EECS 1012: LAB 06 Tracing Algorithms; using Sub-Algorithms (June 16-24, 2021)

A. REMINDERS

1) Lab6 is due on **Thursday (Mar 3)** at 11pm. No late submission will be accepted.
2) The pre-lab mini quiz is considered part of this lab. You are required to complete the pre-lab mini quiz 6 posted on eClass no later than the beginning of your lab time, i.e., 10am on Monday (Feb 28th) if you are in lab01 and 02, or 9:30am on Tuesday (Mar 1st) if you are in lab 03 and 04
3) You are welcome to attend the lab session, if you stuck on any of the steps below. TAs and instructor will be available to help you. Note that from now on, t**he labs will be in-person now, but still optional. The location is WSC (Willam Small Center) 105 and 106**
4) Feel free to signal a TA for help if you stuck on any of the steps below. Yet, note that TAs would need to help other students too.
5) You can submit your lab work any time before the specified deadline.

B. IMPORTANT PRE-LAB WORKS YOU NEED TO DO BEFORE GOING TO THE LAB

1) Download this lab files and read them carefully to the end.
2) Practice tracing algorithms with different sample inputs. There are several examples in Lect06-ct3.

C. GOALS/OUTCOMES FOR LAB

- To practice computational thinking by first drawing flowcharts for basic computation problems, verify if they are correct (by tracing them), then implement them in JS.

D. TASKS

1) Your first task in this lab is to verify some algorithms, by tracing them for some sample inputs. You also provide pre- and post- conditions for each algorithm. Do NOT open VS-Code or browsers before finishing Task 1.

1) In Part 2, you implement all 6 exercises in JavaScript. You should create one html, one css, and one js file for all your work. See next pages for further instructions.

E. SUBMISSIONS

eClass submission. More information can be found at the end of this document.

F. COMPUTATIONAL THINKIG

Part 1: For this part you are encouraged to discuss with other people (peers, TAs, instructor), even to share your <u>partial</u> solutions. But you should not copy other people's solution.

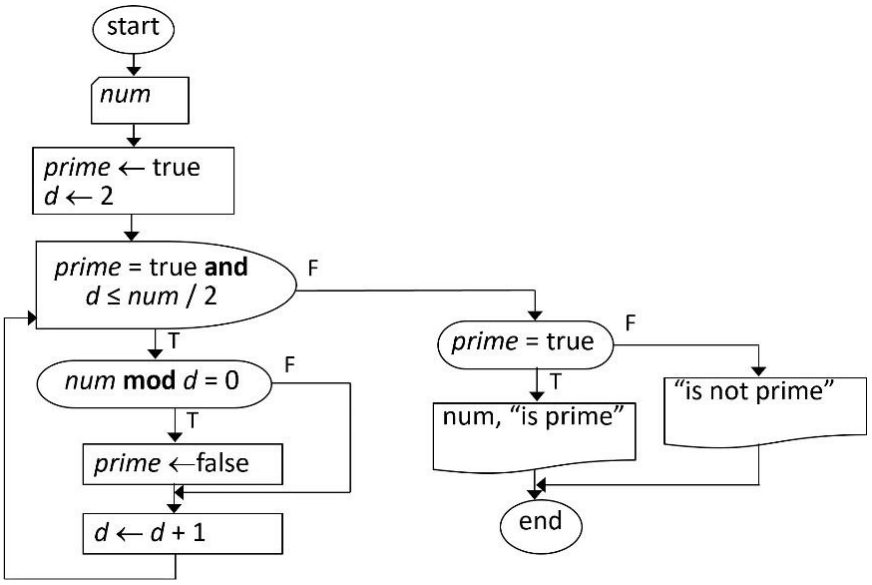
In Exercises 1 to 5, you verify the correctness of all algorithms by tracing them with some sample inputs. You also write pre- and post- conditions for each algorithm. In Exercise 6 and 7, you devise an algorithm that calls two sub-algorithms. In that Exercise you trace your new sub-algorithms for some sample inputs. Note that terms algorithm, sub-algorithm, function, method, programs are used interchangeably, yet they have subtle differences.

For each exercise, take one picture of all your trace tables and pre- post- conditions, including your name. By end of this lab, you should submit these pictures to eClass as **img_**{01,02,03,04,05,06, 07}**.jpg** files, where **img_x** is the picture for exercise **x** below. Make sure the size of each image is less than 500KB, e.g. by reducing the resolution of your camera.

IMPORTANT: You are required to provide preconditions and postconditions for each solution you provide.

Ex 1) Trace the following flowchart for when input is 25 and complete the trace table. Then, trace with 13. Complete the 2nd table. Also, write pre- post- conditions for this flowchart. Assume input is a

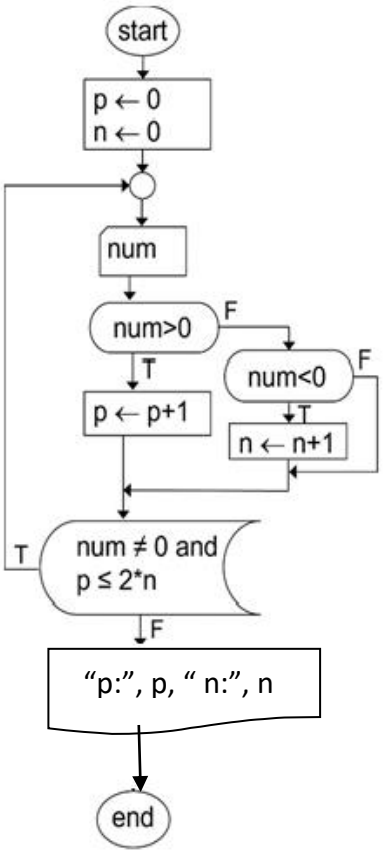
positive integer.



num	prime	d	output

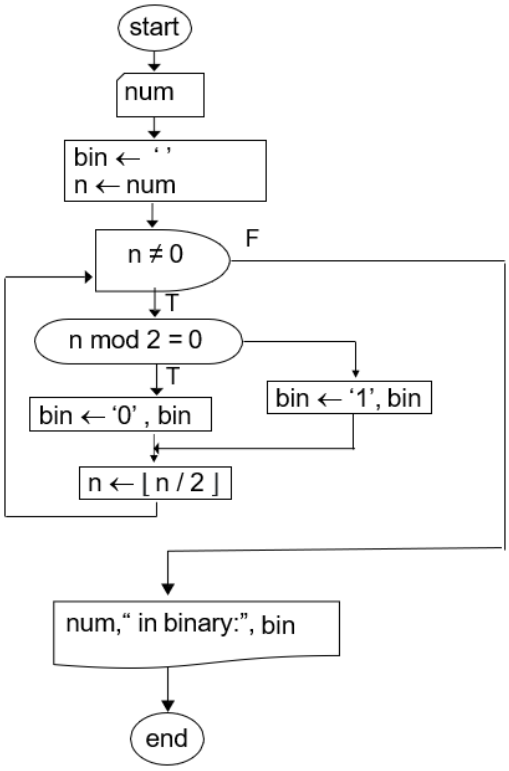
num	prime	d	output

Ex 2) Trace the following flowchart for when input values are -12, 3, -3, 6, -11, 14, 1,10, 3, 2, 6,0 and complete the trace table. Also, write pre- post-conditions for this flowchart. Not that the algorithm stop early so may not take all the inputs. You don't have to trace the inputs that are not taken.



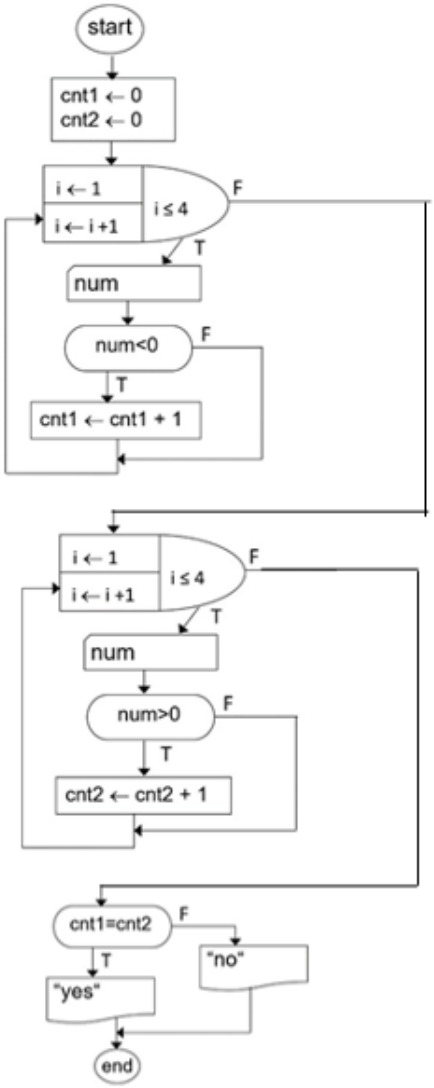
p	n	num	output

Ex 3) Trace the following flowchart for when input value is 9 and complete the trace table. Also, write pre-post-conditions for this flowchart. The algorithm receives a positive whole number and output its binary representation. For more information about binary representation and conversion see the provided pdf file "Number Systems" or search the web. To convert from decimal number to its binary (base 2) representation, we keep on dividing the number by 2, and list the reminders (which is either 0 or 1) from right to left. The process repeats until the number is reduced to 0.



num	bin	n	output

Ex 4) Trace the following flowchart for when input values are 12, 3, -3, 6, 11, 3, -1, -2 and complete the trace table. Also, write pre- post-conditions for this flowchart.

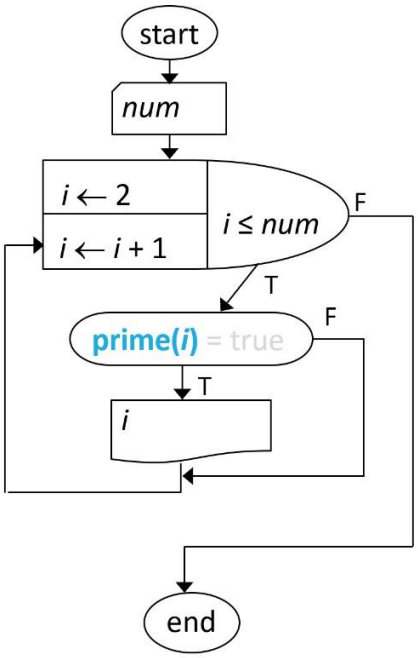


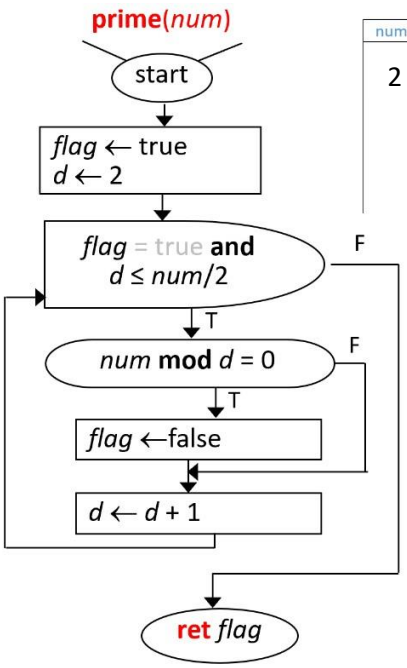
Ex 5) Devise an algorithm that scans a given array a of *n* elements, and get the sum of even values stored in the even position of the array. Assume the array index starts from 0 (so even position are index 0,2,4,6.. Trace your algorithm by assume the array is 3,1,6,4,8,12. In this case the sum is 6+8=14. You can see examples in slides to get started from there. The trace table here assume you use *sum* and *i* as variable. If you use other variables, change the trace table accordingly. Your image for this question should contain both the flowchart, trace table and pre-post conditions.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	sum	i	output
3	1	6	4	8	12			

Ex 6) This exercise uses sub-algorithms. Trace it for when input value is 6 and complete the trace tables. Also note that there are two variables name num, one in the main algorithm and another in the sub-algorithm. Assume input is a positive integer. Write pre- post-conditions for each of these algorithms (the main one and the sub-algorithm). Note that the prime sub-algorithm is similar to the algorithm in Ex 1) but instead of reading input and producing outputs, it has a parameter *num*, and it returns the result. Also note that the =true is in gray color, which means it can be ignored. Note that since the sub-program is called several times, it's better that you use separate trace tables, one for each call, as shown below.

num	i	output
6		





num	flag	d	returned value
2			

num	flag	d	returned value
3			

num	flag	d	returned value
4			

.....

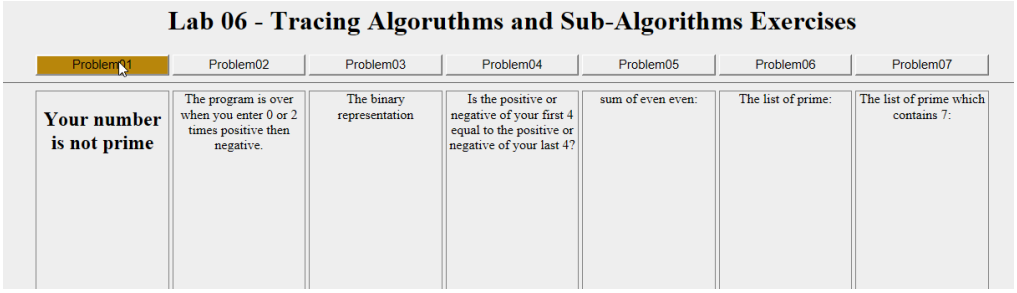
Ex 7) Devise an algorithm to scan an input array a of *n* elements, and output all prime numbers in the array that contain digit 7. For example, given array 27, 7, 13,17, 31, 37, where prime number are 7, 13, 17,31,37, the program should output 7, 17, 37.

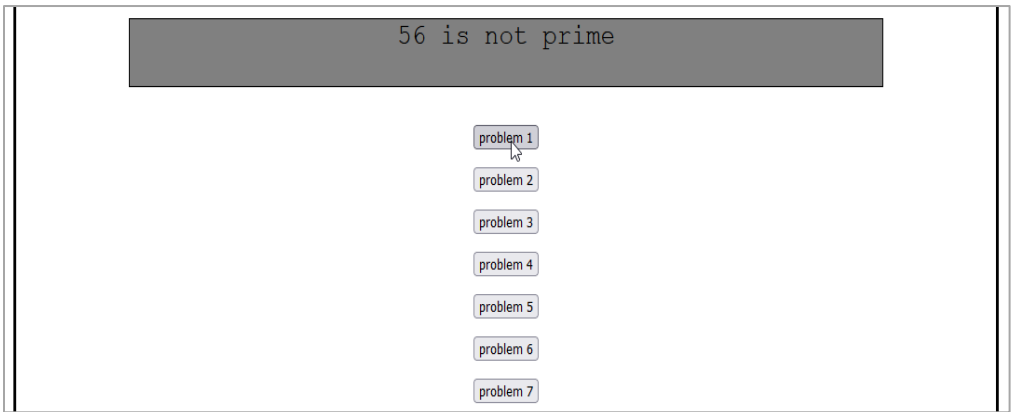
Your solution should have a main algorithm and two sub-algorithms, let's call them *prime(num)* and *has7(num)*. You may reuse the prime sub-algorithm of Ex 6. You do not need to verify (e.g. by tracing) it though because you did it in Ex 5. You need to trace your main algorithm and the has7() algorithm. You should provide pre- and post-conditions for all your 3 (sub)algorithms. Note that since the sub-program has7 is called several times, it's better that you use separate trace tables, one for each call.

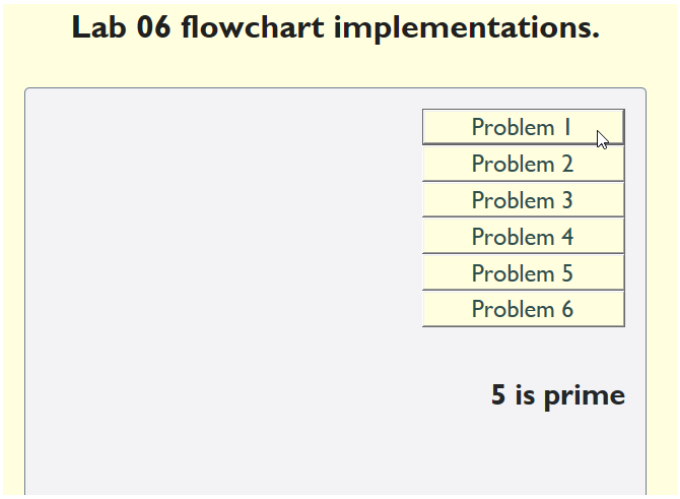
Hint, in has7(), you may want to use a variable *flag* and a while loop so that you can terminate the loop early as long as a 7 is detected. Specifically, *flag* is initially false, and one of the while condition is that *flag* is false. When a 7 is detected, set *flag* to be true. This will terminate the loop. Finally return *flag*.

Do part2 after you finish part1.

Part 2: Create a **lab06<yourname>.html** file supported by a **lab06<yourname>.css** and a **lab06<yourname>.js** files. In your html file, you should have 7 buttons with captions Problem 1, Problem 2, …, Problem 7. Then, you implement all 7 flowcharts in your **lab06<yourname>.js** . When each button is clicked, the event handler should execute the corresponding algorithm that you have in your **js** file. For Problem 5, define a 6 element array var a = [3,1,6,4,8, 12]; For problem 7, define a 6 element array var a = [6,7,13,17,31, 37]; Your function takes inputs from the *prompt* window, and write output on a textbox in the html. Don't display output using *alert* or *confirm* window. Feel free to design the interface you have. Form example, you can adapt the learning kit program, or the light on/off program etc. A few examples are given here for your reference.







G. AFTER-LAB WORKS (THIS PART WILL NOT BE GRADED)

In order to review what you have learned in this lab as well as expanding your skills further, we recommend the following questions and extra practices:

1) Revisit other flowcharts you have drawn/seen in the course and trace them for some sample inputs.
2) Devise an algorithm to receive a positive number, n, and output all prime numbers that are smaller than *n* and all their digits are prime too. For example, if n is 100, the program should output 2, 3, 5, 7, 23, 37, 53,

and 73.

3) Add all algorithms and programs that you have drawn throughout the semester in your **Learning Kit** Project. Your Learning Kit project should include at least 30 problems by now.

Please feel free to discuss any of these questions in the course forum or see the TAs and/or Instructors for help.

H. SUBMISSION

eClass submission

You will see an assignment submission link on eClass. Create a **folder** named "**Lab06**" and copy all your lab materials inside (`img_{01,02,03,04,05,06,07}.jpg`, `Lab06<yourName>.html`, `Lab06<yourName>.css` and `Lab06<yourName>.js`). This folder should be compressed (or tar.gz) and the compressed file submitted.