Problem Set 1 - Haskell

Ana Neri
February 26, 2021

The goal of the problem set 1 is to recall programming with Haskell and some essential concepts of quantum computing mathematical framework.

1. Write a Haskell function that:
   - (a) Calculates the perimeter of a circle given its radius.
   - (b) Calculates the area of a circle given its radius.

2. Write a Haskell function to calculate the factorial of a number n. Remember that by convention, $fac(0) = 1$.

3. Prime numbers can be an advantage in many fields, such as cryptography.
   - (a) Write a function that returns all numbers from 2 to a given number n.
   - (b) Implement a function to eliminate the multiples of a number n from a list.
   - (c) Admit the Eratosthenes sieving algorithm:
     - It starts with a list of numbers from `2..n`.
     - The algorithm iteratively takes out the multiples of the prime elements.
     - In each iteration, the prime number to use always lies in the following position of the element used in the previous iteration.

     ```
     Initial list: [2,3,4,5,6,7,8, 9,10]
     Iteration 1: [2,3,5,7,9] // eliminate the multiples of 2
     Iteration 2: [2,3,5,7] // eliminate the multiples of 3
     Iteration 3: [2,3,5,7] // eliminate the multiples of 5.
     (...)
     Iteration n: [2,3,5,7]
     // in the end, only the primes remain
     ```

     Implement a function that returns all primes up to a number n.
   - (d) Implement a function that tests the primality of a given number n.

(e) Implement a function to factorize a number into prime factors, based on the previous functions. Example: $15 = 3 \times 5$.

4. Matrices are an invaluable tool in quantum information. Here we will try to implement some of the primitive operations involving matrices. Consider that the definition of any matrix in Haskell can be a list of type [[Int]]. For instance, the representation of the Pauli X matrix reads as follows:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \to \texttt{[[0,1],[1,0]]}$$

(a) Calculate the transpose of a matrix A^T

(b) Multiply matrices $A \cdot B$

(c) Calculate the tensor product $A \otimes B$

(d) Calculate projection matrices $A \otimes A^T$

5. The representation of an n-qubit quantum state can be a column vector with 2^n entries. Implement a function that takes a vector representing a 2-qubit state and applies a CNOT operation to it. Note that a CNOT representation can be a 4×4 matrix:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

You may find the following book helpful:
- Haskell Programming From First Principles, by Christopher Allen, and Julie Moronuki