I define a shortened string as a string after being shortened from a full-string (a full-string is a string just contains uppercase English letters) and it follows the rules:

* Xm - The character X rewrites mtimes;
* (S)m - The string S rewrites mtimes.

For example, "H3A" is a shortened string of "HHHA".  
(HDL)5 is a shortened string of "HDLHDLHDLHDLHDL".  
(AH0)2AD is a shortened string of "AAAD".

Given a shortened string, return its full-string.

**Example**  
For ss = "KB2 (Y2F)2 B5A". the output should be  
shortenedString(ss) = "KBBYYFYYFBBBBBA".

**Input/Output**

* **[execution time limit] 3 seconds (cs)**
* **[input] string ss**

A string consisting of uppercase English letters (A, B, C,...,Z), numbers, whitespace characters and brackets. It is guaranteed that parentheses form a regular bracket sequence.

*Guaranteed constraints:*  
1 ≤ ss.length ≤ 1000.

* **[output] string**

A string just contains uppercase English letters. The whitespace characters is unnecessary in a full-string.

**[C#] Syntax Tips**

// Prints help message to the console

// Returns a string

string helloWorld(string name) {

Console.Write("This prints to the console when you Run Tests");

return "Hello, " + name;

}

<https://codefights.com/challenge/uTaa6ELcB5u6cwifr/solutions>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp10

{

class Program

{

static string shortenedString(string ss)

{

ss = ss.Replace(" ", "");

for (int i = 1; i < ss.Length; i++)

{

if (char.IsNumber(ss[i]))

{

int t = i + 1;

while (t < ss.Length && char.IsNumber(ss[t]))

{

t++;

}

int num = int.Parse(ss.Substring(i, t - i));

int lennum = ss.Substring(i, t - i).Length;

//Console.WriteLine(num);

if (i - 1 >= 0 && ss[i - 1] == ')')

{

int j = i - 1;

while (j >= 0 && ss[j] != '(')

{

j--;

}

string subs = ss.Substring(j + 1, i - j - 2);

string rep = string.Concat(Enumerable.Repeat(subs, num));

//Console.WriteLine(rep);

ss = ss.Remove(j, i + lennum - j);

//Console.WriteLine(ss);

ss = ss.Insert(j, rep);

i = rep.Length;

}

else

{

string rep = string.Concat(Enumerable.Repeat(ss[i - 1], num));

if (num > 0)

{

ss = ss.Remove(i - 1, 1 + lennum);

ss = ss.Insert(i - 1, rep);

i = rep.Length;

}

else if (num == 0)

{

ss = ss.Remove(i - 1, 1 + lennum);

i -= lennum;

}

}

}

}

return ss;

}

static void Main(string[] args)

{

//string s = string.Concat(Enumerable.Repeat("ab", 4));

//Console.WriteLine(s);

//string ss = "(HA)3";

//string ss = "(A(HI3)2)1";//"AHIIIHIII"

//string ss = "(X3Y2(Z1)3 M4B0)2H2O"; //"XXXYYZZZMMMMXXXYYZZZMMMMHHO"

// string ss = "(AH0)2AD"; //"AAAD"

//string ss = "(A)10B"; //"AAAAAAAAAAB"

string ss = "(AB)0000Z";

Console.WriteLine(shortenedString(ss));

Console.ReadLine();

}

}

}