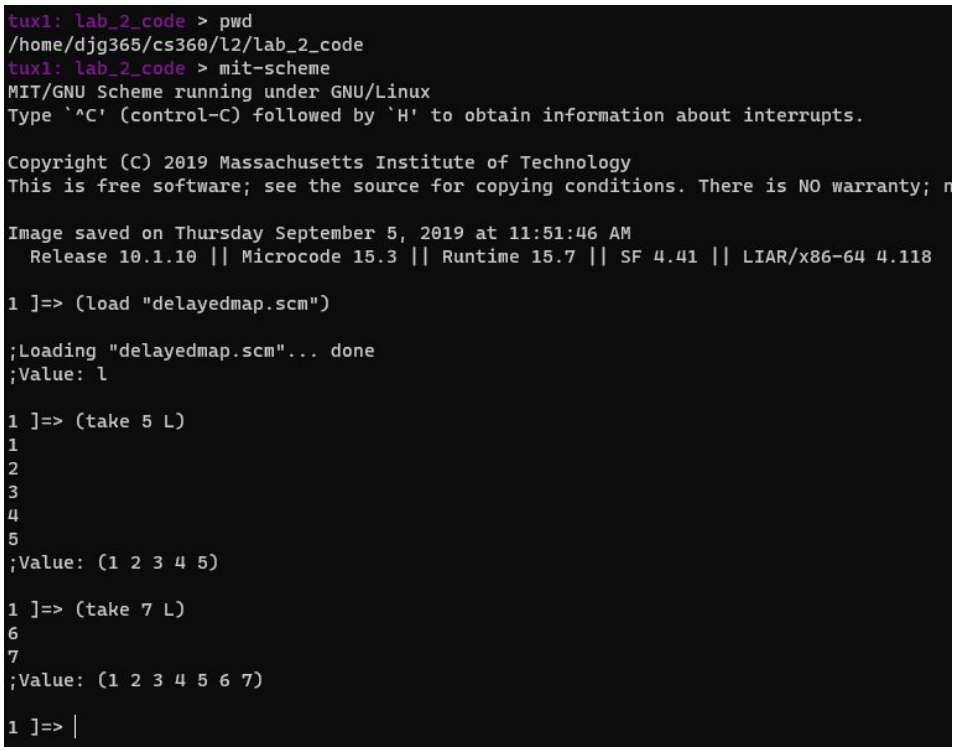
Dennis George, Stephen Hansen, Shivanshi Nagar
CS360 Lab 2 Writeup
July 27, 2020

1. (i)Load the file delayedmap.scm. What will the Scheme interpreter print when asked to evaluate the following expressions in the given order?
   (take 5 L)
   (take 7 L)
   Explain why in the second call to the take function, the numbers 1-5 are not printed out.



   The second call to the "take" function will not print out the numbers 1-5 because L is a delayed list.
   The value of a promise that is returned by a call to force will be cached, so that if it is forced a second time, the previously computed value is returned without any recomputation. This is why the numbers 1 - 5 are not printed out in the second call to (take).

(ii) Load the file BST.scm. It contains functions (member) and (insert). 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?

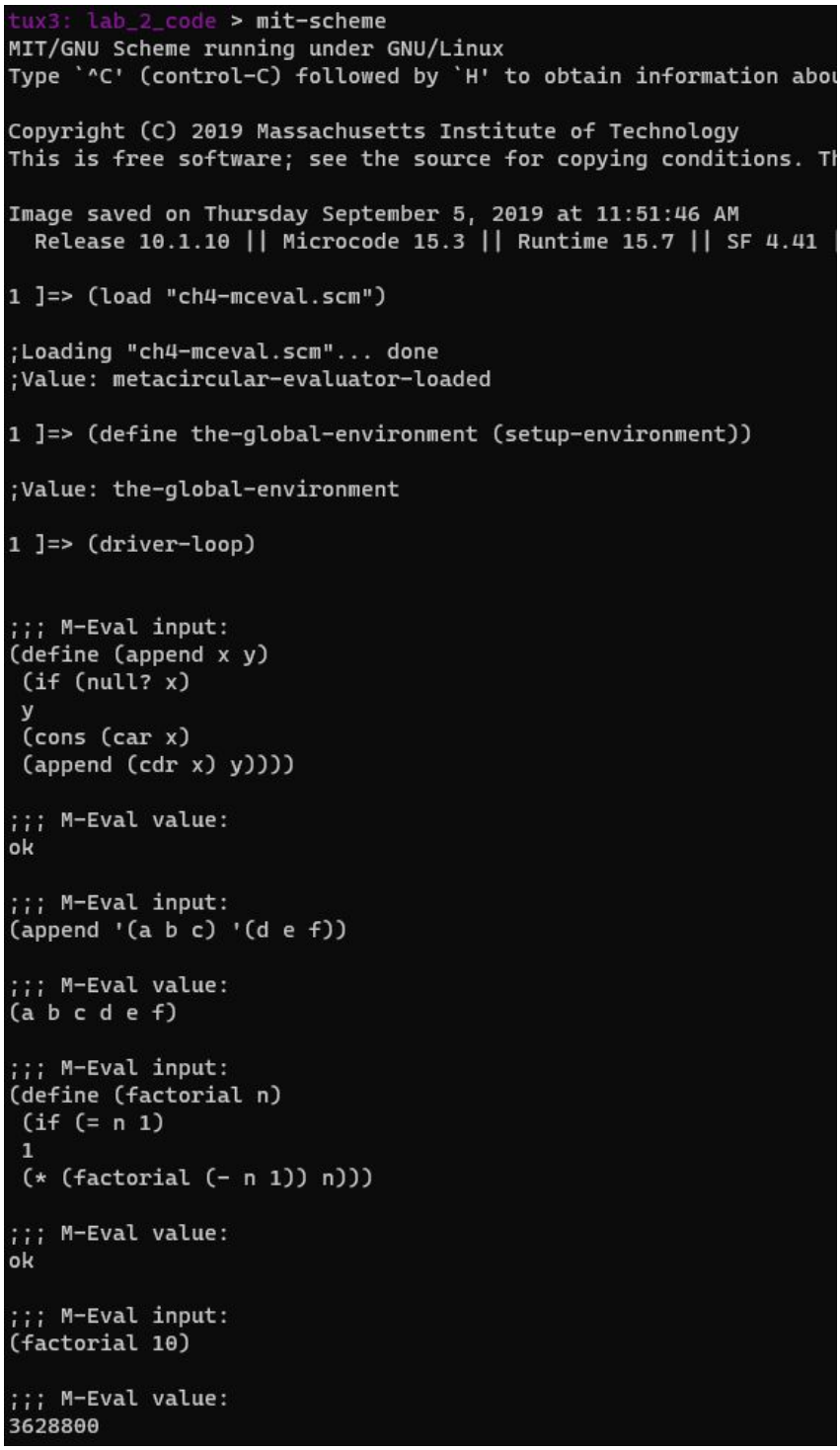
Inside of BST.scm, I added code to create the necessary BST. A screenshot of the code and the results are below.

```
(define BST '())
(define BST (insert 4 BST))
(define BST (insert 2 BST))
(define BST (insert 1 BST))
(define BST (insert 3 BST))
(define BST (insert 6 BST))
(define BST (insert 5 BST))
(define BST (insert 7 BST))
```

```
tux1: lab_2_code > pwd
/home/djg365/cs360/l2/lab_2_code
tux1: lab_2_code > mit-scheme
MIT/GNU Scheme running under GNU/Linux
Type `^C' (control-C) followed by `H' to obtain information abou

Copyright (C) 2019 Massachusetts Institute of Technology
This is free software; see the source for copying conditions. Th

Image saved on Thursday September 5, 2019 at 11:51:46 AM
  Release 10.1.10 || Microcode 15.3 || Runtime 15.7 || SF 4.41

1 ]=> (load "BST.scm")

;Loading "BST.scm"... done
;Value: (4 (2 (1 () ()) (3 () ())) (6 (5 () ()) (7 () ())))

1 ]=> (member 1 BST)

;Value: #t

1 ]=> (member 2 BST)

;Value: #t

1 ]=> (member 3 BST)

;Value: #t

1 ]=> (member 4 BST)

;Value: #t

1 ]=> (member 5 BST)

;Value: #t

1 ]=> (member 6 BST)

;Value: #t

1 ]=> (member 7 BST)

;Value: #t

1 ]=> (member 0 BST)

;Value: #f

1 ]=> (member 8 BST)

;Value: #f
```

I created a helper function bst-height in order to computer the height of a BST. This function is in BST.scm and is named (bst-height).
It recursively takes the max between the left and right child, and adds one each time, and will eventually return the height. In the case of the BST that I defined in BST.scm, the height returned is 2.

2. Load ch4-mceval.scm. Execute the following code in the metacircular evaluator. Code execution is in this screenshot below.



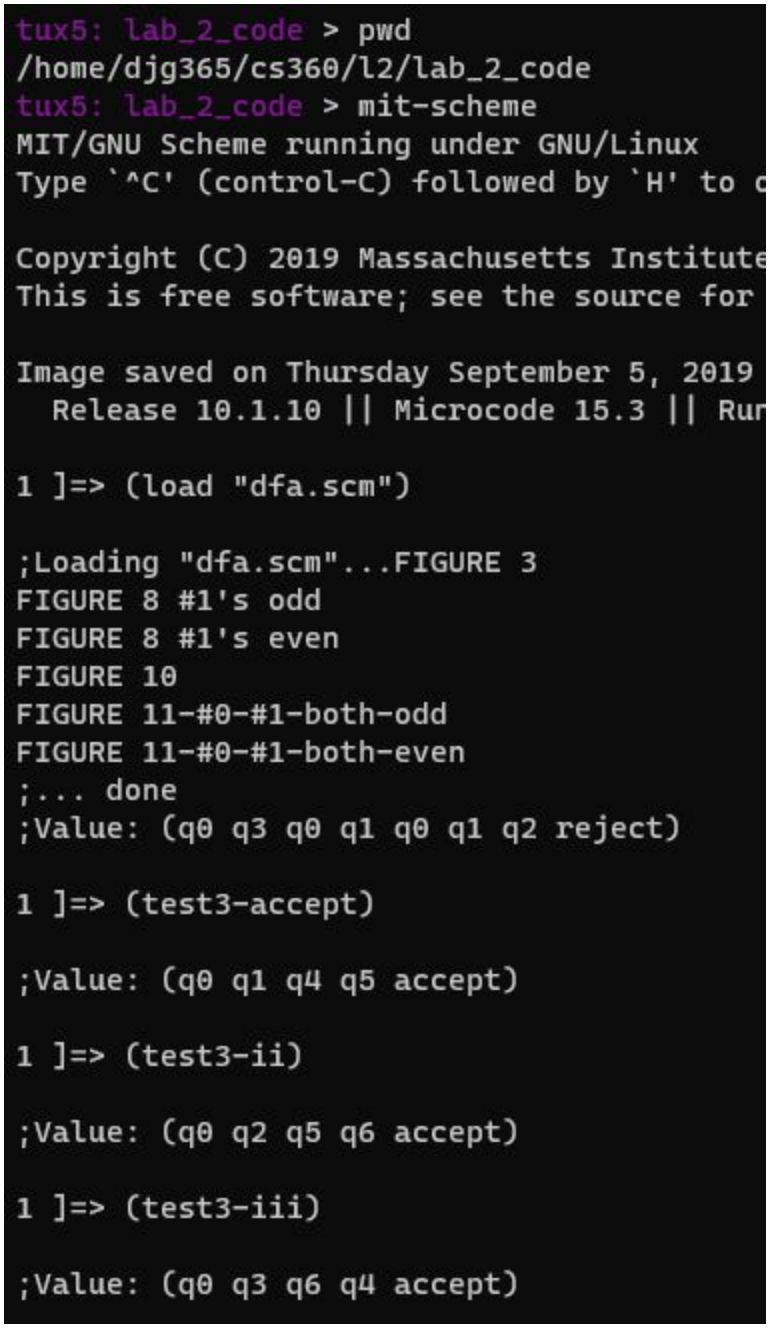
```
tux3: lab_2_code > mit-scheme
MIT/GNU Scheme running under GNU/Linux
Type `^C' (control-C) followed by `H' to obtain information abou

Copyright (C) 2019 Massachusetts Institute of Technology
This is free software; see the source for copying conditions. Th

Image saved on Thursday September 5, 2019 at 11:51:46 AM
  Release 10.1.10 || Microcode 15.3 || Runtime 15.7 || SF 4.41

1 ]=> (load "ch4-mceval.scm")

;Loading "ch4-mceval.scm"... done
;Value: metacircular-evaluator-loaded

1 ]=> (define the-global-environment (setup-environment))

;Value: the-global-environment

1 ]=> (driver-loop)


;;; M-Eval input:
(define (append x y)
 (if (null? x)
 y
 (cons (car x)
 (append (cdr x) y))))

;;; M-Eval value:
ok

;;; M-Eval input:
(append '(a b c) '(d e f))

;;; M-Eval value:
(a b c d e f)

;;; M-Eval input:
(define (factorial n)
 (if (= n 1)
 1
 (* (factorial (- n 1)) n)))

;;; M-Eval value:
ok

;;; M-Eval input:
(factorial 10)

;;; M-Eval value:
3628800
```

Explain why Louis's map fails even though Eva's works.

Louis’s map calls the Scheme system version of map which takes two parameters: a Scheme function (lambda)3 and a list. When calling functions in the metacircular evaluator, functions in the evaluator have a different representation from what Scheme uses (to account for local variable binding, environments, and other things).That is, a function defined in the metacircular evaluator has a different underlying data structure from what Scheme uses to represent functions. When Louis calls map, he is effectively passing a metacircular evaluator function to the system map, which expects a Scheme function. The system version of map has no clue how to evaluate metacircular evaluator functions and fails.
Eva’s map works because it is fully defined in the metacircular evaluator and properly accepts and evaluates a metacircular evaluator function as part of its inputs. The general rule of thumb for when to use and when not to use primitives in the metacircular evaluator is if the input types (the arguments) and the return types of the metacircular evaluator function are the same data types that the Scheme version of the function uses. Simple functions like addition and subtraction can be used as primitives since the representation of numbers is the same for both the metacircular evaluator and Scheme. But complex functions like map cannot be used since the data type representation of functions differs between Scheme and the metacircular evaluator; map must be defined within the context of the metacircular evaluator, as Eva has shown.

3. Load dfa.scm. Simulate the DFAs of Figures 3,8,10,11 of Weeks 1-2 file. Demonstrate on example inputs that the DFA simulation works properly.

   All of the DFAs are simulated in the file dfa.scm. I am attaching a screenshot of the tests that I ran to verify that the DFA simulation works properly.
   test3-accept
   test3-ii
   test3-iii



test8-1-accept
test8-1-reject

```
tux5: lab_2_code > pwd
/home/djg365/cs360/l2/lab_2_code
tux5: lab_2_code > mit-scheme
MIT/GNU Scheme running under GNU/Linux
Type `^C' (control-C) followed by `H' to ob

Copyright (C) 2019 Massachusetts Institute
This is free software; see the source for c

Image saved on Thursday September 5, 2019 a
  Release 10.1.10 || Microcode 15.3 || Runt

1 ]=> (load "dfa.scm")

;Loading "dfa.scm"...FIGURE 3
FIGURE 8 #1's odd
FIGURE 8 #1's even
FIGURE 10
FIGURE 11-#0-#1-both-odd
FIGURE 11-#0-#1-both-even
;... done
;Value: (q0 q3 q0 q1 q0 q1 q2 reject)

1 ]=> (test8-1-accept)

;Value: (q0 q1 q1 q0 q1 q0 q0 q1 accept)

1 ]=> (test8-1-reject)

;Value: (q0 q1 q1 q0 q1 q1 q0 reject)
```

test8-2-accept
test8-2-reject

```
tux5: lab_2_code > pwd
/home/djg365/cs360/l2/lab_2_code
tux5: lab_2_code > mit-scheme
MIT/GNU Scheme running under GNU/Linux
Type `^C' (control-C) followed by `H' to ob

Copyright (C) 2019 Massachusetts Institute
This is free software; see the source for c

Image saved on Thursday September 5, 2019 a
  Release 10.1.10 || Microcode 15.3 || Runt

1 ]=> (load "dfa.scm")

;Loading "dfa.scm"...FIGURE 3
FIGURE 8 #1's odd
FIGURE 8 #1's even
FIGURE 10
FIGURE 11-#0-#1-both-odd
FIGURE 11-#0-#1-both-even
;... done
;Value: (q0 q3 q0 q1 q0 q1 q2 reject)

1 ]=> (test8-2-accept)

;Value: (q0 q1 q1 q0 q1 q1 q0 accept)

1 ]=> (test8-2-reject)

;Value: (q0 q1 q1 q0 q1 q1 q0 q1 reject)
```

test10-accept
test10-reject

```
tux5: lab_2_code > pwd
/home/djg365/cs360/l2/lab_2_code
tux5: lab_2_code > mit-scheme
MIT/GNU Scheme running under GNU/Linux
Type `^C' (control-C) followed by `H' to

Copyright (C) 2019 Massachusetts Institut
This is free software; see the source for

Image saved on Thursday September 5, 2019
  Release 10.1.10 || Microcode 15.3 || Ru

1 ]=> (load "dfa.scm")

;Loading "dfa.scm"...FIGURE 3
FIGURE 8 #1's odd
FIGURE 8 #1's even
FIGURE 10
FIGURE 11-#0-#1-both-odd
FIGURE 11-#0-#1-both-even
;... done
;Value: (q0 q3 q0 q1 q0 q1 q2 reject)

1 ]=> (test10-accept)

;Value: (q0 q1 q0 q1 q0 q1 q0 accept)

1 ]=> (test10-reject)

;Value: (q0 q1 q0 q1 q2 q2 q2 q2 reject)
```

test11-1-accept
test11-1-reject

```
tux5: lab_2_code > pwd
/home/djg365/cs360/l2/lab_2_code
tux5: lab_2_code > mit-scheme
MIT/GNU Scheme running under GNU/Linux
Type `^C' (control-C) followed by `H' to

Copyright (C) 2019 Massachusetts Institut
This is free software; see the source for

Image saved on Thursday September 5, 2019
  Release 10.1.10 || Microcode 15.3 || Ru

1 ]=> (load "dfa.scm")

;Loading "dfa.scm"...FIGURE 3
FIGURE 8 #1's odd
FIGURE 8 #1's even
FIGURE 10
FIGURE 11-#0-#1-both-odd
FIGURE 11-#0-#1-both-even
;... done
;Value: (q0 q3 q0 q1 q0 q1 q2 reject)

1 ]=> (test11-1-accept)

;Value: (q0 q3 q0 q1 q0 q3 q2 accept)

1 ]=> (test11-1-reject)

;Value: (q0 q3 q0 q3 q2 q1 q0 reject)
```

test11-2-accept
test11-2-reject

```
tux5: lab_2_code > pwd
/home/djg365/cs360/l2/lab_2_code
tux5: lab_2_code > mit-scheme
MIT/GNU Scheme running under GNU/Linux
Type `^C' (control-C) followed by `H' to obtain

Copyright (C) 2019 Massachusetts Institute of T
This is free software; see the source for copyi

Image saved on Thursday September 5, 2019 at 11
  Release 10.1.10 || Microcode 15.3 || Runtime

1 ]=> (load "dfa.scm")

;Loading "dfa.scm"...FIGURE 3
FIGURE 8 #1's odd
FIGURE 8 #1's even
FIGURE 10
FIGURE 11-#0-#1-both-odd
FIGURE 11-#0-#1-both-even
;... done
;Value: (q0 q3 q0 q1 q0 q1 q2 reject)

1 ]=> (test11-2-accept)

;Value: (q0 q3 q0 q1 q0 q3 q2 q3 q0 accept)

1 ]=> (test11-2-reject)

;Value: (q0 q3 q0 q1 q0 q1 q2 reject)
```

4. This code is available in the file “fib.scm”. A screenshot of a test is below.