An ARCA script to generate a Cayley diagram for S4

S4 is the set of permutations of {1,2,3,4}
Every permutation is a product of the 3 transpositions (1,2), (2,3), (3,4)
These correspond to edges of colour R, G and B respectively in the diagram.
It is convenient to use r, g and b as names for the transpositions.

The presence of a cycle of edges RBRB reflects the fact that rbrb=1 etc

First set up a SCOUT window to display the arca diagram

```
%scout
point p1 = {10, 10};
point q1 = {490, 490};
window arca1 = {
        type:   ARCA
        box:    [p1, q1]
        pict:   "VIEW1"
        xmin:   -500
        ymin:   -500
        xmax:   500
        ymax:   500
        bgcolour: "black"
        border: 1
};
screen = <arca1>;
```

The arca file begins at this point: it has 4 principal data types
the diagram that is made up of a set of vertices and colours
a vertex of dimension n represents a point in n-dimensional space
a colour of degree n is a partial permutation of indices {1,2,...,n}
an integer has a modulus n (with special conventions for dim 0 and 1:
modulus 0 represents a standard integer that can be coerced to any
modulus, and modulus 1 represents a geometric unit)

If X is a diagram with m vertices of dimension n and colours a,b,c, ...
then the vertices of X are denoted by X!1, X!2, etc, and the colours
are denoted by a_X, b_X, c_X, ... Both vertices and colours have values
that are represented by arrays of integers: these specify the coordinates
vectors associated with vertices and the partial permutation mappings
(of degree m in the case of X) associated with the colours.

```
%arca
mode sym4 = 'abc'-diag 24
mode dia = 'ab'-diag 4

mode sym4!1 = abst vert 3
```

```
mode sym4!2 = abst vert 3
mode sym4!3 = abst vert 3
mode sym4!4 = abst vert 3

mode sym4!5 = abst vert 3
mode sym4!6 = abst vert 3
mode sym4!7 = abst vert 3
mode sym4!8 = abst vert 3
```

The definitions to this point define variables of different types, and also
define their 'mode of definition'. Thus:

sym4 is a diagram with 3 colours a,b,c and 24 vertices
There are definitions for a_sym4, b_sym4, c_sym4 and sym4!1, sym4!2, etc.
If we did not intend to define sym4 in this explicit way, we should declare
the mode of the variable instead as:
mode sym4 = abst diag

sym4!1 is a vertex of dimension 3 that will be defined by an expression
that returns an array of 3 integers as its value
If we intend to define the vertices of sym4 less abstractly, we should
declare the mode of the variable instead as:
mode sym4!1 = vert 3
and go on to specify the modes of its component integer values, e.g. as
mode sym4!1[1] = int 1; etc

```
mode sym4!9 = abst vert 3
mode sym4!10 = abst vert 3
mode sym4!11 = abst vert 3
mode sym4!12 = abst vert 3

mode sym4!13 = abst vert 3
mode sym4!14 = abst vert 3
mode sym4!15 = abst vert 3
mode sym4!16 = abst vert 3

mode sym4!17 = abst vert 3
mode sym4!18 = abst vert 3
mode sym4!19 = abst vert 3
mode sym4!20 = abst vert 3

mode sym4!21 = abst vert 3
mode sym4!22 = abst vert 3
mode sym4!23 = abst vert 3
mode sym4!24 = abst vert 3
```

We can declare vertices. colours and integers independent of any diagram:

```
mode point = abst vert
```

```
mode k = abst vert 4
mode unit = int 0
mode l = int 0
k = [20, 20, 20, 20]
l = 50
unit= 30
point = [0, 0, 0-l * unit]

sym4!1 = point + [0-unit, unit,0]
sym4!2 = point - [unit, unit,0]
sym4!3 = point - [0-unit,unit,0]
sym4!4 = point + [unit, unit,0]
```

The above formulae return arrays of integers as values, consistent with the
mode of definition of sym4!1, sym4!2, etc.

```
sym4!5 = point + [2*unit, 4*unit,0]
sym4!6 = point + [4*unit, 4*unit,0]
sym4!7 = point + [4*unit, 2*unit,0]
sym4!8 = point + [2*unit, 2*unit,0]

sym4!9 = point - [6*unit, 0-6*unit,0]
sym4!10 = point + [6*unit, 6*unit,0]
sym4!11 = point + [6*unit, 0-6*unit,0]
sym4!12 = point - [6*unit, 6*unit,0]

sym4!13 = point - [4*unit, 2*unit,0]
sym4!14 = point - [2*unit, 2*unit,0]
sym4!15 = point - [2*unit, 4*unit,0]
sym4!16 = point - [4*unit, 4*unit,0]

sym4!17 = point + [0-4*unit, 4*unit,0]
sym4!18 = point + [0-2*unit, 4*unit,0]
sym4!19 = point + [0-2*unit, 2*unit,0]
sym4!20 = point + [0-4*unit, 2*unit,0]

sym4!21 = point + [2*unit, 0-2*unit,0]
sym4!22 = point + [4*unit, 0-2*unit,0]
sym4!23 = point + [4*unit, 0-4*unit,0]
sym4!24 = point + [2*unit, 0-4*unit,0]

mode a_sym4 = abst col
mode b_sym4 = abst col
mode c_sym4 = abst col
mode a_dia = abst col
mode b_dia = abst col
```

The colours of diagram sym4 are abstractly defined by expressions to return arrays
of integers that represent perms of {1,2, ..., 24}; those of diagram
dia also return arrays of integers, but these are perms of {1,2,3,4}.

```
a_dia = {1,2}${3,4}
b_dia = {1,4}${2,3}
```

a_dia represents the mapping taking 1 to 2, 2 to 1, 3 to 4 and 4 to 3 *etc*

```
a_sym4 = a_dia :: a_dia :: a_dia :: a_dia :: a_dia :: a_dia
b_sym4 = b_dia :: b_dia :: b_dia :: b_dia :: b_dia :: b_dia
```

:: is a special operator that generates a perm of {1,2,...,24} by 'replicating' a partial permutation of {1,2,3,4} six times (applying the same pattern to {1,2,3,4}, {5,6,7,8}, {9,10,11,12}, ..., {21,22,23,24} and composing these perms).

```
c_sym4 =
      {2,14}${6,10}${4,8}${12,16}${1,19}${3,21}${5,18}\
            ${7,22}${9,17}${11,23}${13,20}${15,24}
```

```
display 'abc'-sym4 on VIEW1 with labels
```

This simple ARCA file only gives a flavour of the possibilities. Note that in principle the mode of definition can itself be defined by an expression that involves applying operators to modes to create new modes. The mode resembles a template for value definition.