Spiral Matrix (Scheme+Haskell - 7+7 Points)

June 16, 2022

A *Spiral Matrix* is a square $n \times n$-matrix filled with natural numbers, starting from 1 in the top-left corner, increasing in inward, clockwise spiral order, like these examples:

$$\mathbf{S}_3 = \begin{pmatrix} 1 & 2 & 3 \\ 8 & 9 & 4 \\ 7 & 6 & 5 \end{pmatrix} \qquad \mathbf{S}_5 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 16 & 17 & 18 & 19 & 6 \\ 15 & 24 & 25 & 20 & 7 \\ 14 & 23 & 22 & 21 & 8 \\ 13 & 12 & 11 & 10 & 9 \end{pmatrix}$$

Even though one can define a spiral matrix for each size n, your task is to implement a function generating a spiral $n \times n$-matrix only for odd n. Note that such matrices can be generated recursively because the spiral matrix $\mathbf{S}_n$ of size $n \times n$ can be constructed from the spiral matrix $\mathbf{S}_{n-2}$ of size $(n-2) \times (n-2)$ as follows:

$$\mathbf{S}_n = \begin{pmatrix} 1 & 2 & \cdots & n-1 & n \\ 4n-4 & & & & n+1 \\ \vdots & & \mathbf{B} & & \vdots \\ 3n-1 & & & & 2n-2 \\ 3n-2 & 3n-3 & \cdots & 2n & 2n-1 \end{pmatrix}$$

where $\mathbf{B}$ is basically the matrix $\mathbf{S}_{n-2}$ whose all elements are increased by $4n-4$.

Task 3 - Scheme

In Scheme, implement a function (`spiral-matrix n`) that accepts a positive odd integer and returns the spiral matrix $\mathbf{S}_n$ of size `n`. A matrix is represented as a list of its rows, e.g. $\mathbf{S}_3$ is represented as

```
'((1 2 3) (8 9 4) (7 6 5))
```

Your file is to be called `task3.rkt` and must provide the function `spiral-matrix`. Hence, the head of your file should start with

```
#lang racket
(provide spiral-matrix)

; your code here
```

Hint

To generate a sequences of numbers, you may use the function `(range start end step)` generating a sequence of numbers starting at `start` and whose successive elements are computed by adding `step` until `end`. Note that the element `end` is excluded. E.g.

```
> (range 5 0 -1)
'(5 4 3 2 1)
```

Examples

The following shows the behaviour of the `spiral-matrix` function.

```
> (spiral-matrix 3)
'((1 2 3) (8 9 4) (7 6 5))
```

```
> (spiral-matrix 1)
'((1))
```

Task 4 - Haskell

In Haskell, implement a function `spiralMatrix :: Int -> Matrix` that accepts a positive odd integer n and returns the spiral matrix $\mathbf{S}_n$ of size n. A matrix is represented as a list of its rows by data type `type Matrix = [[Int]]`, e.g. $\mathbf{S}_3$ is represented as

```
[[1, 2, 3], [8, 9, 4], [7, 6, 5]]
```

Your task is to be called `Task4.rkt` and must export the `spiralMatrix` function. Hence, the head of your file should read

```
module Task4 ( spiralMatrix ) where
type Matrix = [[Int]]

-- your code goes here
```

Hint

You may find the function `zipWith3 f xs ys zs` useful which processes three lists `xs`, `ys`, `zs` simultaneously applying the ternary function `f` to the triples of corresponding elements, e.g.,

```
> zipWith3 (\c1 c2 c3 -> [c1,c2,c3]) "abcd" "blaa" "haha"
["abh","bla","cah","daa"]
```

Generating a sequence of numbers in Haskell can be done by `[start,start+step..end]`, e.g.,

```
> [5,4..1]
[5,4,3,2,1]
```

In comparison with the Scheme function `range`, the final number `end` is included in the generated sequence.

Examples

The following shows the behaviour of the `spiralMatrix` function.

```
> spiralMatrix 3
[[1,2,3],[8,9,4],[7,6,5]]
```

```
> spiralMatrix 1
[[1]]
```