<https://www.geeksforgeeks.org/print-all-combinations-of-balanced-parentheses/>

Print all combinations of balanced parentheses

Last Updated: 15-06-2020

Write a function to generate all possible n pairs of balanced parentheses.

**Examples:**

**Input:** n=1

**Output:** {}

**Explantaion:** This the only sequence of balanced

parenthesis formed using 1 pair of balanced parenthesis.

**Input :** n=2

**Output:**

{}{}

{{}}

**Explantaion:** This the only two sequences of balanced

parenthesis formed using 2 pair of balanced parenthesis.

[**Recommended: Please solve it on “*PRACTICE*” first, before moving on to the solution.**](https://practice.geeksforgeeks.org/problems/generate-all-possible-parentheses/1)

**Approach:**To form all the sequences of balanced bracket subsequences with n pairs. So there are n opening brackets and n closing brackets.  
So the subsequence will be of length 2\*n. There is a simple idea, the i’th character can be ‘{‘ if and only if the count of ‘{‘ till i’th is less than n and i’th character can be ‘}’ if and only if the count of ‘{‘ is greater than the count of ‘}’ till index i. If these two cases are followed then the resulting subsequence will always be balanced.  
So form the recursive function using the above two cases.

**Algorithm:**

1. Create a recursive function that accepts a string (s), count of opening brackets (o) and count of closing brackets (c) and the value of n.
2. if the value of opening bracket and closing bracket is equal to n then print the string and return.
3. If the count of opening bracket is greater than count of closing bracket then call the function recursively with the following parameters String *s + “}”*, count of opening bracket *o*, count of closing bracket *c + 1*, and n.
4. If the count of opening bracket is less than n then call the function recursively with the following parameters String *s + “{“*, count of opening bracket *o + 1*, count of closing bracket *c*, and n.

/ C# program to print all

// combinations of balanced parentheses

using System;

class GFG

{

    // Function that print all combinations of

    // balanced parentheses

    // open store the count of opening parenthesis

    // close store the count of closing parenthesis

    static void \_printParenthesis(char []str,

            int pos, int n, int open, int close)

    {

        if (close == n)

        {

            // print the possible combinations

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

                Console.Write(str[i]);

            Console.WriteLine();

            return;

        }

        else

        {

            if (open > close) {

                str[pos] = '}';

                \_printParenthesis(str, pos + 1,

                                n, open, close + 1);

            }

            if (open < n) {

                str[pos] = '{';

                \_printParenthesis(str, pos + 1,

                                n, open + 1, close);

            }

        }

    }

    // Wrapper over \_printParenthesis()

    static void printParenthesis(char []str, int n)

    {

        if(n > 0)

        \_printParenthesis(str, 0, n, 0, 0);

        return;

    }

    // driver program

    public static void Main()

    {

        int n = 3;

        char[] str = new char[2 \* n];

        printParenthesis(str, n);

    }

}