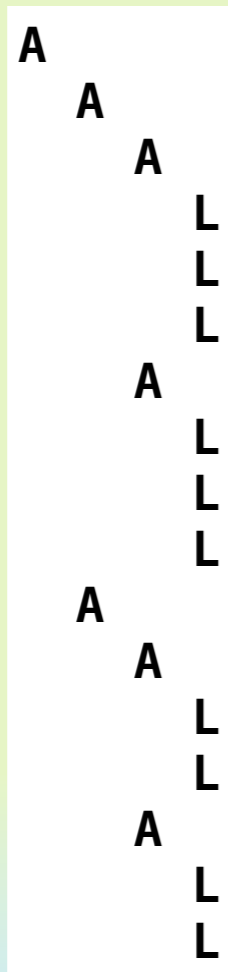


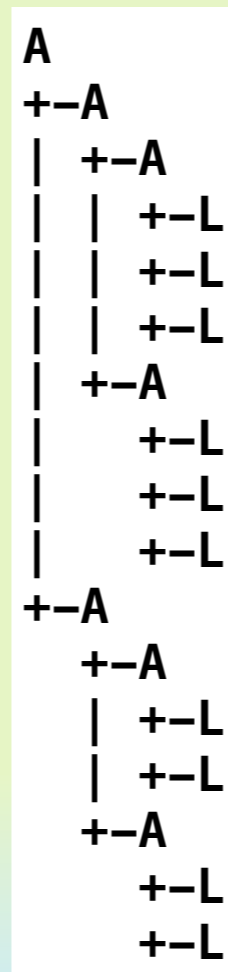
# 第7回課題

本日解説したスライドの5枚目「木構造の表現方法(2)」において下図(1)のように木を表現したこれを下図(2)のように出力してより見やすくなるようにプログラムを修正せよ。ただし、プログラムは関数kadai(tree)を提出すること。kadai(tree)はputsで出力するのではなく関数の値として一括して木のイメージを文字列として(3)のようなデータをreturnするようにせよ (\nは改行文字)。

(1)

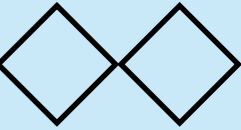


(2)



(3)

"A\n+-A\n| +-A\n| |  
+-L\n..."



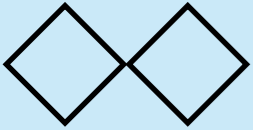
# 解答例 (1)

右のような図を作るためには子供のノードについては、"+-"に続けてノードの記号を出力する。また、それぞれの子供のノードについてそれが兄弟の中で最後のノードであれば、"|"をつけない。

まず、縦棒"|"のパターンpatが指定されたとき、|を含む文字列を出力する関数make\_patを定義する。縦棒がないところは空白2文字となる。

```
function kadai(tree){
    function make_pat(pat, level){
        var s = ""
        for(var ll = 0; ll < level; ll++){
            var ss = "  "
            for (var i = 0; i < pat.length; i++){
                if (pat[i] === ll) ss= "| "
            }
            s += ss
        }
        return s
    }
}
```

```
A
+-A
| +-A
| | +-L
| | +-L
| | +-L
+-A
  +-L
  +-L
  +-L
+-A
  +-A
  | +-L
  | +-L
+-A
  +-L
  +-L
```



## 解答例 (2)

make\_patを用いて、最後の子供以外はlevelをppatに付け加えて再帰的に呼び出す。文字列sに文字列を蓄えて最後に出力する。

```
var s = ""
function print_tree3(tree, level, pat){
  var len = tree.length
  var ss = ""
  if (level > 0) ss = make_pat(pat, level - 1) + "+-"
  s += ss + tree[0] + "\n"
  for (var i = 1; i < len; i++){
    var ppat = pat.slice(0)
    ppat.push(level)
    if (i != len - 1) print_tree3(tree[i], level + 1, ppat)
    else print_tree3(tree[i], level + 1, pat)
  }
}
print_tree3(tree, 0, []);
return s;
}
```