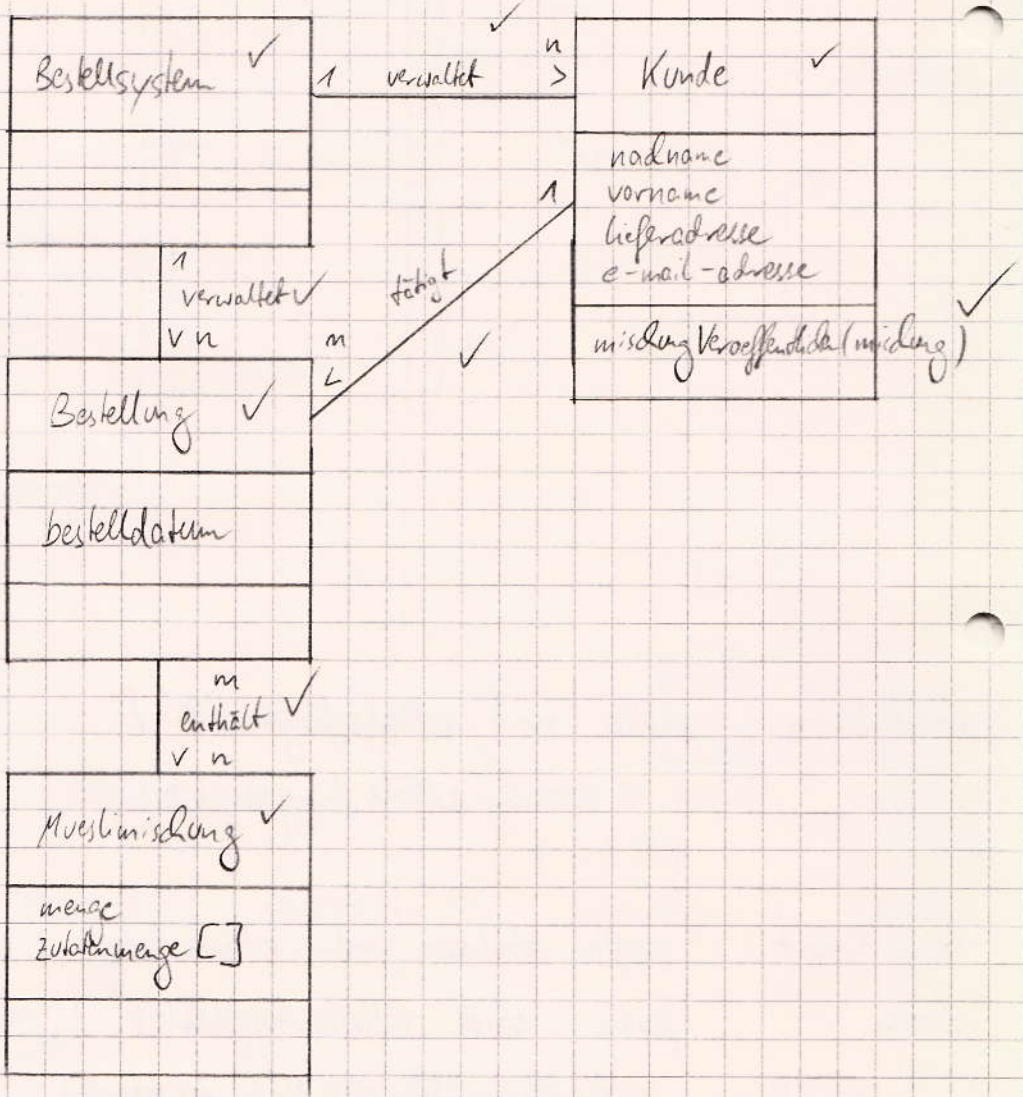
Informatik - I

1.) - Analyse: Auftraggeber schreibt gewünschte Leistungen (Lastenheft), Auftragnehmer erstellt Pflichtheft, in dem die zu erbringenden Leistungen stehen.
- Systementwurf: Struktur wird erstellt und dokumentiert
- Implementation: Umsetzung und Test der einzelnen (Software-) Teile.
- Zusammenführung: Die einzelnen Teile werden zusammengeführt und auf Funktionsfähigkeit getestet.

4

2.)



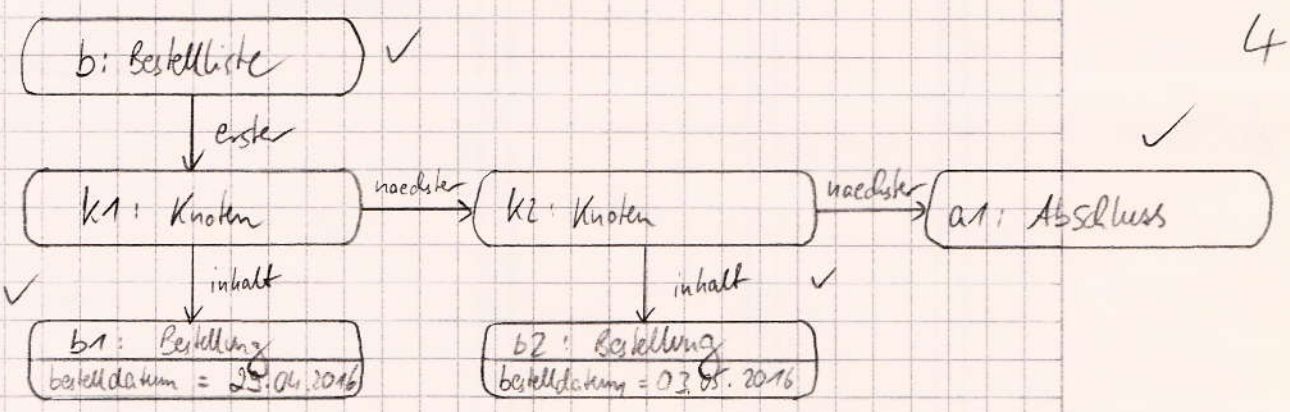
9

3a) Da die Anzahl der Bestellungen nicht vorhersehbar ist, hätte eine sehr großes Feld gewählt werden müssen, das unter Umständen zum großen Teil leer hätte sein können (⇒ unnötiger Speicherplatzverbrauch)

3

und das Entfernen des ersten Elements der Bestellliste gelingt in der verketteten Liste leichter, da die restlichen Elemente nicht verschoben werden müssen. ✓
Für ein Feld würde evtl. der direkte Zugriff auf jede Bestellung sparen, was in diesem Anwendungsfall jedoch keine große Rolle spielen dürfte. ✓

b)



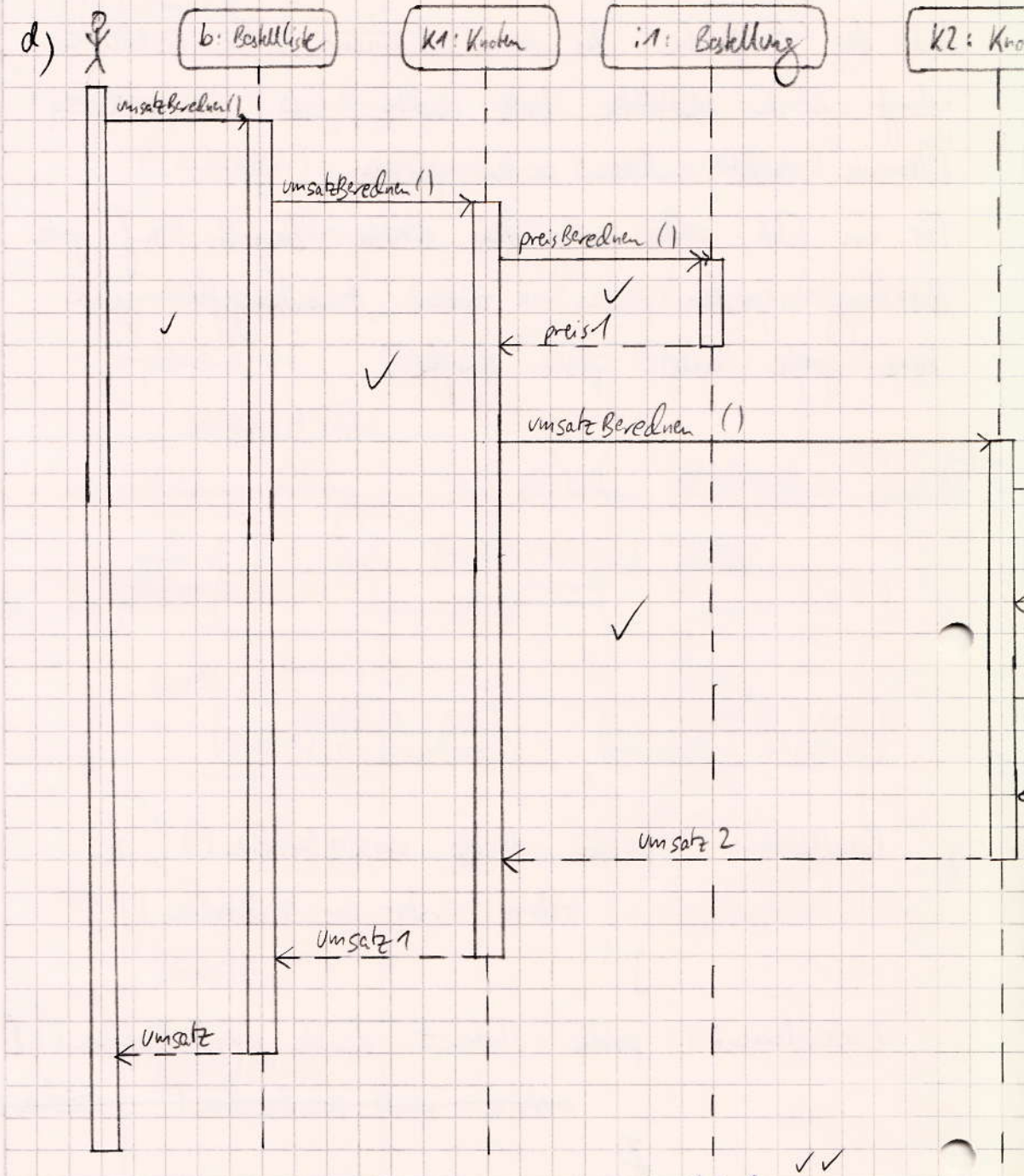
✓ ✓ ✓ ✓ 4

c)

```
Bestellliste:   public double umsatzBerechnen() {
                    return erster.umsatzBerechnen();
                }

Listenelement:  public abstract double umsatzBerechnen();

Knoten:         public double umsatzBerechnen() {
                    return inhalt.preisBerechnen() + naechster.umsatzBerechnen();
                }

Abschluss:      public double umsatzBerechnen() {
                    return 0;
                }
```

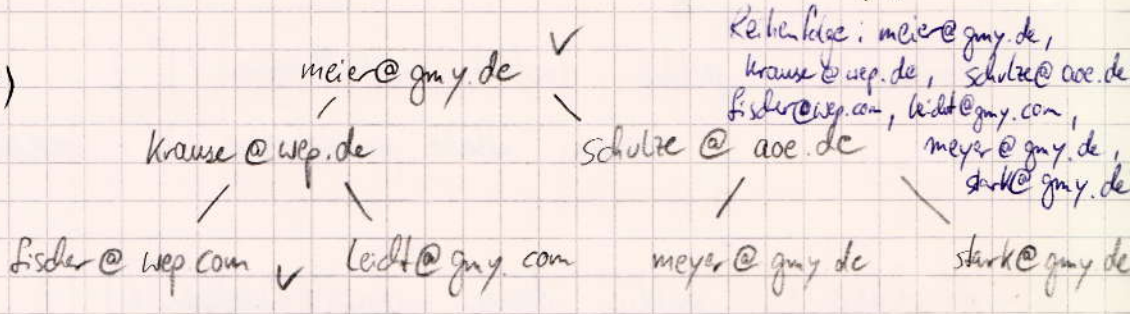
✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ 8

d)



6

4a)



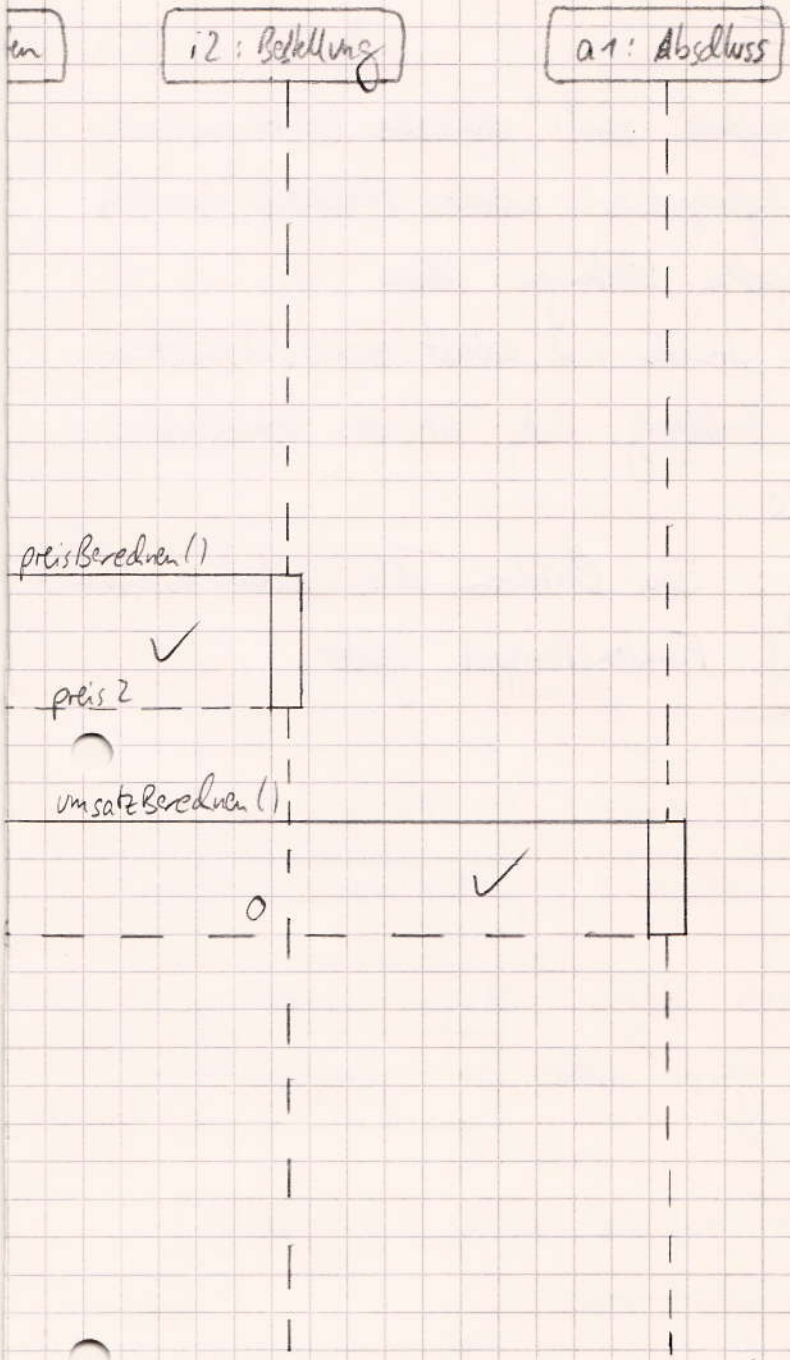
4

b) $\log_2 10000 \approx 13{,}\overline{3}$

Es genügen also 14 Vergleiche (im worst case)

4

c) Jeder Knoten hat jetzt zwei (statt vorher einen) Nachfolger.



c) Kundenbaum : public void eMailsAusgeben() {
    wurzel.eMailsAusgeben();
}

Baumelement : public abstract void eMailsAusgeben();

Knoten : public void eMailsAusgeben() {
    linkerNachfolger.eMailsAusgeben();
    inhalt.emailadresseAusgeben();
    rechterNachfolger.eMailsAusgeben();
}

Abschluss : public void eMailsAusgeben() { }

Inf - I
Bogen 2

6

5.) a)

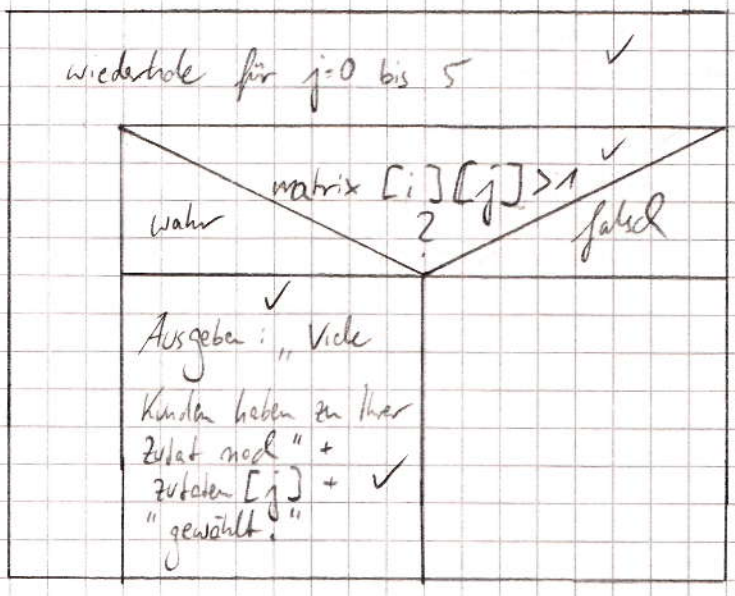
	HF	DF	Rosinen	Mandeln	Schokolade	Banane	
HF	0	1	1	3	2	0	✓
DF	1	0	0	0	0	0	✓
Rosinen	1	0	0	1	0	0	✓
Mandeln	3	0	1	0	2	0	✓
Schokolade	2	0	0	2	0	0	
Banane	0	0	0	0	0	0	

Der Graph ist ungerichtet ✓, gerichtet ✓ (, unzusammenhängend, zyklisch).

6

b)
```
public class Mueslimatrix {
    private String [] zutaten;                    ✓
    private int [][] matrix;                      ✓
    public Mueslimatrix () {
        zutaten = new String [6];                 (✓)
        zutaten [0] = "Haferflocken";
        zutaten [1] = "Dinkelflocken";            ✓
        ...
        zutaten [5] = "Banane";
        matrix = new int [6][6];                  (✓)
        for (int i=0; i<6; i++) {                 ✓
            for (int j=0; j<6; j++) {
                matrix [i][j] = 0                 ✓
            }
        }
    }
    .
    .
}
```

c)



5

d) Verwende in der Klasse Mueslimatrix zusätzlich ein Boolean-Feld besucht.

```
tiefensuche (startknoten) {
    wiederhole von i=0 bis zutatenanzahl - 1
        setze besucht[i] = false
    besuchen (startknoten)
}

besuchen (knoten) {
    besucht[knoten] = true;
    ausgeben (knoten)
    wiederhole von i=0 bis zutatenanzahl - 1
        wenn matrix[knoten][i] > 0 && besucht[i] == false
            besuchen (i)
}
```

8

e)
```
SELECT bezeichner FROM zutaten
WHERE menge < verbrauch;
```

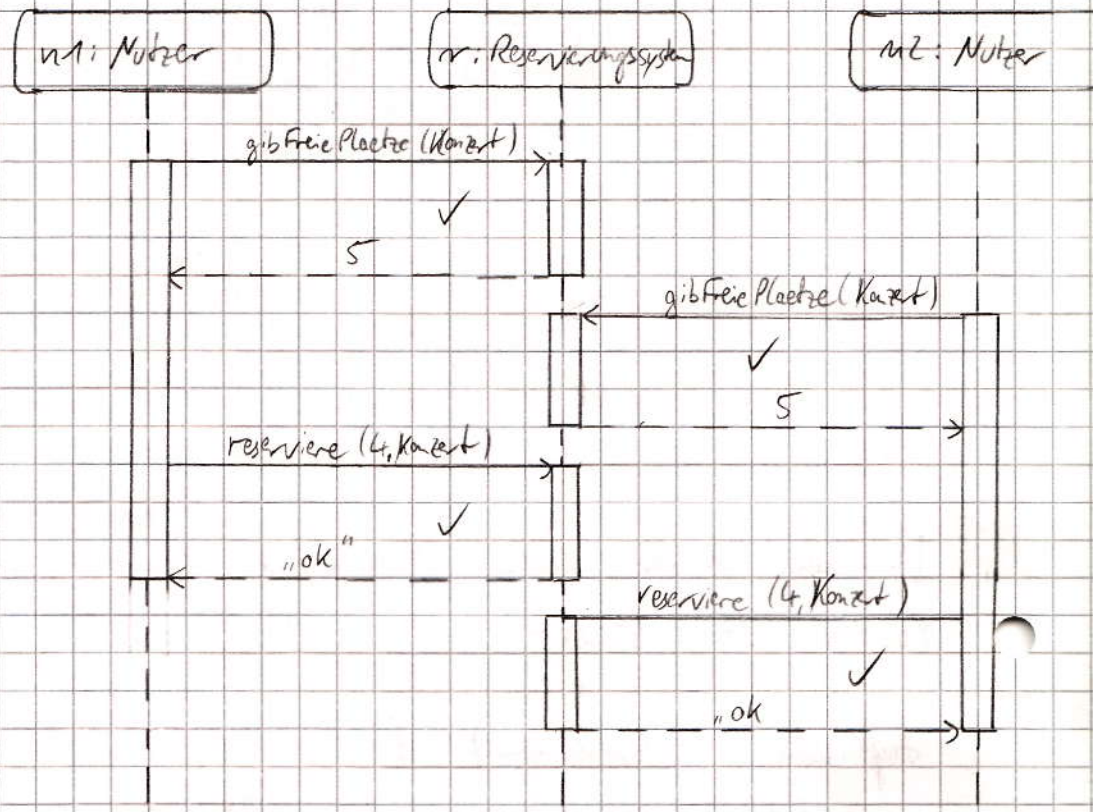
3

f) 1.) Verbindung mit Datenbank aufbauen
2.) Abfrage O senden
3.) Antworttabelle auslesen
4.) Verbindung schließen

2

Informatik – III

1a)



5

Erfolgt bei der Reservierung keine erneute Überprüfung der freien Plätze, so würden im obigen Fall mehr Plätze reserviert als verfügbar sind. ✓

b) Monitore können gewährleisten, dass Methoden, die durch einen Monitor überwacht werden, immer nur von einem Prozess genutzt werden können. ✓

3

In obigem Fall müssen die Methoden gibFreiePlaetze (...) und reservieren (...) durch einen Monitor vor mehrfachem gleichzeitigen Zugriff geschützt werden. ✓

2.) Eine Verklemmung liegt vor, wenn mehrere Prozesse zur Fertigstellung mehrere Resourcen benötigen und diese erst wieder freigeben, wenn der Prozess beendet wurde. ✓

4

Die Verklemmung entsteht dadurch, dass Prozesse Resourcen belegen, die andere Prozesse zur Fertigstellung benötigen und umgekehrt. ✓

Im Beispiel: A2 und A3 stehen auf Rot und von beiden Seiten fahren lange Kolonnen ein, die sich jeweils in den einspurigen Bereich stauen. ✓ Hierdurch werden A2 und A3 dauerhaft daran gehindert, auf Grün zu schalten und der Stau ist dauerhaft.

3a)

	A	SR	PC	100	101
	0		0	17	2
	2		2	17	2
	2		6	17	2
	1		8	17	2
	1		10	17	1
	17		12	17	1
	18		14	17	1
	18		16	18	1
	18		0	18	1
	1		2	18	1
	1		6	18	1
	0	Z	8	18	1
	0	Z	10	18	0
	18		12	18	0
	19		14	18	0
	19		16	19	0
	19		0	19	0
	0	Z	2	19	0
	0	Z	4	19	0
→	0	Z	6	19	0

5

```
wiederhole solange b > 0
    b = b - 1
    a = a + 1
```

✓ ✓

b: Wert in 101

a: Wert in 100

b)

```
anfang :  LOAD 101
          JMPP positiv
          JMPZ fertig
          JMPNP negativ
positiv : DEC
          STORE 101
          LOAD 100
          INC
          STORE 100
          JMP anfang

negativ : INC
          STORE 101
          LOAD 100
          DEC
          STORE 100
          JMP anfang
fertig :  HOLD
```

✓ ✓ ✓ ✓ ✓

5

Inf - III

Bogen 3

4a) GTK = Landesk ("-" | " ") [Spezk] Nummer " " Abk ✓

Landesk = b [b] [b] ✓

Spezk = b [b] " "

Nummer = zzz {z} ✓

Abk = b b

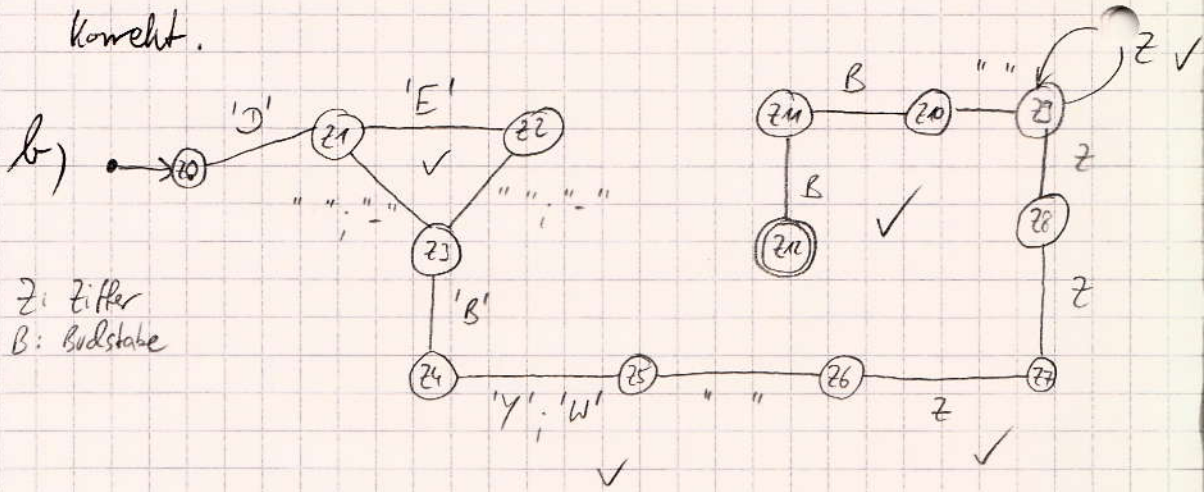
b = "A" | "B" | ... | "Z"

z = "0" | "1" | ... | "9"

6

Nicht alle Buchstabenkombinationen aus maximal drei Buchstaben ✓ kennzeichnen ein Land, sind also somit nicht semantisch korrekt.

b)



5

c) public class Automat {

private int zustand; ✓

public boolean KennungPruefen (String k) {

zustand = 0;

for (int i = 0; i < k.length(); i++) {

zeichenBearbeiten (k.zeichenAn(i)); ✓

}

return (zustand == 12); ✓

}

7

```
private void zeichenBearbeiten (char z) {
    switch (zustand) {
        ...
        case 1 : { if (z == 'E') zustand = 2;
                   else if (z == ' ' || z == '-') zustand = 3;
                   else zustand = -1; // Fehlerzustand
                   break;
                 }
        ...
    }
}
}
```