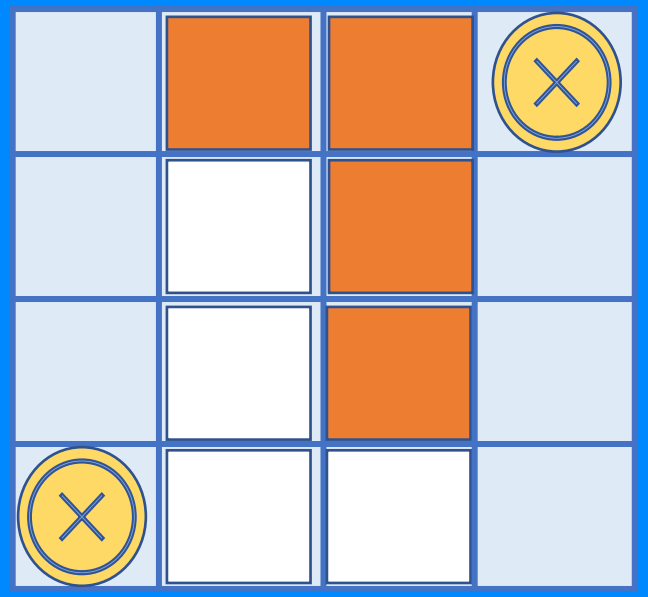
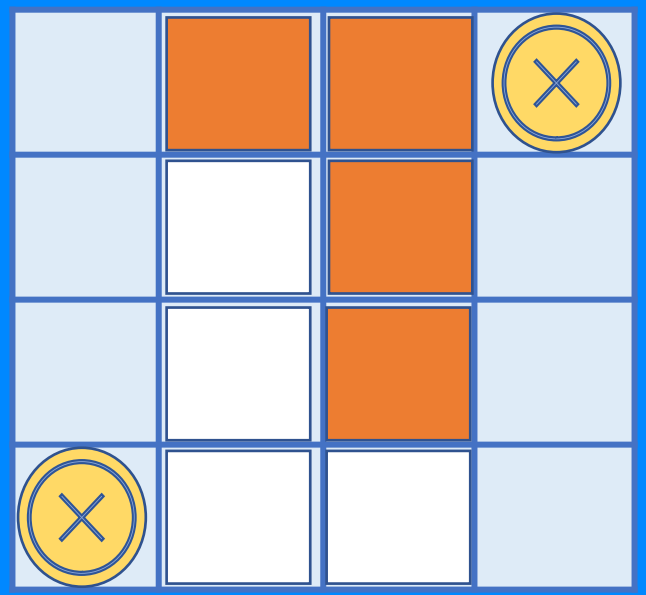
Start position and game rules for the L-Game.



The L-Game is a two player game.
Goal is to move your 4 square pieces and the coins on the board in such a position that your opponent can not create an L-shape anymore.

- In each turn one coin may be moved
- In each turn at least one square piece needs to be in a different position
- A move is only valid if an L-shape if formed in the turn.

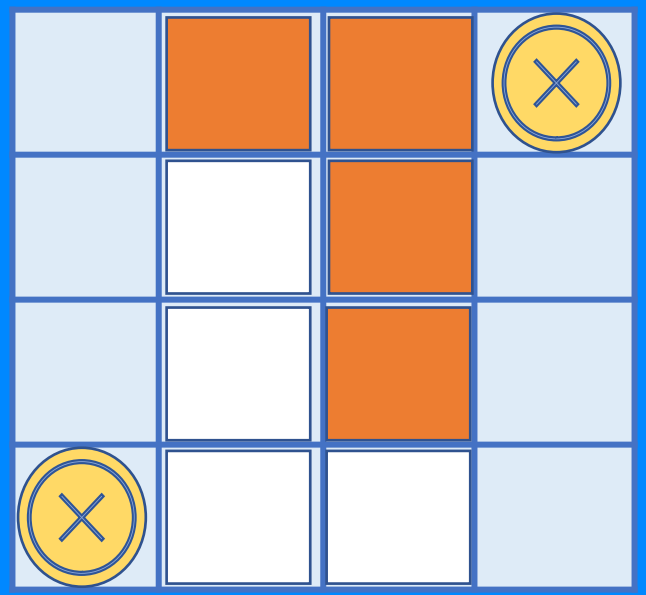
What routines do we need in the Basic program



Main routines needed are:

- ❑ A gameboard a square of 4x4 cells
- ❑ A pointer
- ❑ Joystick steering and selection
- ❑ Mapping the pointer to the cell
- ❑ Filling and removing the cell
- ❑ User input for game selection
- ❑ Validation if a L shape is formed
- ❑ Only free or own cells are used
- ❑ Display the previous L shape

I Start with the most challenging one



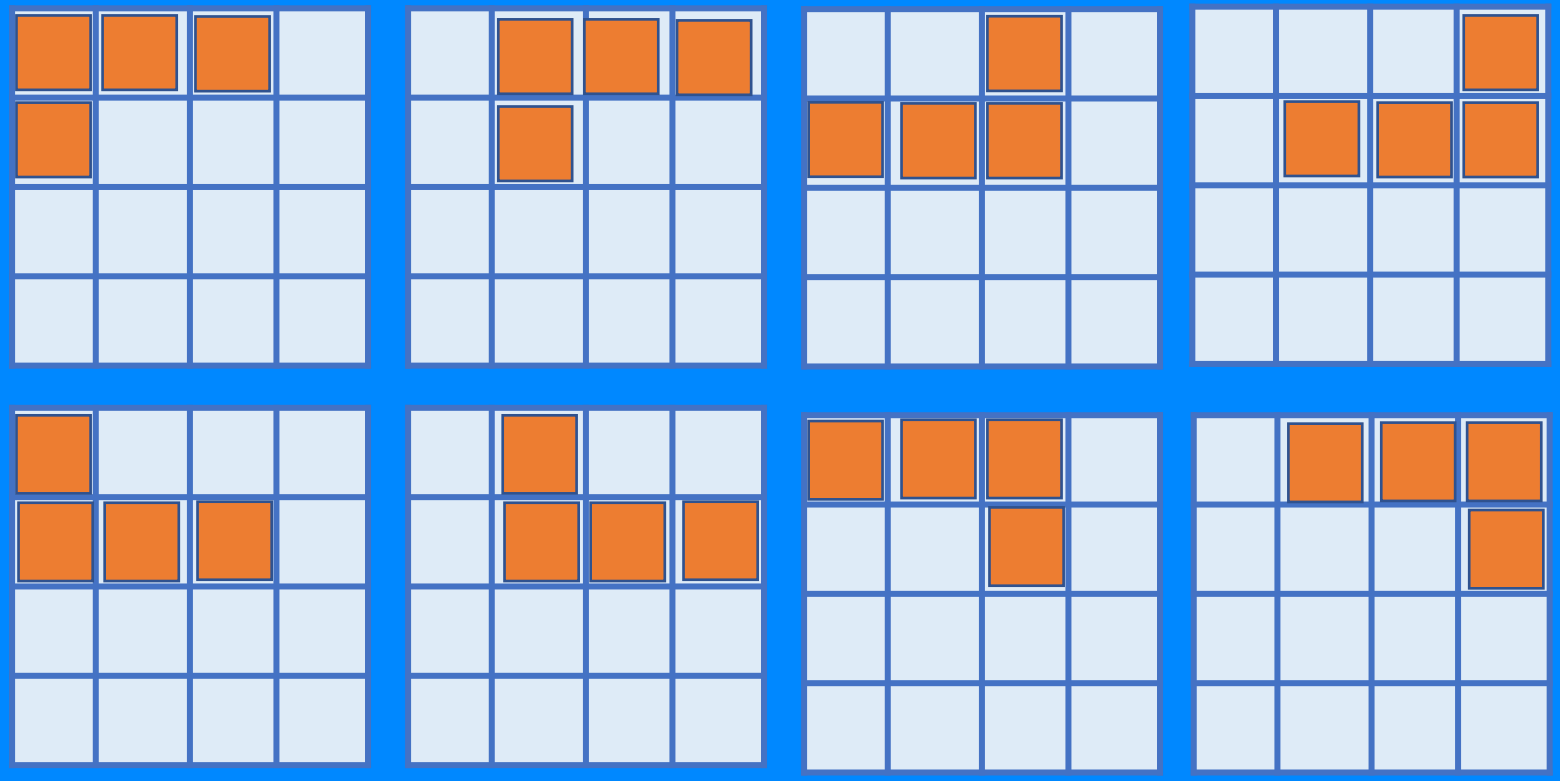
How to determine if a pattern L is formed by a player.

First I just drawn a lot of patterns on paper just to get a feeling about it.

Let us start with the horizontal ones.

I Start with the most challenging one

Horizontally the following patterns can be drawn.



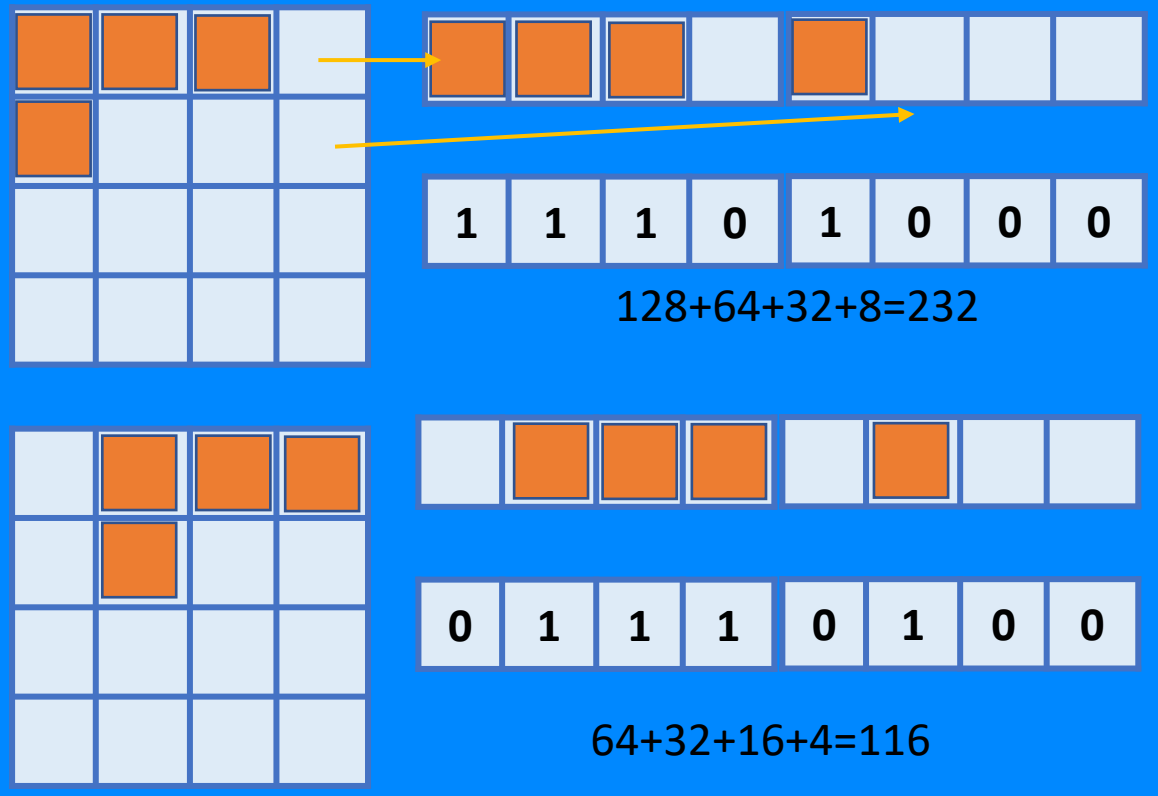
These can of course also be made a on the second and again on the third row of the matrix.

also the patterns can be made vertically.

This means a lot of different possibilities to validate.

A Lot of combinations. Can we do it more smart?

I did not want to define al possible positions and then do the validation.. So...pattern folding

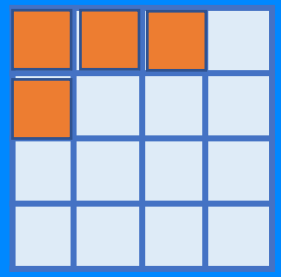


I came up with the idea to see the pattern as a bit/byte pattern.

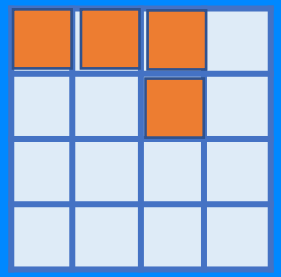
So I combined the first 8 cells and represented that as bit values of a byte.

For the horizontal pattern I now only need 8 combinations. And.. I have another trick for the vertical ones.

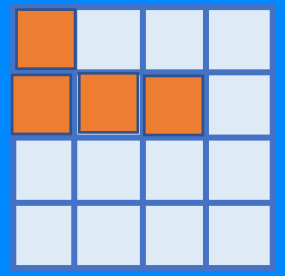
So lets calculate them all



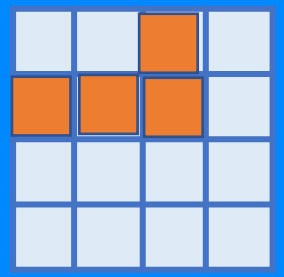
128+64+32+8=232



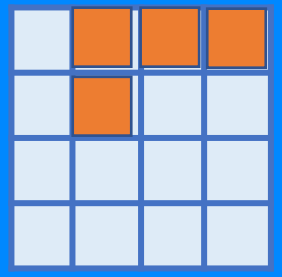
128+64+32+2=226



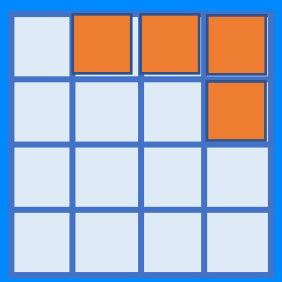
128+8+4+2=142



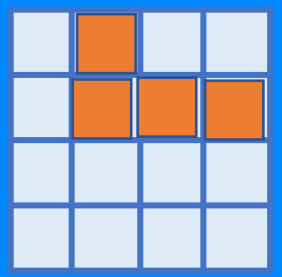
32+8+4+2=46



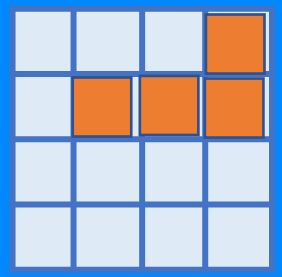
64+32+16+4=116



64+32+16+1=113



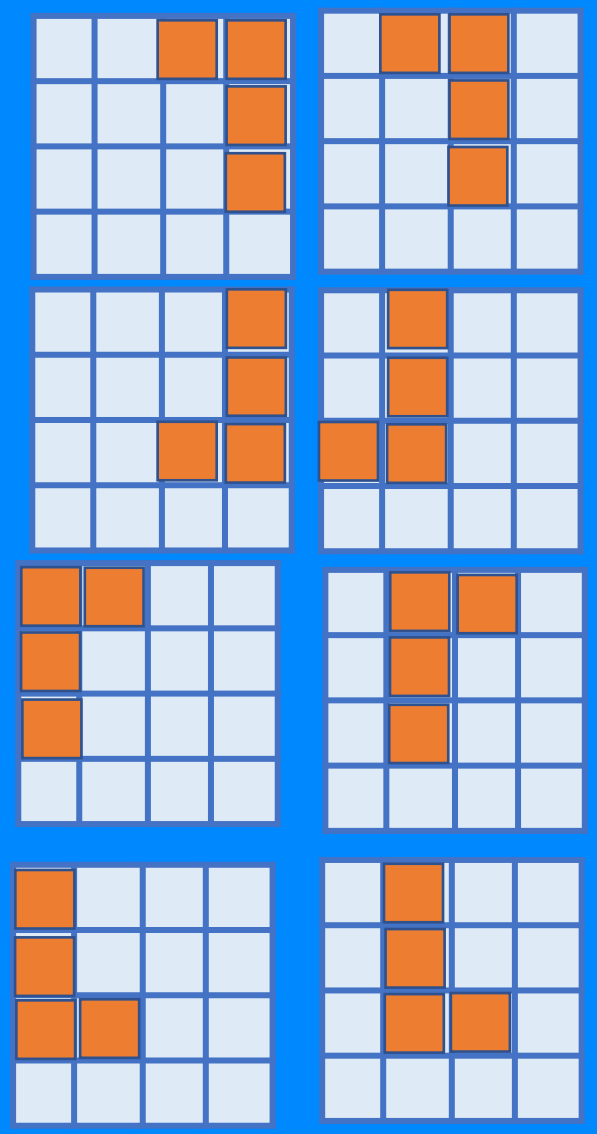
64+4+2+1=71



16+4+2+1=23

Eight values are:
232, 226,142,46,
116,113,71,23

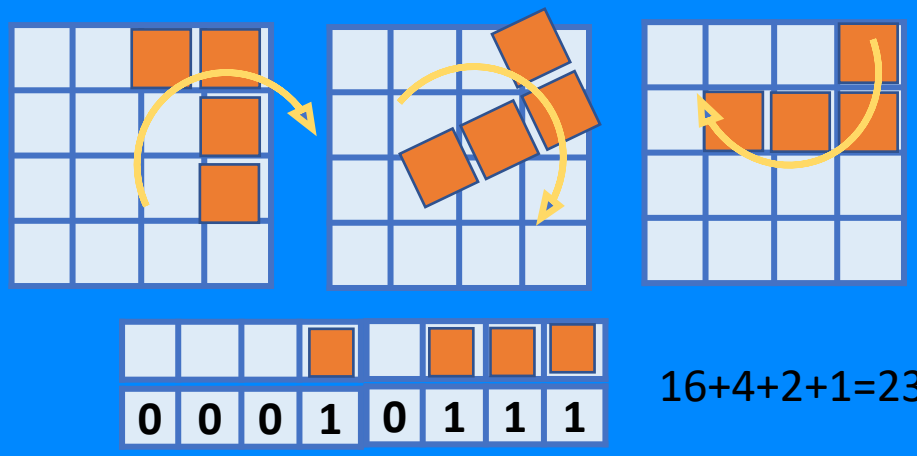
Mmm ok, but what about the horizontal patterns?



So what do we do with these ? How can we validate if those pattern are in the matrix?

Well actually these are the same depending how you look at them.

If you rotate them you get the horizontal equivalent of it. Let look at the first one and rotate it.



16+4+2+1=23

Now this one is exactly the same as the one on the previous page.

Let see if we define that in Basic code

So what do we do with these ? How can we validate if those pattern are in the matrix?

Before we start with the complete program it is always good to first start with only the program concept to see if you can solve the main problems. Decorating everything is something that needs to be done after that. If not… then you could end up with a program that will never be finished 😁

What do we need

- A structure for the player positions. We define a 4x4 array for that.
- A list with values to validate if a player has his pieces in a l-shape.
- A routine to convert the positions into a byte (which we need for the pattern validation)
- A routine to just print the array (for debug/validation)
- A routine to fill the 4x4 array.
- A routine to rotate the positions so that we can also validate the horizontal positions