Opgaveløsninger (sæt 3)

Opgave 1a (5.6)

```
void traverse_inorder(Node t) {
    while (true) {
        while (t != z) { stack.push(t); t = t.l; }
        if (stack.empty())
            return;
        t = stack.pop();
        visit(t);
        t = t.r;
    }
}
```

Opgave 1b (5.7)

```
void traverse_postorder(Node t) {
    while (true) {
        while (t != z)
           { stack.push(t.r); stack.push(t); t = t.l; }
        if (stack.empty())
            return;
        t = stack.pop();
        Node s = t.r;
        if (s != z) { stack.push(t); t.r = z; t = s; }
        else { visit(t); t.r = stack.pop(); t = z; }
    }
}
```

Opgave 2a (6.9)

```
int log2_rec(int N) {
    return N == 1 ? 0 : 1 + log2_rec(N/2);
}
```

Opgave 2a (6.10)

En kørsel med nedenstående program

```
public class Program {
    static int log2_rec(int N) {
        return N == 1 ? 0 : 1 + log2(N/2);
    }

    static int log2_it(int N) {
        int r = 0;
        while (N  > 1) { r++; N /= 2; }
        return r;
    }

    static int log2_java(int N) {
        return (int) Math.floor(Math.log(N)/Math.log(2));
    }

    public static void main(String args[]) {
        double startTime = System.currentTimeMillis();
        for (int i = 1; i <= 1000000; i++)
            log2_rec(i);
        IO.println("Rec:  " + (System.currentTimeMillis() -
                              startTime)/1000 + " seconds");
        startTime = System.currentTimeMillis();
        for (int i = 1; i <= 1000000; i++)
            log2_it(i);
        IO.println("It:   " + (System.currentTimeMillis() -
                              startTime)/1000 + " seconds");
        startTime = System.currentTimeMillis();
        for (int i = 1; i <= 1000000; i++)
            log2_java(i);
        IO.println("Java: " + (System.currentTimeMillis() -
                              startTime)/1000 + " seconds");
     }
}
```

gav følgende udskrift:

```
Rec:  22.699 seconds
It:   12.132 seconds
Java: 11.568 seconds
```

Opgave 3

```
     int height(Node root) {
         return root == null ? –1 :
                1 + Math.max(height(root.left),height(root.right));
     }
```

Opgave 4

```
int comb[];

void print_combinations(int k, int n) {
    comb = new int[k+1];
    choose(k, 1, n);
}

void choose(int k, int from, int to) {
// fyld comb[1..k] med en faldende talrække udtaget af from..to
    if (k == 0)
        { print_comb(); return; }
    for (int i = from; i <= to-k+1; i++) {
        comb[k] = i;
        choose(k-1, i+1, to);
    }
}

void print_comb() {
    for (int i = 1; i < comb.length; i++)
        IO.print(comb[i]);
    IO.println();
}
```

Kombinationerne udskrives i faldende orden. En løsning af opgaven, der udskriver kombinationerne i stigende orden, ses nedenfor.

```
void print_k_combinations2(int k, int n) {
    comb = new int[k+1];
    k_comb2(1, k, 1, n);
}

void k_comb2(int j, int k, int from, int to) {
// fyld comb[j..k] med en stigende talrække udtaget af from..to
    for (int i = from ; i <= to-(k-j+1)+1; i++) {
        comb[j] = i;
        if (j == k)
            print_comb();
        else
            k_comb2(j+1, k, i+1, to);
    }
}
```

Nedenfor ses en løsning af opgaven, der ikke benytter rekursion.

```
void print_k_combinations3(int k, int n) {
    comb = new int[k+1];
    k_comb3(k, n);
}

void k_comb3(int k, int n) {
    for (int i = 1; i <= k; i++)
        comb[i] = i;
    print_comb();
    int current = k;
    while (current >= 1)
        if (++comb[current] <= n - (k - current)) {
            for (int i = 1; i <= k - current; i++)
                 comb[current+i] = comb[current] + i;
            print_comb();
            current = k;
        }
        else
            current--;
}
```