課題１の解答例

配列 h にヒストグラムを作る

```
function kadai1(str){
    var h = []
    var i
    for (i = 0; i < 26 + 26 + 10; i++) h[i] = 0
    n = str.length
    for (i = 0; i < n; i++){
        var c = str.charAt(i)
        var ci = str.charCodeAt(i)
        if ('a' <= c && c <= 'z')
            h[ci - 'a'.charCodeAt(0)] += 1
        else if ('A' <= c && c <= 'Z')
            h[ci - 'A'.charCodeAt(0) + 26] += 1
        else
            h[ci - '0'.charCodeAt(0) + 26 + 26] += 1
    }
    return h
}

var aa = kadai1("yaaaamamoto998meijo")
puts(aa.length)
puts(aa)
```

62
5,0,0,0,1,0,0,0,1,1,0,0,3,0,3,0,0,0,0,1,0,
0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2

課題２の解答例

定義にしたがって再帰的にプログラムを作れば良い.

```
function kadai2(n, k){
    function stirling(n, k){
        if (k == 1 || n == k) return 1
        else return stirling(n - 1, k - 1) + k + stirling(n - 1, k)
    }
    function bell(n){
        var s = 0
        for (var k = 1; k <= n; k++)
            s += stirling(n, k)
        return s
    }
    return [stirling(n, k), bell(n)]
}
```

課題３の解答例

解法２の方法では以下のようになる．計算量は多いがプログラムは単純である．

```
function kadai3(n){
    var res = []
    function carpet(n, pt1, pt2){
        if (n === 0) res.push([pt1, pt2])
        else {
            var [x1, y1] = pt1
            var [x2, y2] = pt2
            var mx1 = (2 * x1 + x2) / 3
            var mx2 = (x1 + 2 * x2) / 3
            var my1 = (2 * y1 + y2) / 3
            var my2 = (y1 + 2 * y2) / 3
            carpet(n - 1, [x1, y1], [mx1, my1])
            carpet(n - 1, [x1, my1], [mx1, my2])
            carpet(n - 1, [x1, my2], [mx1, y2])
            carpet(n - 1, [mx1, my2], [mx2, y2])
            carpet(n - 1, [mx1, y1], [mx2, my1])
            carpet(n - 1, [mx2, my2], [x2, y2])
            carpet(n - 1, [mx2, my1], [x2, my2])
            carpet(n - 1, [mx2, y1], [x2, my1])
        }
    }
    carpet(n, [0, 0], [1, 1])
    return res
}
```

課題３の解答例（別解）

大きさ1/3でレベルが1小さいカーペットを描いて，それを並行移動させて描画させるプログラム．この場合，carpet関数の外で定義された変数resに結果を蓄える必要がない．

```
function kadai3(n){
    function translate(pt, s){
    var ptx = pt[0]
    var pty = pt[1]
    var sx = s[0]
    var sy = s[1]
    return [ptx + sx, pty + sy]
    }
    function translate_r(rect, s){
        var pt1 = rect[0]
        var pt2 = rect[1]
        return [translate(pt1, s),
                translate(pt2, s)]
    }
    function carpet(n, s){
        if (n === 0)  return [[[0, 0], [s, s]]]
        else {
          var aa = carpet(n - 1, s / 3)
          var res = []
          var shift = [[0, 0], [0, s / 3],
                  [0, s / 3 * 2],
                 [s / 3, 0], [s / 3, s / 3 * 2],
                 [s / 3 * 2 , 0], [s / 3 * 2, s / 3],
                 [s / 3 * 2, s/ 3 * 2]]
          for (var iss = 0; iss < shift.length; iss++){
            var ss = shift[iss]
            for (var iaa = 0; iaa < aa.length; iaa++){
              res.push(translate_r(aa[iaa], ss))
            }
          }
          return res
        }
    }
    return carpet(n, 1.0)
}
```

translateは点を平行移動のための関数であり，translate_rは四角形を平行移動させるための関数．