Lab Exercise 4

a) <u>Natural Numbers</u>

Consider the set of natural numbers ℕ, and observe that:

- zero is a natural number, and
- any other natural number is the successor of some other natural number.

1) Define a data type Nat representing natural numbers using the above observation.

2) Write functions toInt :: Nat -> Int and fromInt :: Int -> Nat that allows you to convert between Nat and Int.

b) <u>Complex Numbers</u>

i) Write a data type Complex to represent complex numbers of the form *a* + *b* * *i*.
ii) Make Complex an instance of Show, Eq, and Num

c) <u>Proposition</u>

Consider the following datatype for representing boolean expression (propositions), where variable names consist of a single character:

data Prop = Basic Bool | Var Char | Not Prop | Prop : /\ : Prop | Prop : \/ : Prop | Prop : −>: Prop

1) Give the value of type Prop that represents the proposition p → (p ∨ q) .

2) Write a function vars :: Prop → [Char] that takes a proposition and finds all the variables that occur in the proposition (you may choose to delete duplicates, or not).

3) Write an evaluator beval :: Prop → [(Char , Bool)] → Bool. It takes a proposition and an environment env. The latter contains pairs of values, like (’p’, True), that give values to variables. Compute the truth value of the proposition using the truth value given for every variable in that proposition.

d) <u>Queue</u>

Implement a (polymorphic) queue data structure, make instance of show, and define functions makeQueue, isEmpty, enqueue, dequeue.