BarChart (harder)

Given a vector of numbers, produce a horizontal bar chart of hash characters for how many numbers fit into each of ten equal-sized groups. For example, if the data ranges from 0-100, the ranges will be 0-9.9, 10-19.9, etc. (Formally, [0,10), [10,20), etc.). You may assume that there will be at least two numbers and that not all numbers will be the same.

Example:

```
                BarChart ⎕AVU/⍨⎕AVU<255
#############
######################
##########################
#########################
#########################
#
########
################
########################
##########################
      BarChart ⎕AVU/⍨⎕AVU>1E3
#
####
#########
############
######
#########################


###########
#
```

DiseaseSpread

Given a Boolean matrix world, generate the next iteration where

- any cell which is rectangularly adjacent to (share a side with) an infected cell is infected.
- infected cells stay infected forever.

Example:

```
      DiseaseSpread 2⊥⍣¯1⊢6↑⎕AVU
0 0 0 1 1 1
0 1 1 1 1 1
1 1 1 1 1 1
0 1 1 1 1 1
0 1 1 1 0 1
0 0 1 1 1 0
      DiseaseSpread 1=4⊥⍣¯1⊢¯8↑⎕AVU
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 0 0 0 0 1 1
1 1 1 0 0 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0
1 1 1 0 1 1 1 0
```

KnightMovesFrom (harder)

Given:

- a number of turns
- a simple 2-element vector indicating a starting position,
  or
  a vector of two or more starting positions,

on an 8-by-8 chess board, return the positions that knight(s) can be at after the given number of turns.

- Each knight must move with every turn.
- A knight can only move to the positions marked with X relative to its current position, marked with ♞:

Example:

```
      1 KnightMovesFrom 1 1
 2 3  3 2
      2 KnightMovesFrom 1 1
 1 1  1 3  1 5  2 4  3 1  3 5  4 2  4 4  5 1  5 3
      1 KnightMovesFrom (1 1) (5 7)
 2 3  3 2  3 6  3 8  4 5  6 5  7 6  7 8
```

	X		X	
X				X
		♞		
X				X
	X		X	

CompleteTeams

Given a vector of each participant's team number, return the teams numbers that have exactly two members.

Examples:

```
      CompleteTeams 1 4 1 5 9 2 6 5
1 5
      CompleteTeams 5 5,⍳10

      CompleteTeams 7 1 8 2 8 1 8 2 8
1 2
```

GameOfLife (harder)

Given a Boolean matrix world, generate the next generation where

- any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- 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 overpopulation.
- any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
- there are hard walls (equivalent to always dead cells) surrounding the world.

You may assume the world has at least three rows and three columns.

Example:

```
      GameOfLife 2⊥⍣¯1⊢6↑⎕AVU
0 0 0 0 0 0
0 0 1 1 1 0
0 0 1 1 0 0
0 1 0 1 0 0
0 0 1 1 1 0
0 0 0 0 0 0

      GameOfLife 2⊥⍣¯1⊢8↑⎕AVU
0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0
0 0 1 1 0 1 0 0
0 1 0 1 0 1 0 1
0 0 1 1 1 1 0 0
0 0 0 0 0 0 1 1
```

MonadicKey

Write a limited model of αα⌸ω: For each unique element of ω, call αα with the unique element as left argument and the indices of that unique element in ω as right argument, then Mix (↑ω) the result. Assume ω is a simple vector.

Examples:

```
      {⊂ω}MonadicKey 2 7 1 8 2 8 1 8 2 8
 1 5 9  2  3 7  4 6 8 10
      {α,≢ω}MonadicKey'Mississippi'
M 1
i 4
s 4
p 2
```

IntervalIndex

Write a limited model of α⍸ω: For each element of ω, find which "gap" it belongs in α:

- 0 for ω[i] means ω[i]<α[1]
- 1 for ω[i] means α[1]≤ω[i] and ω[i]<α[2], etc.
- 2 for ω[i] means α[2]≤ω[i] and ω[i]<α[3], etc.

Assume that α and ω are vectors with the same datatype and that α is sorted, duplicate-free, and has at least one element.

Examples:

```
      1 4 5 9 IntervalIndex 1 2 7 8 0
1 1 3 3 0
      'aegilops' IntervalIndex 'goatgrass'
3 6 1 8 3 7 1 8 8
```

IsomorphIn

Given two simple arrays of the same rank (1 or higher), determine whether the left argument is an isomorph sub-array of the right argument. Two arrays are isomorphic if they have the same pattern of repetitions. For example, both `'ESTATE'` and `'DUELED'` have pattern `abcdca`. In other words, you need to check if there exist vectors `a` and `b` such that `⍺` is isomorphic with `a↑b↓⍵`.

Examples:

```
      'adca' IsomorphIn 'ddaddabdaabbcc'
1
      'adac' IsomorphIn 'ddaddabdaabbcc'
0
```

FillSteps

Given a simple Boolean array return a vector of thusly shaped arrays where the leftmost array is identical to the argument and the rightmost is all ones. All intermediary steps must have one more 1 then its neighbor to the left. For each step, the bit that is changed must be randomly chosen.

Examples (your results may vary):

```
      FillSteps 0 1 0 0
┌───────┬───────┬───────┬───────┐
│0 1 0 0│0 1 1 0│1 1 1 0│1 1 1 1│
└───────┴───────┴───────┴───────┘
      FillSteps 0 1 0 0
┌───────┬───────┬───────┬───────┐
│0 1 0 0│0 1 0 1│0 1 1 1│1 1 1 1│
└───────┴───────┴───────┴───────┘
      FillSteps 2 3⍴0 1 0 0
┌─────┬─────┬─────┬─────┬─────┐
│0 1 0│1 1 0│1 1 0│1 1 1│1 1 1│
│0 0 1│0 0 1│0 1 1│0 1 1│1 1 1│
└─────┴─────┴─────┴─────┴─────┘
      FillSteps 2 3⍴0 1 0 0
┌─────┬─────┬─────┬─────┬─────┐
│0 1 0│0 1 1│0 1 1│0 1 1│1 1 1│
│0 0 1│0 0 1│1 0 1│1 1 1│1 1 1│
└─────┴─────┴─────┴─────┴─────┘
```