SI 6: Function Composition With Modular Arithmetic

1 Integer Division

1. Using modular arithmetic, define integer division as it is implemented in most programming languages. That is: a function $Idiv : \mathbb{R} \times \mathbb{R} \to \mathbb{I}$ such that $Idiv(4, 2) = 2$ and $Idiv(3.2, 2) = 1$ (note that decimal remainders are truncated). This function should take no more than one line.

2 Matrix Operations

Imagine you have been tasked with designing an indexing scheme for dealing with a 2D array (i.e a Matrix), where very intensive matrix operations will be carried out. In order to simplify the operations, you will have to carefully construct your indexing function so that every cell can be encoded by a single number n, rather than the usual coordinates (i, j).

2. Design your indexing scheme such that each cell is encoded by one number, and each number can only correspond to one cell (drawing a sample matrix with the ordering scheme will suffice).

3. Write a function $getIndex : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ that extracts the indices (i, j) from the number n of a given cell. Recall that you must be able to go back in forth between a number n and indices (i, j), which implies you function must be 1-1 in the domain $D : (0, n_{max})$ where n_{max} is the value of the highest-valued cell.

4. Now define the inverse of $getIndex$, where an index (i, j) is encoded into a single number n. Again, you must be 1-1 in the domain
$D : \{(i, j) : i, j \in \mathbb{N},\ 0 \leq i < h, 0 \leq j < w\}$ where h is the height and w the width of your matrix.