EIGHTH QUIZ

You have 15 minutes from the start of class to complete this quiz. Read the questions with care; work with deliberate speed. Don’t give us more than we ask for. The usual instructions apply. Good luck!

Problem 1 (5 points)

For each of the algorithms or operations described below, check the box corresponding most closely to its complexity (i.e., its O-notation).

(a) \((\text{cons} \ ‘\text{turkey} \ L)\) where \(L\) is a list of \(n\) symbols.

- [ ] Constant—\(O(1)\)
- [ ] Logarithmic—\(O(\log n)\)
- [ ] Linear—\(O(n)\)
- [ ] Quadratic—\(O(n^2)\)

(b) \((\text{foldr} + 0 \ (\text{map} f \ L))\) where \(f\) is a function and \(L\) is a list of \(n\) items.

- [ ] Constant—\(O(1)\)
- [ ] Logarithmic—\(O(\log n)\)
- [ ] Linear—\(O(n)\)
- [ ] Quadratic—\(O(n^2)\)

(c) In a binary search tree of \(n\) restaurants, ordered by name, find a restaurant given its name.

- [ ] Constant—\(O(1)\)
- [ ] Logarithmic—\(O(\log n)\)
- [ ] Linear—\(O(n)\)
- [ ] Quadratic—\(O(n^2)\)

(d) Adding the 45th and the 729th elements of an \(n\)-element vector (where \(n \geq 729\)).

- [ ] Constant—\(O(1)\)
- [ ] Logarithmic—\(O(\log n)\)
- [ ] Linear—\(O(n)\)
- [ ] Quadratic—\(O(n^2)\)

(e) From a binary search tree of \(n\) restaurants, ordered by name, print an alphabetical list of all the restaurants.

- [ ] Constant—\(O(1)\)
- [ ] Logarithmic—\(O(\log n)\)
- [ ] Linear—\(O(n)\)
- [ ] Quadratic—\(O(n^2)\)

Problem 2 (2 points)

List two characteristics of binary circuitry that makes it especially effective for storage in modern computers (as opposed to using a component that directly represents more than two different values).

Problem 3 (3 points)

Choose one or the other:

- List three kinds of computational resources that an operating system helps manage
- List three categories of functionality that a modern operating system performs

Problem 4 (1 point)

How many bits does it take to represent 8 different values?
Problem 5 (11 points)

Suppose you have a vector of rrant structures, which are defined as usual:

```
(define-struct rrant (name cuisine phone dish price))
```

(a) (3 points) Complete the following definition:

```
;; Nth-rrant: vector-of-rrant number -> rrant
;; Return the rrant at the specified position in the vector (zero-based)
(define Nth-rrant
  (lambda (RV N)
    (cond
      ((or (< N 0) (>= N (vector-length RV))) (error N "Vector ref out of range"))
      (else
        (vector-ref RV N))))
```

SCORING: 1 point for each. Only take off 1/2 if they try to add/subtract one from N.

(b) (3 points) Suppose you have the function vector-map, which takes a vector and a function and returns a new vector whose contents are the results of applying the function to each element of the input vector. Thus, `(vector-map (vector 3 4 5) double)` would return `(vector 6 8 10)`. Complete the following definition that changes all the prices in a vector of rrant structures:

```
;; adjust-prices-in-vector: vector-of-rrant number -> vector-of-rrant
;; Adjust each price in the input vector by the specified percentage
(define adjust-prices-in-vector
  (lambda (RV adjustment-percentage)
    (local ((;; adjust-one-price: number -> number
              define adjust-one-price
                (lambda (n)
                  (/ (* n (+ 100 adjustment-percentage)) 100))))
      (vector-map
        __________
        (lambda (R) (__________ (rrant-name R) (rrant-cuisine R)
                      (rrant-phone R) (rrant-dish R)
                      (__________ (__________ __________))))))
```

SCORING: 1/2 point for each, but one point for adjust-one-price.

(c) (5 points) Complete the definition of vector-map below.

```
;; vector-map: vector-of-X (X -> Y) -> vector-of-Y
;; Applies the function to each element of the vector of Xs, returning a new vector of Ys.
(define vector-map
  (lambda (v f)
    (build-vector
      __________
      (lambda (i) (f (vector-ref v i))))))
```