Task 2: Generalized Pascal Triangle (Haskell - 8 Points)

June 18, 2021

Introduction

The Pascal Triangle (or Tartaglia's triangle) is a well-known mathematical object. It is an infinite triangle of integers with 1 as a vertex. Each element of the next rows composing the triangle is obtained by summing the two above entries of the previous row. At any row, the extreme left and right entries are set equal to the vertex. "A picture is worth a thousand words", as the adage goes...

row 0:						1						
row 1:					1		1					
row 2:				1		2		1				
row 3:			1		3		3		1			
row 4:		1		4		6		4		1		
row 5:	1		6		10		10		6		1	
	...		...		...		...		...		...	

Figure 1: The classical Pascal triangle

One can easily extend Pascal triangle construction to other operations, possibly starting with a different integer vertex. For example, if we use the multiplication as operation and choose 2 as vertex we get:

row 0:						2						
row 1:					2		2					
row 2:				2		4		2				
row 3:			2		8		8		2			
row 4:		2		16		64		16		2		
row 5:	2		32		1024		1024		32		2	
	...		...		...		...		...		...	

Figure 2: A Pascal triangle constructed using multiplication as operation and 2 as vertex

We could also use the subtraction as operation, but for symmetry's sake let us consider the *symmetric difference* between two integers $m, n \in \mathbb{Z}$ defined as $m\Delta n = n\Delta m := (m-n)(n-m) = 2mn - m^2 - n^2$ If we now choose -2 as vertex and use the symmetric difference as an operation we get:

row 0:			-2				
row 1:			-2	-2			
row 2:		-2	0	-2			
row 3:		-2	-4	-4	-2		
row 4:		-2	-4	0	-4	-2	
row 5:	-2	-4	-16	-16	-4	-2	
	...	...	...	...	...	...	...

Figure 3: A Pascal triangle constructed using the symmetric difference as operation and -2 as vertex

Assignment

As a first part of this task, write a function `pascal :: (Int -> Int -> Int) -> Int -> Int -> [Int]` that takes a binary operation between integers `op :: Int -> Int -> Int`, a given vertex `v :: Int` and a row number `r :: Int`, and returns as a **list of integers** the r^{th} row of the Pascal triangle constructed with operation `op` and vertex `v`. For example:

```
> pascal (+) 1 3
[1,3,3,1]

> pascal (*) 2 4
[2,16,64,16,2]

sym_diff :: Int -> Int  -> Int
sym_diff n m = (n-m)*(m-n)

> pascal sym_diff -2 5
[-2,-4,-16,-16,-4,-2]
```

Warning: the row indexing starts from 0 at the level of the vertex.
As a second part, add an IO interface `main :: IO ()` to your code which works as follows:

1. prints on screen the string `"Enter operation:"`
2. takes as input a string among `"add"`, `"mul"`, `"sym_diff"` selecting the operation
3. print on screen the string `"Enter vertex:"`
4. takes as input a string representing an integer of any digits number, e.g. `"-123"`
5. prints on screen the string `"Enter row:"`
6. takes as input a string representing positive integer of any digits number, e.g. `"11"`
7. prints on screen the list which is the result of the function `pascal` evaluated on the three inputs inserted.

You may assume only valid inputs.

Examples

```
> main
Enter operation:
sym_diff
Enter vertex:
-2
Enter row:
5
[-2,-4,-16,-16,-4,-2]

> main
Enter operation:
add
Enter vertex:
1
Enter row:
2
[1,2,1]

> main
Enter operation:
mult
Enter vertex:
2
Enter row:
3
[2,8,8,2]
```