Vejledende løsninger

Opgave 1

Spørgsmål 1.1

```
boolean isEmpty() { return head == tail; }
```

Alternativt

```
boolean isEmpty() { return head.next == null; }
```

Spørgsmål 1.2

```
void insert(int value) {
    finger.next = new Node(value, finger.next);
    if (tail == finger)
        tail = finger.next;
}
```

Spørgsmål 1.3

```
int remove() {
    if (!more())
        throw new RuntimeException("illegal call of remove");
    int v = finger.next.value;
    if (tail == finger.next)
        tail = finger;
    finger.next = finger.next.next;
    return v;
}
```

Spørgsmål 1.4

```
void sort() {
    List newList = new List();
    while (!isEmpty())
        newList.append(removeMin());
    head.next = newList.head.next;
    tail = newList.tail;
}
```

Opgave 2

Spørgsmål 2.1

Der benyttes en rekursiv hjælpemetode `printLeavesAscendingRec`.

```
void printLeavesAscendingRec(Node t) {
    if (t == null)
        return;
    if (t.left == null && t.right == null)
        System.out.print(t.key + " ");
    printLeavesAscendingRec(t.left);
    printLeavesAscendingRec(t.right);
}

void printLeavesAscending() {
    printLeavesAscendingRec(root);
    System.out.println();
}
```

Sætningen

```
if (t.left == null && t.right == null)
    System.out.println(key + " ");
```

kan placeres før, imellem eller efter de to rekursive kald af `printLeavesAscendingRec`.

Spørgsmål 2.2

Der benyttes en rekursiv hjælpemetode `printLeaveDescendingRec`.

```
void printLeavesDescendingRec(Node t) {
    if (t == null)
        return;
    if (t.left == null && t.right == null)
        System.out.print(t.key + " ");
    printLeavesDescendingRec(t.right);
    printLeavesDescendingRec(t.left);
}

void printLeavesDescending() {
    printLeavesDescendingRec(root);
    System.out.println();
}
```

Spørgsmål 2.3

Til metoden er tilføjet sætningerne markeret med **fede** typer:

```
Node rotateWithLeftChild(Node k2) {
    Node k1 = k2.left;
    k2.left = k1.right;
    k1.right = k2;
    k1.dad = k2.dad;
    k2.dad = k1;
    if (k2.left != null)
        k2.left.dad = k2;
    return k1;
}
```

Opgave 3

Spørgsmål 3.1

Der udskrives

```
1 2 3
1 3 2
2 1 3
2 3 1
3 2 1
3 1 2
```

Dette kan f.eks. indses ved at opstille et skema over variablernes værdier umiddelbart før hvert kald af `pocus`.

i	j	a[0]	a[1]	a[2]	
-	-	1	2	3	
0	0	1	2	3	
1	1	1	2	3	
2	2	**1**	**2**	**3**	●
1	2	1	3	2	
2	2	**1**	**3**	**2**	●
0	1	2	1	3	
1	1	2	1	3	
2	2	**2**	**1**	**3**	●
1	2	2	3	1	
2	2	**2**	**3**	**1**	●
0	2	3	2	1	
1	1	3	2	1	
2	2	**3**	**2**	**1**	●
1	2	3	1	2	
2	2	**3**	**1**	**2**	●