SCHEME 2

COMPUTER SCIENCE 61A

November 1, 2016

1 Scheme Basics

1. What will Scheme output? Draw box-and-pointer diagrams.

   1.
   ```
   (cons (cons 1 nil) (cons 2 (cons (cons 3 (cons 4 5)) (cons 6 nil))))
   ```

   2.
   ```
   (define a 4)
   ((lambda (x y) (+ a)) 1 2)
   ```

   3.
   ```
   ((lambda (x y z) (y x)) 2 / 2)
   ```

   4.
   ```
   ((lambda (x) (x x)) (lambda (y) 4))
   ```

5. `(define boom1 (/ 1 0))`

6. `boom1`

7. `(define boom2 (lambda () (/ 1 0)))`

8. `(boom2)`

9. Why/How are the two boom definitions above different?

10. How can we rewrite boom2 without using the lambda operator?

2 Writing Scheme Procedures

1. Write a procedure `blastoff` that takes in a number `n` and returns a list of all numbers from `n` and 1 followed by `BLASTOFF!`.

   ```
   > (countdown 10)
   (10 9 8 7 6 5 4 3 2 1 BLASTOFF!)

   > (countdown 3)
   (3 2 1 BLASTOFF!)

   (define (countdown n)






   )
   ```

2. Write `before-in-list`, which takes a list, `lst` and two elements `a` and `b`. It should return `#t` if `a` appears in `lst` before `b`. Check the doctests for more details.

   Hint: Recall `contains?` from Homework 9.

   ```
   > (before-in-list '(1 2 3) 1 3)
   #t
   > (before-in-list '(1 2 3) 3 1)
   #f
   > (before-in-list '(1 2 3) 1 4)
   #f
   > (before-in-list '(1 2 3) 0 3)
   #f

   (define (before-in-list lst a b)






   )
   ```

3. Describe the result of calling the following procedure with a list as its argument. What would

```
(mystery '(1 2 3))
```

return?

```
(define (mystery lst)
   (mystery-helper lst '()))

(define (mystery-helper lst other)
   (if (null? lst)
     other
     (mystery-helper (cdr lst) (cons (car lst) other))))
```

4. Write `wheres-waldo`, a Scheme procedure which takes in a scheme list and outputs the index of `waldo` if the symbol `waldo` exists in the list. Otherwise, it outputs the symbol `nowhere`.

```
> (wheres-waldo '(moe larry waldo curly))
2
> (wheres-waldo '(1 2))
nowhere

(define (wheres-waldo lst)
    (cond
        ((null? lst) __________________)
        ((equal? _______________________________________)
        (else
            (let ((found-him _______________________________))
                (if (equal? ___________________________________
                   )
                    _________________
                    (+ 1 ___________________________)
                )
            )
        )
    )
)
```

5. Write a procedure that takes in a number `n` and returns a binary representation of `n`

```
> (to-binary 2)
(0 1 0)

> (to-binary 7)
(0 1 1 1)
```

Note: Here is an approach to finding the binary representation of a number.

1. What is the value of `n` % 2? Take note of this number.
2. Let `n` = `n` // 2
3. Repeat steps 1 and 2 until `n` becomes 0.
4. Reverse the order of the remainders you took note of in step 1.

Example:

$$\begin{aligned} n &= 9 \\ 9\%2 &= 1 \\ 4\%2 &= 0 \\ 2\%2 &= 0 \\ 1\%2 &= 1 \\ 0\%2 &= 0 \end{aligned}$$

So the binary representation of 9 is: 01001

```
(define (to-binary n)












)
```