CS 511 – Quiz 5: Sequential Erlang

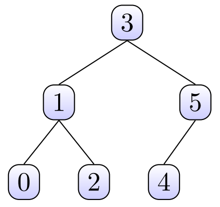
28 October 2020

Names:
Pledge:

Exercise 1

Implement a simple function `isComplete` that determines whether a binary tree is complete.

Recall that a *perfect binary tree* is a binary tree where all nodes have either 2 children or 0 children and all leaves have the same depth. A *complete binary tree* of height n is a perfect binary tree through level $n-1$ with some extra leaf nodes at level n (the tree height), all toward the left. Here is an example:



Binary trees are represented as follows:

```
-type btree() :: {empty}
              |  {node,number(),btree(),btree()}.
```

It may be useful to use a queue. The queue operations are:

- `new() -> queue()`. Returns an empty queue.
- `is_empty(Q :: queue()) -> boolean()`. Tests if Q is empty and returns true if so, otherwise false.
- `in(Item, Q1 :: queue(Item)) -> Q2 :: queue(Item)`. Inserts `Item` at the rear of queue Q1. Returns the resulting queue Q2.
- `out(Q1 :: queue(Item)) -> {{value, Item}, Q2 :: queue(Item)} | {empty, Q1 :: queue(Item)}`. Removes the item at the front of queue Q1. Returns tuple value, Item, Q2, where Item is the item removed and Q2 is the resulting queue. If Q1 is empty, tuple empty, Q1 is returned.

For example,

```
1> Q0 = queue:new().
{[],[]}
2> queue:out(Q0).
{empty,{[],[]}}
3> Q1 = queue:in(2,queue:in(1,Q0)).
{[2],[1]}
4> queue:out(Q1).
{{value,1},{[],[2]}}
```