



Aufgabe 1

3 * T

Wiederholung Rekurrenzen

Betrachten Sie das folgende Programm:

```
import Bits

radix_sort xs = radix_sort1 xs 0 (maximum xs)

radix_sort1 xs i m = if 2^i > m then rs
                    else radix_sort1 (ls++rs) (i+1) m
  where ls = [x | x <- xs, testBit x i]
        rs = [x | x <- xs, not (testBit x i)]
```

Welchen Aufwand hat das Sortierverfahren? Beweisen Sie Ihre Aussage.

Aufgabe 2

1 * + 1 * = 2 * T

Wiederholung Rekurrenzen

Bestimmen Sie eine geschlossene Form für die folgenden Rekurrenzen und beweisen Sie wenn nötig Ihre Aussage.

- a) $g_0 = 0, g_n = g_{n-1} + n^2$
b) $g_0 = 0, g_1 = 0, g_n = g_{n-1} + g_{n-2} + 1$

Aufgabe 3

2 * + 2 * = 4 * T

Wiederholung O-Kalkül

Beweisen oder widerlegen Sie die folgenden Aussagen:

- a) $O(\sqrt{n}) = O(\sqrt[3]{n})$
b) $O\left(\frac{n}{\log_2 n}\right) = O(n)$

Aufgabe 4

20R

Java, Programmierprojekt

In der Datei `Spielregeln` (enthalten in `Robo.tar.gz`) finden Sie eine Regelbeschreibung eines Spiels. Dieses Spiel sollen Sie in Java implementieren. Um Ihnen die Aufgabe zu erleichtern ist für diese Programmieraufgabe Gruppenarbeit mit bis zu drei Teilnehmern erwünscht. Das Spiel gliedert sich in zwei Teilbereiche: Einen Spielserver und einen Spielclient. Jede Gruppe hat die Wahl welchen Teil sie implementieren will.

Hinweise:

- Sie bekommen für beide Teile ein Programmgerüst auf das Sie aufbauen sollen in `Robo.tar.gz`.

Aufgabe 5

4T

Java, Vererbung

Betrachten Sie das folgende Programm:

```
class A {
  Number n;
  A(Number n) { this.n = n; }
  void method_1() { System.out.println("A_1: " + n); }
  void method_2(Number param) { System.out.println("A_2: " + param); }
}
```

```

class B extends A {
    B(Double d) { super(d); }
    void method_1() { System.out.println("B_1: " + n); }
    void method_2(Double param) { System.out.println("B_2: " + param); }
}
class Test {
    public static void main(String[] args) {
        A variableA = new A(new Double(1.0));
        B variableB = new B(new Double(2.0));
        variableA.method_1();
        variableB.method_1();
        ((A)variableB).method_1();
        System.out.println();
        variableA.method_2(new Double(3.0));
        variableB.method_2(new Double(4.0));
        ((A)variableB).method_2(new Double(5.0));
    }
}

```

Interpretieren Sie die Ausgaben des Programms.

Aufgabe 6

5T

Ausnahmen in Java

Es seien A und B Anweisungsfolgen, in denen Ausnahmen des Typs `Exception` ausgelöst werden können. Welche Unterschiede bestehen im Verhalten der folgenden Methoden?

```

void m1() throws Exception {
    A;
    B;
}
void m2() throws Exception {
    try {A;}
    catch (Exception e) {
        B;
    }
    return;
}
void m3() throws Exception {
    try {A;}
    finally {B;}
}
void m4() throws Exception {
    try {A;}
    catch (Exception e) {B;}
    finally { return;}
}

```

Aufgabe 7

5 * T

Sortieren durch Einfügen

Zeigen Sie, daß der maximale Aufwand von Sortieren durch Einfügen linear ist, d.h. $O(n)$, falls die gegebene Reihung $A[0 : n - 1]$ derart in Teilreihungen $A[j_0 : j_1 - 1]$, $A[j_1 : j_2 - 1]$, ..., $A[j_{p-1} : j_p]$ (mit $j_0 = 0$, $j_p = n - 1$) zerlegt werden kann (mit $k = \max\{(j_{i+1} - j_i) \mid i \in \{0, \dots, p - 1\}\}$, und $k \ll n$, k fest), so daß stets gilt:

$$\forall i \in \{0, \dots, p - 2\} \forall l \in \{j_i, \dots, j_{i+1} - 1\} \forall m \in \{j_{i+1}, \dots, j_{i+2} - 1\} : A[l] \leq A[m]$$