1. Exercise 13.2-2 (p337)
   A rotation in a tree is defined on a parent-child edge, where a left rotation is possible when a node has a right child, and a right rotation when it has a left child. So, each edge in a tree corresponds to one possible rotation. Because we see that an n-node tree has $n-1$ edges, it has exactly $n-1$ possible rotations.

2. Exercise 13.3-1 (p346)
   If we set z's color to black, we violate property 5: for each node, all simple paths from the node to descendant leaves contain the same number of black nodes.

3. Exercise 13.3-2 (p346)

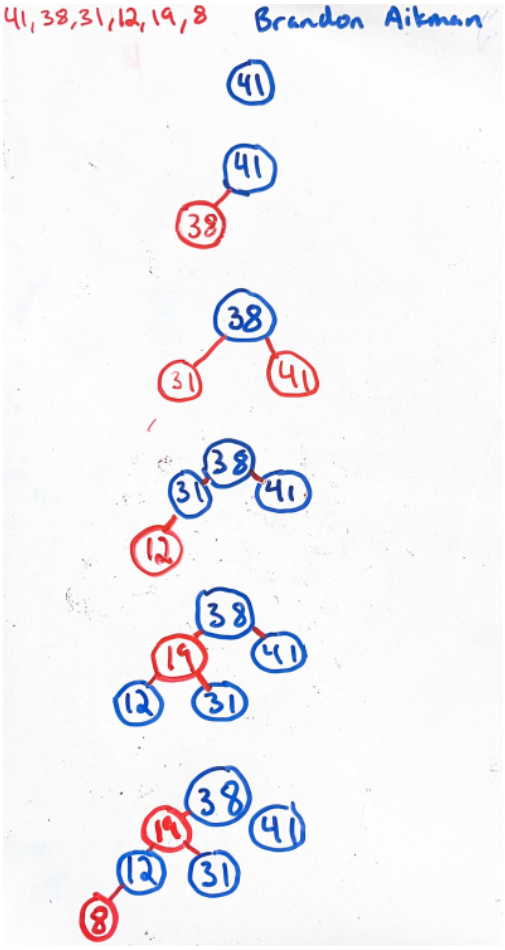

Figure 1: Red-Black Tree Insertions

4. Exercise 13.4-4 (p354)

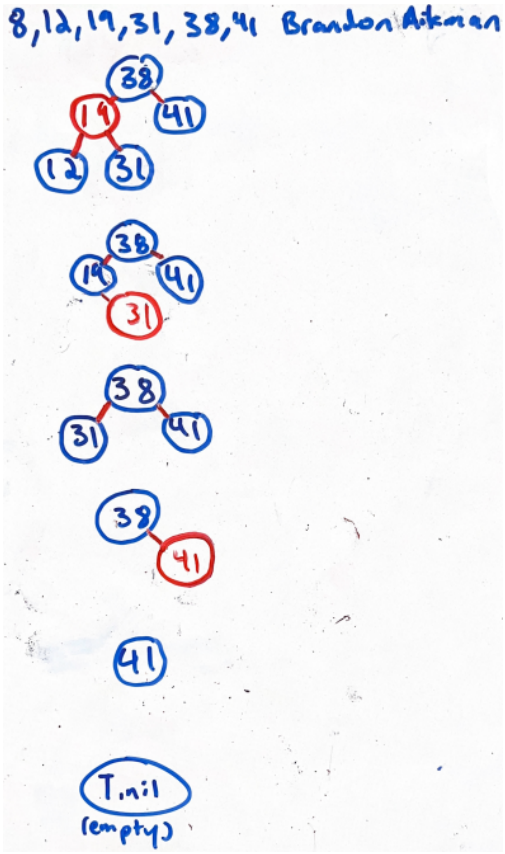

Figure 2: Red-Black Tree Deletes

5. Exercise 17.1-1 (p485)

```
key:  26; i = 10; r = 13
key:  17; i = 10; r = 8
key:  21; i = 2; r = 3
key:  19; i = 2; r = 1
key:  20; i = 1; r = 1 ⇒ return 20
```

6. Exercise 17.1-2 (p485)

```
x.key:  35
r = 1→3→16
y.key:  35→38→30→41→26
return r = 16
```

7. Exercise 17.3-1 (p495)

```
Left-Rotate(T,x)
  y = x.right
  x.right = y.left
  if y.left ≠ T.NIL
    y.left.p = x
  y.p = x.p
  if x.p == T.NIL
    T.root = y
  else if x == x.p.left
    x.p.left = y
  else
    x.p.right = y
  y.left = x
  x.p = y
  x.max = max(x.high, x.left.max, x.right.max)
  y.max = max(y.high, y.left.max, y.right.max)
```

8. Problem 17-2 (p496)

(a)
```
Josephus(n,m)
head = new Node(1)
prev = head
for i = 2 to n
  newNode = new Node(i)
  prev.next = newNode
  prev = newNode
prev.next = head

while prev != prev.next
  for j = 1 to m-1
    prev = prev.next
  print prev.next.key
  prev.next = prev.next.next

print prev.key
```

(b)
```
Josephus(n,m)
create order-statistic tree with keys 1...n
r = 0
for k = n downto 1
  r = (r + m - 1) mod k
  if r == 0
    r = k
  x = Select(T.root, r)
  print x.key
  Delete(T, x)
```