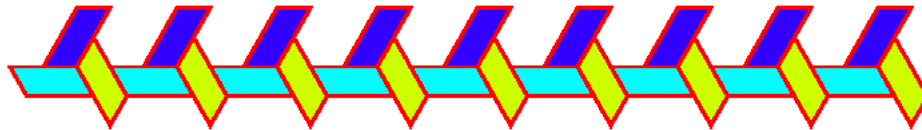


## QUIZ #5

### ESCHER WITH POSTSCRIPT

Here you will have to demonstrate your newly acquired “deep understanding” of the `POSTSCRIPT` language to create a beautiful display of a braded pattern obtained by juxtaposing a sequence of rows similar to the one given below.



You may do it any way you want as long as you get brades. But if you prefer to be guided, you will find in the following the sequence of steps that will produce the desired result.

#### Step 1

Write a `POSTSCRIPT` procedure “**losenge**” that draws the top losenge of the following triple



Given that the size of the small side is  $a$ , its four vertices are

$$(0, 0) , \quad (2a \cos(60^\circ), (2a \sin(60^\circ))) , \quad (2a \cos(60^\circ) - a, (2a \sin(60^\circ))) , \quad (-a, 0)$$

Here the origin  $(0, 0)$  gives the center of the triple. Recall that a `POSTSCRIPT` procedure by name **losenge** with input parameters “**a**” (the size) and “**u, v, w**” (the rgb the color parameters) has the structure

```

/losenge
{
/w  exch  def
/v  exch  def
/u  exch  def
/a  exch  def
command1
command2
.....
commandn
} def

```

Now the sequence of `POSTSCRIPT` commands that depicts a polygon with vertices

$$(a_0, b_0) , (a_1, b_1) , (a_2, b_2) , \dots , (a_r, b_r)$$

filled with rgb color parameters  $u, v, w$  (all between 0 and 1) is

```

a0b0  moveto
a1b1  lineto
a2b2  lineto
.....
arbr  lineto
closepath
u v w  setrgbcolor  fill

```

That only fills the polygon with the chosen color, if you want also a colored boundary you must repeat the same commands but at the end replace the word “**fill**” by the sentence “**x setlinewidth stroke**”, (here  $x$  denotes the thickness of the line, which will be drawn  $x/72$  inches thick.) Also make sure you change the color for the boundary otherwise you will not see it.

Note after you have written the procedure test it by viewing a page with the following commands

```
200 200 translate
40 losenge
```

### Step 2

Next write a procedure by name of “**triple**” that draws the triple illustrated above with input parameters  $x$ ,  $y$ ,  $a$ , giving the coordinates of the center of the triple and its size. Remember that in declaring input parameters within a `POSTSCRIPT` procedure you must place them in reverse order. All you need to do in **triple** is, first translate the origin to the point  $(x, y)$  by the call “**x y translate**” then make the call “**a u<sub>1</sub> v<sub>1</sub> w<sub>1</sub> losenge**” then “**120 rotate**” then call “**a u<sub>2</sub> v<sub>2</sub> w<sub>2</sub> losenge**” then again “**120 rotate**” and finally “**a u<sub>3</sub> v<sub>3</sub> w<sub>3</sub> losenge**” to get the different colors you want to give the three losenges, remember to precede all these calls by “**gsave**” and follow them up with “**grestore**”. Once this procedure is written, test it with the command

```
200 200 40 triple
```

### Step 3

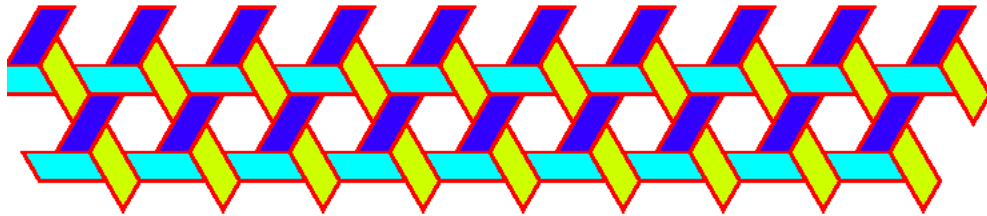
Next write a procedure “**row**” which draws a row of “triples”, with input parameter “**h**” that gives the “height” at which the row will be drawn. . You can get a row of “triples” by means of `POSTSCRIPT` command “**for**”. Remember that the structure of this command is

```
a b c { blahblah } for
```

and “**blahblah**” will be repeated with an input parameter that goes from  $a$  to  $c$  by steps of  $b$ . I suggest to carry out a test to assure that you get something that looks like the row given above.

### Step 4

Now you are almost done. You will only have to copy the commands of procedure “**row**” and make the simple changes needed to construct a procedure “**srow**” which draws a second row on top of the first row as in the figure below



### Step 4

Now by another use of the `POSTSCRIPT` command “**for**” and recalling that **row** and **srow** draw the sequences of triples at the height given by their input parameter  $h$ , you will be able to fill the page with an intricate set of “**brades**”, showing `POSTSCRIPT` in its full glory.