Solutions

Exercise 1:

```
(define (merge L1 L2)
(cond
[(empty? L1) L2]
[(empty? L2) L1]
[(> (first L1) (first L2)) (cons (first L1) (merge (rest L1) L2))]
[else (cons (first L2) (merge L1 (rest L2)))]))
```

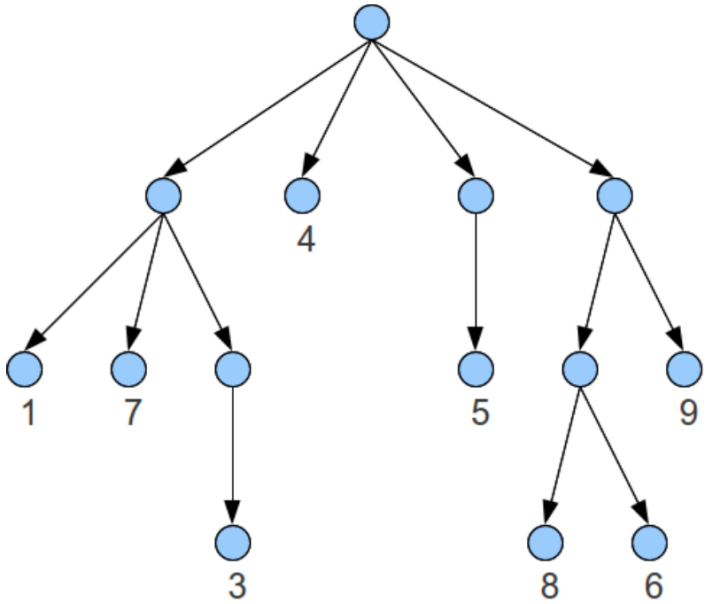
Exercise 2:

```
(define (singletons t)
(cond [(empty? t) 0]
[(and (empty? (node-left t)) (empty? (node-right t))) 0]
[(empty? (node-left t)) (+ 1 (singletons (node-right t)))]
[(empty? (node-right t)) (+ 1 (singletons (node-left t)))]
[else (+ (singletons (node-left t)) (singletons (node-right t)))]))
```

Alternative solution

```
(define (singletons t)
(cond [(empty? t) 0]
[(or (and (empty? (node-left t)) (not (empty? (node-right t))))
(and (empty? (node-right t)) (not (empty? (node-left t)))))
(+ 1 (singletons (node-right t)) (singletons (node-left t)))]
[else (+ (singletons (node-left t)) (singletons (node-right t)))]))
```

Exercise 3:



Exercise 4:

```
(define (sum-leaves tree)
(cond
[(empty? tree) 0]
[(cons? (first tree))
(+ (sum-leaves (first tree)) (sum-leaves (rest tree)))]
[else (+ (first tree) (sum-leaves (rest tree)))]))
```

Exercise 5: (e) is the correct selection.