Problems

Evaluation

Evaluate the following expressions. If they generate an error, give a general description of the error. If they result in a procedure, give the type of the procedure. If they result in a list/pair structure, draw the box-and-pointer for the structure.

(+ 3 4)

(7)

(if (+ 3 4) 8 9)

(define x 17)

(lambda (y) x)

(((lambda (x y) (x y)) (lambda (z) (lambda (a) (+ a z))) 8)

(cons 1 (list 3 4 (cons 5 6)))

(let ((a (list 4 5)))
  (cons a (list 3 a)))
Queues

A data structure that stores elements in order. Elements are enqueued onto the tail of the queue. Elements are dequeued from the head of the queue. Thus, the first element enqueued is also the first element dequeued (FIFO, first-in-first-out). The head operation is used to get the element at the head of the queue.

```
(head (enqueue 5 (empty-queue)))
;Value: 5
```

```
(define q (enqueue 4 (enqueue 5 (enqueue 6 (empty-queue)))))
```

```
(head q)
;Value: 6
```

```
(head (dequeue q))
;Value: 5
```

1. Decide on an implementation for queues, then draw a box-and-pointer representation of the value of q as defined above.

2. Write empty-queue.

```
(define (empty-queue)
```

Order of growth in time? Space?

3. Write enqueue; a procedure that returns a new queue with the element added to the tail.

```
(define (enqueue x q)
```
Can you do this with fold-right?

4. Write dequeue; a procedure that returns a new queue with the head element removed.

\[
\text{(define (dequeue q)}
\]

5. Write head; a procedure that returns the value of the head element.

\[
\text{(define (head q)}
\]
Map, Filter, and Fold-Right

Suppose \( x \) is bound to the list \((1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7)\). Using map, filter, and/or fold-right, write an expression involving \( x \) that returns:

6. \((1 \ 4 \ 9 \ 16 \ 25 \ 36 \ 49)\)

7. \((1 \ 3 \ 5 \ 7)\)

8. \(((1 \ 1) (2 \ 2) (3 \ 3) (4 \ 4) (5 \ 5) (6 \ 6) (7 \ 7))\)

9. \(((2) ((4) ((6) #f)))\)

10. The maximum element of \( x \): 7

11. The last pair of \( x \): (7)