Advanced programming task

Conway's Game of Life
The **Game of Life** is a cellular automaton devised by the British mathematician John Horton Conway in 1970.

The universe of the Game of Life is an two-dimensional orthogonal grid of square *cells*, each of which is in one of two possible states, *alive* or *dead*. Every cell interacts with its eight *neighbours*, which are the cells that are horizontally, vertically, or diagonally adjacent.

At each step in time, the following transitions occur:

- Any live cell with fewer than two live neighbours dies, as if caused by under-population.
- Any live cell with two or three live neighbours lives on to the next generation.
- Any live cell with more than three live neighbours dies, as if by over-population.
- Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

The initial pattern constitutes the *seed* of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed—births and deaths occur simultaneously, and the discrete moment at which this happens is sometimes called a *tick* (in other words, each generation is a pure function of the preceding one). The rules continue to be applied repeatedly to create further generations.

*(source: wikipedia.org)*
Tasks and steps

• Initialize a world matrix. Dimension and relative amount of living cells shall be variable.

• Visualize the world.

• Write a function counting living cells in neighborhood of a given position (no toroidal borders).

• Write a loop realizing changes of the world in each tick. Recall that all events occur simultaneously. Plot the world in each tick. Hint: `Sys.sleep(s)` pauses execution for s seconds.
A solution

Initialize a world matrix. Dimension and relative amount of living cells shall be variable:

```r
> dim.xy <- 50
> perc.alive <- 50

> world <- matrix( sample( c( rep( 1, perc.alive ), rep( 0, 100-perc.alive ) ),
                        dim.xy^2, replace=TRUE ),
                        dim.xy, dim.xy )
```

Visualize the world:

```r
> image( world, axes=FALSE, col=c( "white", "blue" ), main=paste( "alive:", sum( world == 1 ) ) )
> box()
```
A solution

Write a function counting living cells in neighborhood of a given position (no toroidal borders):

```r
> sum.living.neighbors <- function( x, y, world )
{
  s <- 0

  if( x > 1 ) s <- s + world[ x-1, y ]
  if( x < ncol(world) ) s <- s + world[ x+1, y ]
  if( y > 1 ) s <- s + world[ x, y-1 ]
  if( y < nrow(world) ) s <- s + world[ x, y+1 ]

  if( x > 1 && y > 1 ) s <- s + world[ x-1, y-1 ]
  if( x < ncol(world) && y > 1 ) s <- s + world[ x+1, y-1 ]
  if( x > 1 && y < nrow(world) ) s <- s + world[ x-1, y+1 ]
  if( x < ncol(world) && y < nrow(world) ) s <- s + world[ x+1, y+1 ]

  return( s )
}
```
A solution

Write a loop realizing changes of the world in each tick. Recall that all events occur simultaneously. Plot the world in each tick:

```r
> time <- 1
> while(TRUE)
> {
>   image( world, axes=FALSE, col=c("white", "blue"), main=paste("time:", time, " alive:", sum( world == 1) ) )
>   box()
>
>   new.world <- world
>   for( i in c( 1:dim.xy ) )
>     for( j in c( 1:dim.xy ) )
>     {
>       n.neig <- sum.living.neighbors( i, j, world )
>
>       if( world[i,j] == 0 && n.neig == 3 ) new.world[i,j] <- 1
>       if( world[i,j] == 1 && n.neig <  2 ) new.world[i,j] <- 0
>       if( world[i,j] == 1 && n.neig == 2 ) new.world[i,j] <- 1
>       if( world[i,j] == 1 && n.neig == 3 ) new.world[i,j] <- 1
>       if( world[i,j] == 1 && n.neig >  3 ) new.world[i,j] <- 0
>     }
>   world <- new.world
>   time <- time + 1
>   Sys.sleep( 0.1 )
> }
```