Lab 12: Prime Number Design

Comments:

- How long did it take for you to complete this assignment?
    - 12 hours
- What was the hardest part of the assignment?
    - Figuring out how to write for loops in pseudocode. It was difficult to understand
      if I should use a zero-based system or a 1 based system. I had people encouraging both.
      I received help from Brother Gary Godderidge. He helped me format the design how he thought
      it would best be. The hardest part was making sure the design was correct.
      The trace took a long time as well but helped me piece through my logic and verify to me that it was working
- Was there anything unclear about the instructions or how you were to complete this lab?
    - Yes, I was unsure what to do at first and if I could use the square root method or not.
      This was me miss interpreting it the assignment document was very clear. Other than that noting was unclear.

Pseudocode:

```
"""
This code takes a user input integer and creates an array.
The program takes the user number n_range and creates an
array of True values with the number of values equalling
n_range the user input number. It sets 0 and 1 to False
because they are not prime numbers.
The program then uses 2 for loops to turn all of the
non-prime numbers to False. It then creates an array to
store the prime values and appends the prime values to the
array if the value in the is_prime_array is True.
The prime list of values is displayed to the user.
"""
```

```
1. n_range = 0  # O(1)
2. WHILE n_range < 2  # O(1)
3.     GET n_range  # O(1)
4.
5. is_prime_array = []
6. # The loop below is 0 to n_range inclusive
7. FOR p_index in is_prime_array from (0 to (n_range)) # O(n)
8.     is_prime_array.append(True) # O(1) + O(n) = O(n)
9.
10.  is_prime_array[0] = False  # O(1)
11.  is_prime_array[1] = False  # O(1)
12.
13.  square_n = Floor(sqrt(n_range))  # log(n) or sqrt(n)
14.
15.  FOR current_index from 2 to square_n  # O(log(n))
16.      ASSERT 2 <= current_index <= square_n  # O(1) + O(log(n)) = O(log(n))
17.      IF is_prime_array[current_index] == True  # O(1) + O(log(n)) = O(log(n))
18.          # The loop below is (current_index * 2) to n_range inclusive
19.          FOR i_multiple from ((current_index * 2) to n_range, step = current_index)  # O(n log(n))
20.              ASSERT current_index * 2 <= i_multiple <= n_range  # O(1) + O(n log(n)) = O(n log(n))
21.              is_prime_array[i_multiple] = False  # O(1) + O(n log(n)) = O(n log(n))
22.
23.  prime_nums = []  # O(1)
24.  # The loop below is 0 to n_range inclusive
25.  FOR num_index from (0 to (n_range))  # O(n)
26.      IF is_prime_array[num_index] == True  # O(1) + O(n) = O(n)
27.          prime_nums.append(num_index)  # O(1)
28.
29.  ASSERT length(prime_nums) > 0  # O(1)
```

```
30.  ASSERT prime_nums[0] == 2  # O(1)
31.  ASSERT prime_nums[length(prime_nums) - 1] <= n_range  # O(1)
32.
33.  # The loop below is 0 to length(prime_nums) exclusive
34.  FOR index from 1 to length(prime_nums)  # O(n)
35.      ASSERT prime_nums[index - 1] < prime_nums[index]  # O(1) + O(n) = O(n)
36.
37.  PUT prime_nums  # O(1)
```

Algorithmic Efficiency

The algorithmic efficiency of this program is O(n log(n))
You can see this as I put the efficiency of each line of code in the program.
The efficiency is the way it is because the first For loop is O(log(n))
so when the two for loops combine the result is O(n log(n))

Maintainability:

Understandability: I would say this falls under Obvious - The variables are semantic and easy to read, I have included a documentation string and some inline comments. I believe the code is very obvious and easy to understand
Malleability: I would say this falls under Refactorable It would require many structure changes to allow for different functionality. For example, if I wanted this code to be resued and I would need to put this in functions and this would greatly change the structure of the code.

Trace:

Line #	n_range	is_prime_array	is_prime_array_indexes	floor(square_n)	EX_for_loop	IN_for_loop	current_index	i_multiple	Loop_num	prime_nums	num_index
1	0	/	/	/	/	/	/	/	/	/	/
2	0	/	/	/	/	/	/	/	/	/	/
3	10	/	/	/	/	/	/	/	/	/	/
5	10	[]	[]	/	/	/	/	/	/	/	/
6	10	[]	[]	/	/	/	/	/	/	/	/
7	10	[]	[]	/	/	/	/	/	/	/	/
8	10	[T]	[0]	/	/	/	/	/	/	/	/
									/		
8	10	[T,T,T,T,T,T,T,T,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	/	/	/	/	/	/	/	/
10	10	[F,T,T,T,T,T,T,T,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	/	/	/	/	/	/	/	/
11	10	[F,F,T,T,T,T,T,T,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	/	/	/	/	/	/	/	/
13	10	[F,F,T,T,T,T,T,T,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	/	/	/	/	/
15	10	[F,F,T,T,T,T,T,T,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	3	1	/	2	/	/	/	/
17	10	[F,F,T,T,T,T,T,T,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	3	1	/	2	/	/	/	/
19	10	[F,F,T,T,T,T,T,T,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	3	1	1	2	4	/	/	/
21	10	[F,F,T,T,F,T,T,T,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	3	1	1	2	4	/	/	/

19	10	[F,F,T,T,F,T,T,T,T,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	3	1	2	2	6	/	/	/
21	10	[F,F,T,T,F,T,F,T,T,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	3	1	2	2	6	/	/	/
19	10	[F,F,T,T,F,T,F,T,T,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	3	1	3	2	8	/	/	/
21	10	[F,F,T,T,F,T,F,T,F,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	3	1	3	2	8	/	/	/
19	10	[F,F,T,T,F,T,F,T,F,T,T]	[0,1,2,3,4,5,6,7,8,9,10]	3	1	4	2	10	/	/	/
21	10	[F,F,T,T,F,T,F,T,F,T,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	1	4	2	10	/	/	/
15	10	[F,F,T,T,F,T,F,T,F,T,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	2	/	3	/	/	/	/
17	10	[F,F,T,T,F,T,F,T,F,T,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	2	/	3	/	/	/	/
19	10	[F,F,T,T,F,T,F,T,F,T,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	2	1	3	6	/	/	/
21	10	[F,F,T,T,F,T,F,T,F,T,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	2	1	3	6	/	/	/
19	10	[F,F,T,T,F,T,F,T,F,T,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	2	2	3	9	/	/	/
21	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	2	2	3	9	/	/	/
23	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	/	[]	/
25	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	1	[]	0

26	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	1	[]	0
27	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	1	[]	0
25	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	2	[]	1
26	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	2	[]	1
27	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	2	[]	1
25	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	3	[]	2
26	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	3	[]	2
27	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	3	[2]	2
25	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	4	[]	3
26	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	4	[]	3
27	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	4	[2,3]	3
25	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	5	[2,3]	4
26	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	5	[2,3]	4
27	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	5	[2,3]	4

25	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	6	[2,3]	5
26	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	6	[2,3]	5
27	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	6	[2,3,5]	5
25	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	7	[2,3,5]	6
26	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	7	[2,3,5]	6
27	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	7	[2,3,5]	6
25	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	8	[2,3,5]	7
26	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	8	[2,3,5]	7
27	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	8	[2,3,5,7]	7
25	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	9	[2,3,5,7]	8
26	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	9	[2,3,5,7]	8
27	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	9	[2,3,5,7]	8
25	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	10	[2,3,5,7]	9
26	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	10	[2,3,5,7]	9

27	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	10	[2,3,5,7]	9
25	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	11	[2,3,5,7]	10
26	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	11	[2,3,5,7]	10
27	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	11	[2,3,5,7]	10
37	10	[F,F,T,T,F,T,F,T,F,F,F]	[0,1,2,3,4,5,6,7,8,9,10]	3	/	/	3	9	11	[2,3,5,7]	10