string(c).Split('(',')');

return String.Join("", res);

}

static void Main(string[] args)

{

//string s = "a(bcdefghijkl(mno)p)q";

//string s = "(()())";

// string s = "(aa(aaa)aaaa(aaaa)aaa(a)a(a)aa)aaa";

//string s = "((()())(()))";

// Console.WriteLine(arrayNiveles(s));

// char[] s = "(12(3)456(78)9)".ToCharArray();

//string s = "(1(234)56(78)9(ab)(cd))0";

string s = "co(de(fight)s)";

Console.WriteLine(reverseParentheses(s));

s = "Code(Cha(lle)nge)";

Console.WriteLine(reverseParentheses(s));

s = "a(bc)de";

Console.WriteLine(reverseParentheses(s));

s = "abcde";

Console.WriteLine(reverseParentheses(s));

Console.ReadLine();

}

}

}

-----------------OTRAS SOLUCIONES-----------------------------

string reverseParentheses(string s)

{

while (s.IndexOf('(') >= 0)

{

int r = s.IndexOf(')');

int l = s.Substring(0, r).LastIndexOf('(');

string t = s.Substring(0, l);

for (int i = r - 1; i > l; i--)

{

t += s[i];

}

t += s.Substring(r + 1);

s = t;

}

return s;

}

string reverseParentheses(string s)

{

var c = s.ToArray();

var t = new Stack<int>();

for (var i = 0; i < c.Length; i++)

{

if (c[i] == '(') t.Push(i);

if (c[i] == ')')

{

var l = t.Pop();

Array.Reverse(c, l, i - l);

}

}

return String.Join("", new string(c).Split(new[] { '(', ')' }));

}

string reverseParentheses(string s)

{

while (s.Contains("("))

{

int e = s.IndexOf(')');

int b = s.Substring(0, e).LastIndexOf('(') + 1;

var c = s.Substring(b, e - b).ToCharArray();

Array.Reverse(c);

s = s.Substring(0, b - 1) + new String(c) + s.Substring(e + 1);

}

return s;

}