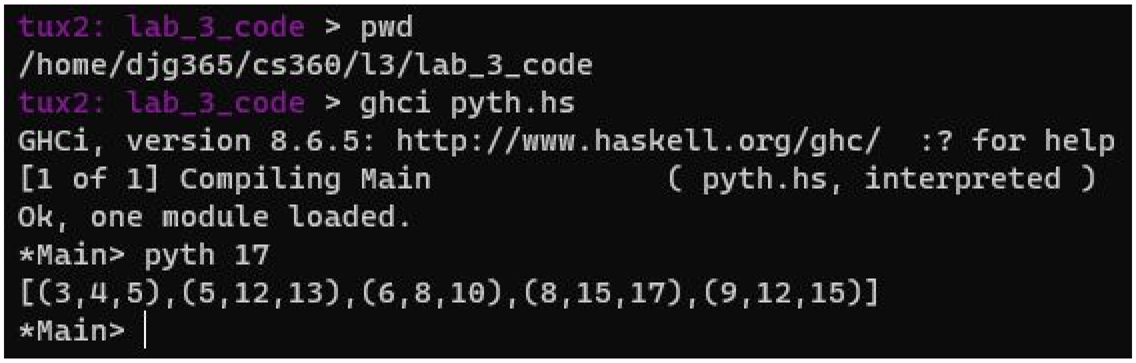
Dennis George, Stephen Hansen, Shivanshi Nagar
Group 4
CS360 Lab 2 Writeup
August 10, 2020

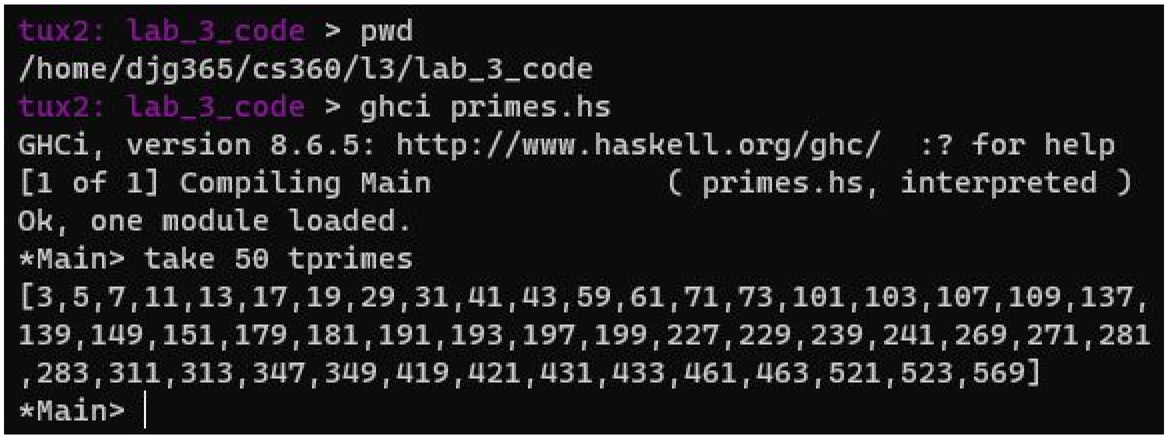
1. (i) Load the file pyth.hs. Use the code of this file to find the first 5 Pythagorean triples.



```
tux2: lab_3_code > pwd
/home/djg365/cs360/l3/lab_3_code
tux2: lab_3_code > ghci pyth.hs
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( pyth.hs, interpreted )
Ok, one module loaded.
*Main> pyth 17
[(3,4,5),(5,12,13),(6,8,10),(8,15,17),(9,12,15)]
*Main> 
```

Calling “pyth 17” will return the first 5 Pythagorean triples as demonstrated in the screenshot above.

(ii) Load the file primes.hs. Use the code of this file to find the first 50 twin primes.



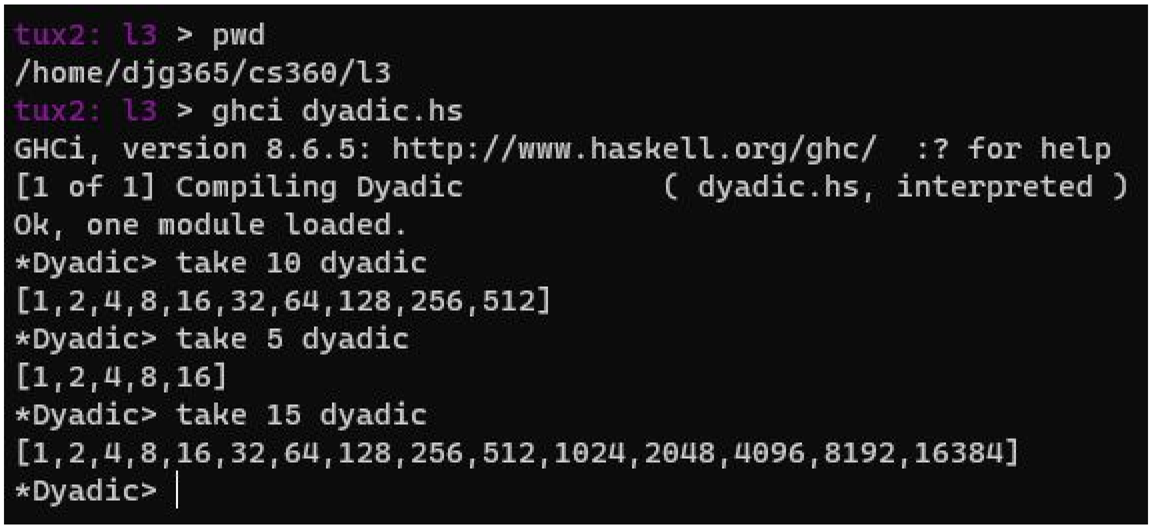
```
tux2: lab_3_code > pwd
/home/djg365/cs360/l3/lab_3_code
tux2: lab_3_code > ghci primes.hs
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( primes.hs, interpreted )
Ok, one module loaded.
*Main> take 50 tprimes
[3,5,7,11,13,17,19,29,31,41,43,59,61,71,73,101,103,107,109,137,
139,149,151,179,181,191,193,197,199,227,229,239,241,269,271,281
,283,311,313,347,349,419,421,431,433,461,463,521,523,569]
*Main> 
```

Calling “take 50 tprimes” will return the first 50 twin primes as demonstrated in the screenshot above.

(iii) It is still an open mathematical problem whether there are infinitely many twin primes. Suppose that there are only finitely many twin primes. Suppose also that you have unlimited time and space resources available. Would it be possible to verify that the Haskell list of twin primes you constructed contains only finitely many items? Justify your answer.

In a finite amount of time, we can access a finite portion of an infinite stream. We cannot access an infinite portion as we don’t have infinite time.
If the stream is infinite we would have to check an infinite number of elements which is impossible in a finite amount of time. We will only be able to check a finite number of elements. This case does not verify the statement.
If the stream is finite, we can access a finite portion of the stream but this does not mean that this finite portion verifies the statement. We would have to keep checking additional elements as we don’t know the extent of the stream, and that is impossible. This case fails as well.
Therefore, even if we have unlimited resources and unlimited time, we won’t be able to verify if the list of twin primes is finite as we would have to check each and every element and that would take infinite time which we do not have.

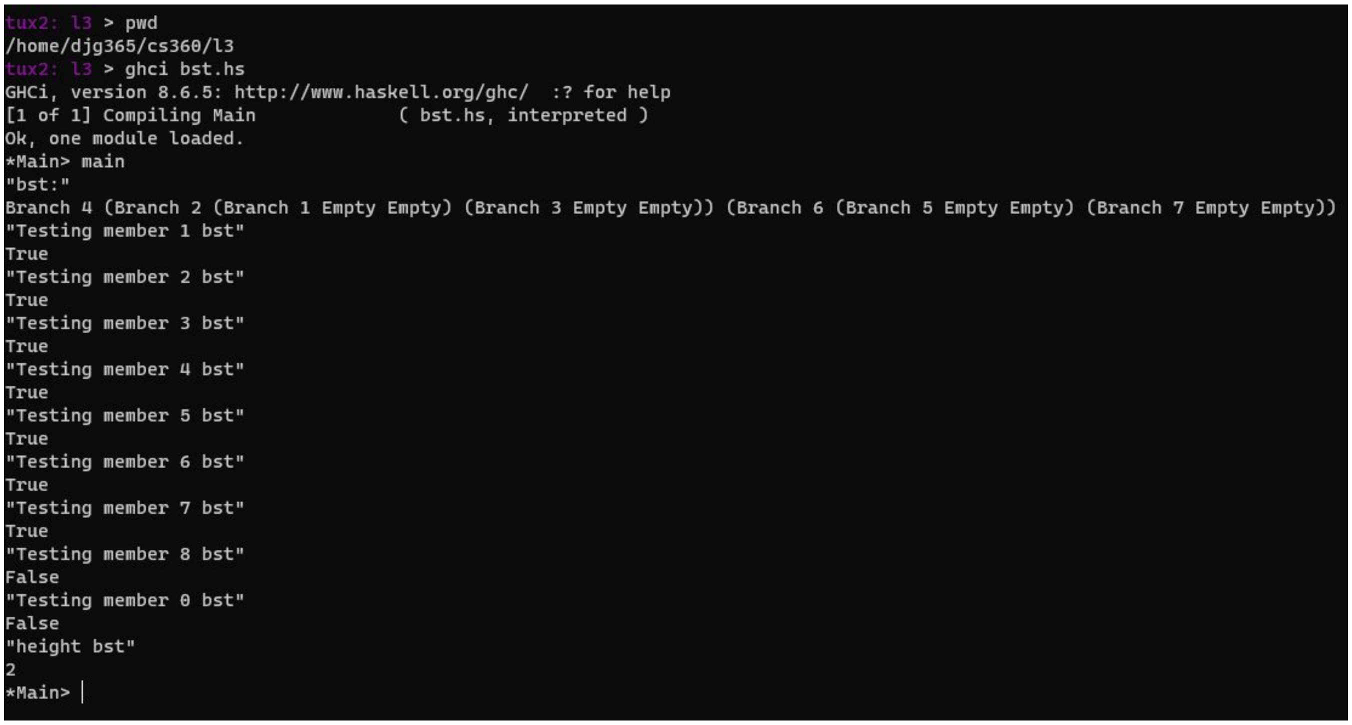
2. Use Haskell list comprehension to construct a list consisting of all dyadic numbers, i.e. the numbers of the form 2^k , where k = 0,1,2,3,… . Demonstrate with operation take that your list is properly constructed.



```
tux2: l3 > pwd
/home/djg365/cs360/l3
tux2: l3 > ghci dyadic.hs
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Dyadic           ( dyadic.hs, interpreted )
Ok, one module loaded.
*Dyadic> take 10 dyadic
[1,2,4,8,16,32,64,128,256,512]
*Dyadic> take 5 dyadic
[1,2,4,8,16]
*Dyadic> take 15 dyadic
[1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384]
*Dyadic> 
```

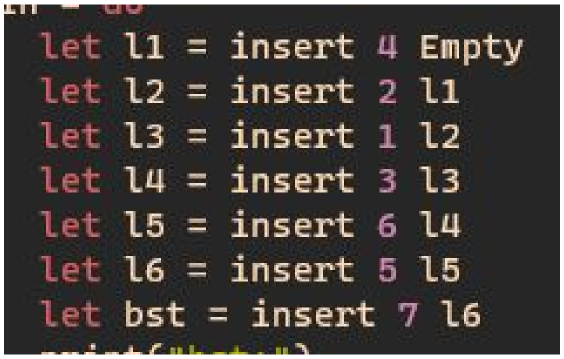
3. Load the BST functions member and insert you prepared for the lab. Use function insert to construct a BST of height 2 consisting of numbers 1-7. Verify that all numbers 1-7 are in the BST using function member. How can you verify that the height of the tree you constructed is 2?

The below screenshot shows the output of the main function running, which creates the BST and tests the member function. I also created a helper function (height) to calculate the height of a BST. This output verifies that the height of the constructed tree is 2.



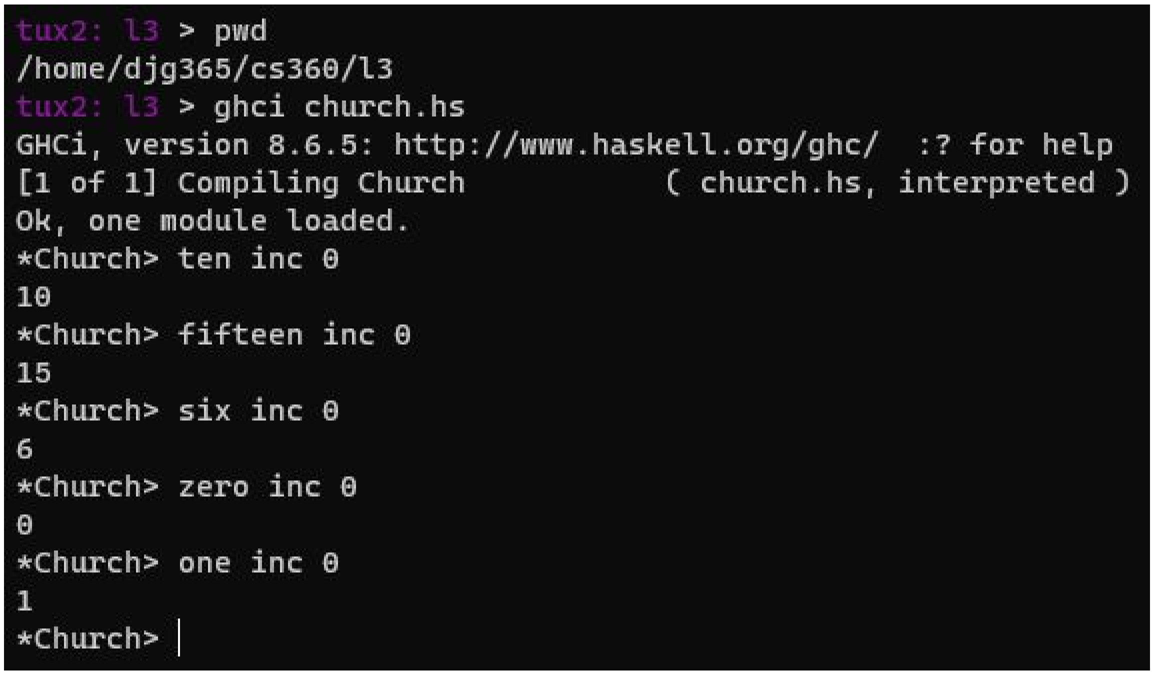
```
tux2: l3 > pwd
/home/djg365/cs360/l3
tux2: l3 > ghci bst.hs
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( bst.hs, interpreted )
Ok, one module loaded.
*Main> main
"bst:"
Branch 4 (Branch 2 (Branch 1 Empty Empty) (Branch 3 Empty Empty)) (Branch 6 (Branch 5 Empty Empty) (Branch 7 Empty Empty))
"Testing member 1 bst"
True
"Testing member 2 bst"
True
"Testing member 3 bst"
True
"Testing member 4 bst"
True
"Testing member 5 bst"
True
"Testing member 6 bst"
True
"Testing member 7 bst"
True
"Testing member 8 bst"
False
"Testing member 0 bst"
False
"height bst"
2
*Main> |
```

The below screenshot is the code that was used to create the BST.



```
  let l1 = insert 4 Empty
  let l2 = insert 2 l1
  let l3 = insert 1 l2
  let l4 = insert 3 l3
  let l5 = insert 6 l4
  let l6 = insert 5 l5
  let bst = insert 7 l6
```

4. Implement addition and multiplication of Church numerals in Haskell.



```
tux2: l3 > pwd
/home/djg365/cs360/l3
tux2: l3 > ghci church.hs
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Church           ( church.hs, interpreted )
Ok, one module loaded.
*Church> ten inc 0
10
*Church> fifteen inc 0
15
*Church> six inc 0
6
*Church> zero inc 0
0
*Church> one inc 0
1
*Church> 
```