CSE 341: Section 1 Worksheet

Q1: Suppose we evaluate the following Racket expressions:

```
(define x  '(apple banana))
(define y  '(racket haskell java python))
```

Draw box-and-arrow diagrams of the result of evaluating the following expressions. What parts of the list are created fresh, and which are shared with the variables x and y?

A. (cons ‘peach x)

B. (cons y y)

C. (append y y)

D. (cdr y)

E. (cadr y)

F. (cons ‘(peach watermelon) x)

Q2: What is the result of evaluating the following Racket expressions?

(a)

```
(let ([x 100]
      [y 5])
  (let ([x 1])
    (+ x y)))
```

(b)

```
(let ([x 5]
      [y 4])
  (let ([z 5]
        [y (+ x 2)])
    (let ([x (+ z y)])
      (+ x y z))))
```

(c)

```
(let* ([x 100]
       [y (* 2 x)]
       [x 50]
       [z (+ y x)])
  (+ y z))
```

(d) *Note: (list x y ...) creates a list of the form '(x y ...)*

```
(define bob (lambda (x) (list x)))
(define jane "Hamburger")
(let ([cathy (list jane jane)]
      [hank (bob jane)]
      [jane "Hotdog"])
      (let* ([bob (lambda (x) (list (car x) jane))]
             [jane (bob hank)])
            (cons (car cathy) (bob jane))))
```

<u>Q3</u>: Define a function, numOdds, that returns the number of odds within a list of numbers.
(Challenge: Can you make this tail recursive?)