Lab: Writing and Proving Theorems

Exercise 1:
A classic benchmark problem in logic is the Pigeon Hole Problem. Given `n+1` pigeons and `n` pigeon holes, each with capacity one pigeon, prove that it is impossible to distribute all pigeons among the holes. ■

Exercise 2:
Here are two functions that take a sequence of numbers as input and returns the maximum number in the input sequence.

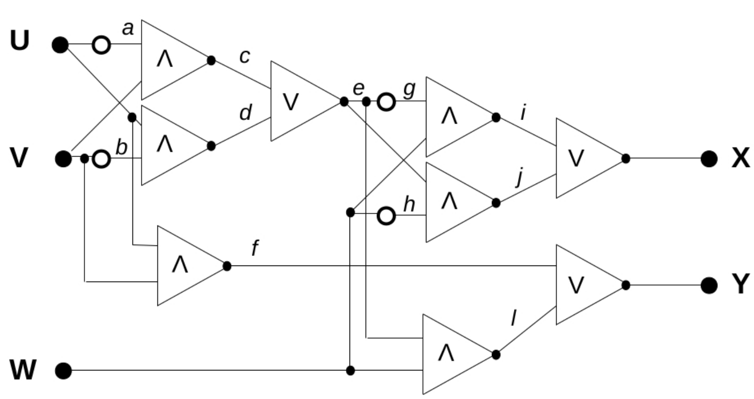
```
fm1 : {a} (fin a, a >= 0) => [a][32] -> [32]
fm1 lst = z ! 0
  where
    z = [0]#[ if x < p then p else x | x <- z | p <- lst ]
fm2 : {a} (fin a, a >= 1) => [a][32] -> [32]
fm2 lst = z ! 0
  where
    z = [lst@0] # [ max a b | a <- (tail lst) | b <- z ]
```

Here is function `member` which returns `True` if and only if `x` is a member of the sequence `lst`:

```
member : {a} (fin a, a >=1) => [32] -> [a][32] -> Bit
member q lst = z ! 0
  where
    z = [False]#[ p == q \/ x  | x <- z | p <- lst ]
```

Prove that `fm1` returns a number that is in the input sequence `lst`. Prove that `fm2` returns a number that is in the input sequence `lst`. Prove that fm1 and fm2 give identical results for sequences of length 100. ■

Exercise 3:
Here is a circuit for a 1 bit adder:



Pin **W** is for the carry-in, pins **U** and **V** are the bits to be added, pin **X** is the sum of **U** and **V** mod 2, pin **Y** is the carry-out of the addition. The values of **X** and **Y** given inputs **U**,**V**, **W** are:

U	**V**	**W**	**X**	**Y**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Write a Cryptol function, call it `adder_netlist`, that constructs the circuit and outputs the pair (**X**,**Y**) given inputs **U**,**V**,**W** according to the table above. Write a Cryptol function, call it `adder_spec`, that expresses the logic of the truth table. Prove that `adder_netlist` is equivalent to `adder_spec`. ■

Exercise 4:
Using the results of Exercise 3 write a function circuit that expresses the output of a 4-bit adder circuit. Write a function spec that expresses the sum of 4 bit numbers. Prove the two are equivalent. ■