CSC 121 — Lab Exercise 4

This is a non-credit exercise, which you do not hand in.
You may work on your own or together with another student, as you please.

In this lab, you will practice working with matrices, and plotting them with the `persp` and `contour` functions.

Exercise 1: Write a function called `lower_one` that takes an integer `n` as its argument, and returns a matrix with `n` rows and `n` columns that has zero above the diagonal and one on and below the diagonal.

For example, here is the result of a call of this function:

```
> lower_ones(5)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    1    1    0    0    0
[3,]    1    1    1    0    0
[4,]    1    1    1    1    0
[5,]    1    1    1    1    1
```

You should write this function using two `for` loops. (When we cover some more advanced R faciltiies, you'll see how one could write it more simply.)

Exercise 2: Write an R function called `X_matrix` that takes one argument, called `n`, that you should assume is a positive integer, and that returns an `n` by `n` matrix of numbers, which are zero except that they are one on the diagonal from the top left to bottom right and on the diagonal from the top right to the bottom left.

Here are two example calls of `X_matrix`:

```
> X_matrix(4)
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    1
[2,]    0    1    1    0
[3,]    0    1    1    0
[4,]    1    0    0    1
> X_matrix(5)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    1
[2,]    0    1    0    1    0
[3,]    0    0    1    0    0
[4,]    0    1    0    1    0
[5,]    1    0    0    0    1
```

Exercise 3: In this exercise, you will simulate how heat flows in a rectangular plate whose edges are kept at specified temperatures. The temperatures at positions in the plate are represented by a matrix with `nr` rows and `nc` columns.

You should first write a function called `initial_temperatures` that takes `nr` and `nc` as argments and returns a matrix with those dimensions in which all values are zero except for the last row and last column, which should contain ones. For example:

```
> initial_temperatures(4,5)
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    1
[2,]    0    0    0    0    1
[3,]    0    0    0    0    1
[4,]    1    1    1    1    1
```

You should then write a function called `heat_flow` that takes a matrix, `M`, as its argument and returns a matrix of the same dimensions as `M`. The matrix returned should have the same values as `M` in its first and last rows and columns, which represent the edges of the plate, where the temperatures are assumed to be kept fixed. Each other value in the matrix returned should be the average of the value at that position in `M` and the four vertically and horizontally adjacent values in `M`. This represents heat flowing to and from neighboring positions in the plate. For example:

```
> heat_flow (initial_temperatures(4,5))
     [,1] [,2] [,3] [,4] [,5]
[1,]    0  0.0  0.0  0.0    1
[2,]    0  0.0  0.0  0.2    1
[3,]    0  0.2  0.2  0.4    1
[4,]    1  1.0  1.0  1.0    1
```

Note: You can use the `nrow` and `ncol` functions to find out how many rows and columns there are in a matrix.

Finally, you should write a script that sets a variable to the matrix representing the initial temperatures for a 30×30 plate, and then repeatedly (in a `for` loop) replaces this variable by the ¡result of calling `heat_flow`, doing this for some number of iterations. The final result should be plotted using `persp`, which you can call with just the matrix of temperatures as its argument (the grids for x and y will default to something not displayed). You can also play around with setting the viewing angles with the `phi` and `theta` options to get a good view of the function.

You can also try plotting the temperatures with the `contour` function (which can also be called with just the matrix of temperatures as its argument).

You might find it interesting to see what the temperatures look like after 10 iterations of heat flow, after 100 iterations, and after 1000 iterations. The result for a large number of iterations will be an approximate solution to a well-known partial differential equation that descibes how heat is distributed after a steady state is reached.