

Abiturprüfung 2016

INFORMATIK

Hinweise zur Korrektur und Bewertung

(nicht für den Prüfling bestimmt)

Die Korrekturhinweise enthalten keine vollständigen Lösungen der Aufgaben, sondern einen Abriss des Erwartungshorizontes. Nicht genannte, aber *gleichwertige* Lösungswege und Begründungsansätze sind *gleichberechtigt*.

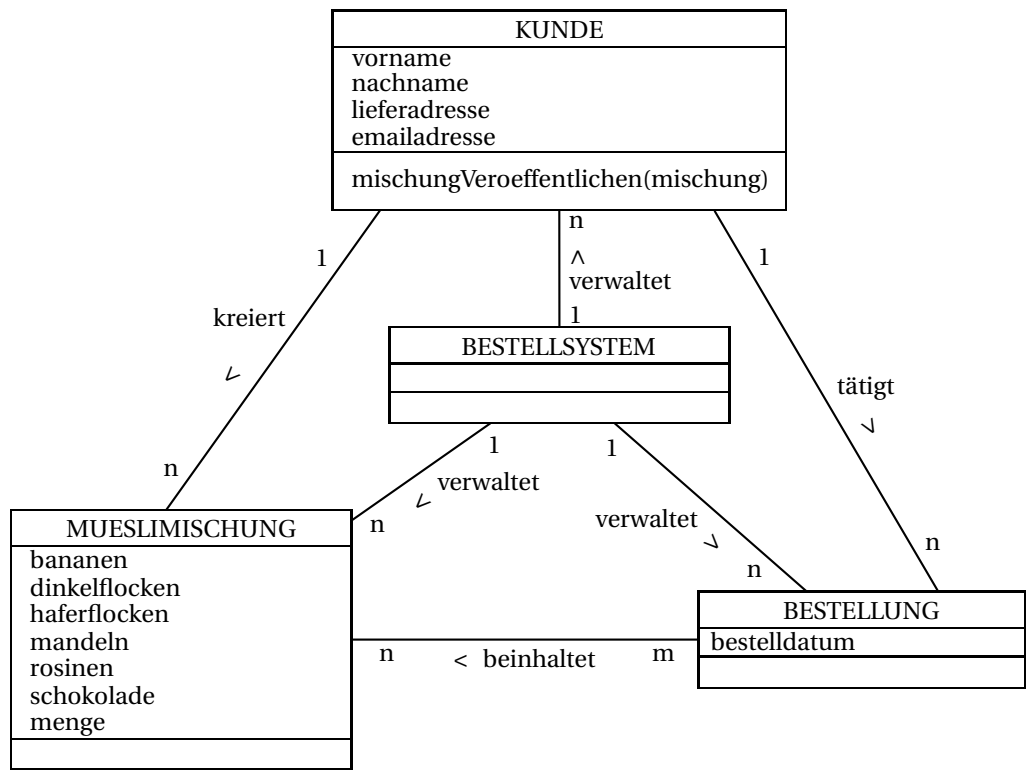
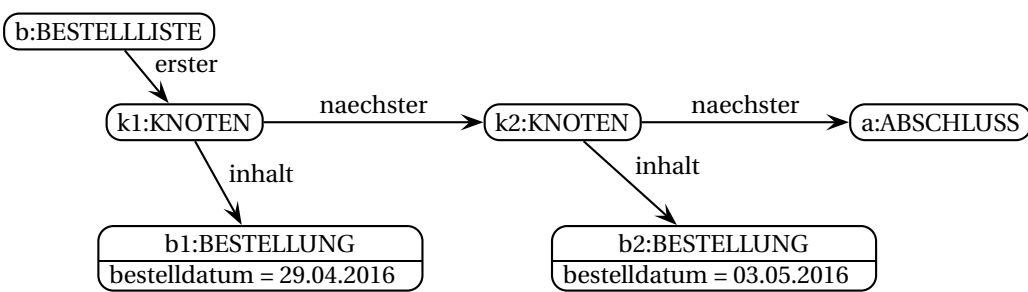
Die in den Lösungshinweisen gewählte objektorientierte Programmiersprache ist Java.

Die Bewertung der erbrachten Prüfungsleistungen hat sich an der bei jeder Teilaufgabe am linken Rand des Angabenblattes vermerkten, maximal erreichbaren Zahl von Bewertungseinheiten (BE) zu orientieren.

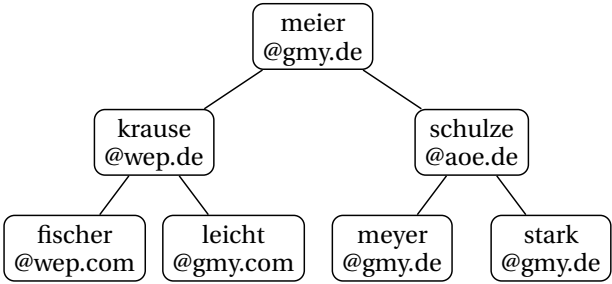
Umrechnung der erreichten Bewertungseinheiten in Notenpunkte:

Die insgesamt erreichten Bewertungseinheiten werden nach der folgenden Tabelle in Notenpunkte umgesetzt:

Notenpunkte	Notenstufen	Bewertungseinheiten	Intervalle in %
15	+1	120 ... 115	15
14	1	114 ... 109	
13	1–	108 ... 103	
12	+2	102 ... 97	15
11	2	96 ... 91	
10	2–	90 ... 85	
9	+3	84 ... 79	15
8	3	78 ... 73	
7	3–	72 ... 67	
6	+4	66 ... 61	15
5	4	60 ... 55	
4	4–	54 ... 49	
3	+5	48 ... 41	20
2	5	40 ... 33	
1	5–	32 ... 25	
0	6	24 ... 0	20

Aufgabe	BE	Hinweise
1.	4	Es kann z. B. das Wasserfallmodell zugrunde gelegt werden.
2.	9	 <pre> classDiagram class KUNDE { vorname nachname lieferadresse emailadresse <<operation>> mischungVeroeffentlichen(mischung) } class BESTELLSYSTEM { } class MUESLIMISCHUNG { bananen dinkelflocken haferflocken mandeln rosinen schokolade menge } class BESTELLUNG { bestelldatum } KUNDE "1" -- "n" MUESLIMISCHUNG : kreiert KUNDE "1" -- "1" BESTELLSYSTEM : ^ verwaltet KUNDE "1" -- "n" BESTELLUNG : tätigt BESTELLSYSTEM "1" -- "n" MUESLIMISCHUNG : verwaltet BESTELLSYSTEM "1" -- "n" BESTELLUNG : verwaltet MUESLIMISCHUNG "n" -- "m" BESTELLUNG : < beinhaltet </pre> <i>Hinweis:</i> Nicht alle im Klassendiagramm angegebenen Beziehungen sind zwingend erforderlich; es muss sich jedoch eine schlüssige Modellierung ergeben.
3. a)	3	Es kann für oder gegen die Verwendung des Feldes argumentiert werden.
b)	4	 <pre> graph LR b[BESTELLISTE] -- erster --> k1[KNOTEN] k1 -- naechster --> k2[KNOTEN] k2 -- naechster --> a[ABSCHLUSS] k1 -- inhalt --> b1[BESTELLUNG bestelldatum = 29.04.2016] k2 -- inhalt --> b2[BESTELLUNG bestelldatum = 03.05.2016] </pre>

c)	8	In BESTELLISTE: <pre>float umsatzBerechnen(){ return erster.umsatzBerechnen(); }</pre> In LISTENELEMENT: <pre>abstract float umsatzBerechnen();</pre> In ABSCHLUSS: <pre>float umsatzBerechnen(){ return 0; }</pre> In KNOTEN: <pre>float umsatzBerechnen(){ return inhalt.preisBerechnen() + naechster.umsatzBerechnen(); }</pre>
d)	6	<pre> sequenceDiagram participant b as b:BESTELLISTE participant k1 as k1:KNOTEN participant b1 as b1:BESTELLUNG participant k2 as k2:KNOTEN participant b2 as b2:BESTELLUNG participant a as a:ABSCHLUSS b->>b: umsatzBerechnen() activate b b->>k1: umsatzBerechnen() activate k1 k1->>b1: preisBerechnen() activate b1 b1-->>k1: 15,30 deactivate b1 k1->>k2: umsatzBerechnen() activate k2 k2->>b2: preisBerechnen() activate b2 b2-->>k2: 10,00 deactivate b2 k2->>a: umsatzBerechnen() activate a a-->>k2: 0 deactivate a k2-->>k1: 10,00 deactivate k2 k1-->>b: 25,30 deactivate k1 b-->>: 25,30 deactivate b </pre>

4. a)	4	 <pre> graph TD A["meier @gmy.de"] --> B["krause @wep.de"] A --> C["schulze @aoe.de"] B --> D["fischer @wep.com"] B --> E["leicht @gmy.com"] C --> F["meyer @gmy.de"] C --> G["stark @gmy.de"] </pre> Mögliche Reihenfolge: meier@gmy.de, krause@wep.de, fischer@wep.com, leicht@gmy.com, schulze@aoe.de, meyer@gmy.de, stark@gmy.de
b)	4	Für 10 000 Kunden ist ein Baum mit 14 Ebenen ausreichend: $2^{13} - 1 < 10\,000 < 2^{14} - 1$. Die maximale Anzahl der Vergleiche beträgt folglich 14.
c)	8	Ein Knoten hat jetzt zwei anstelle von einem Nachfolger. In BINBAUM: <pre>void emaillisteDrucken(){ wurzel.emailDrucken(); }</pre> In BAUMELEMENT: <pre>abstract void emailDrucken();</pre> In ABSCHLUSS: <pre>void emailDrucken(){ }</pre> In KNOTEN: <pre>void emailDrucken(){ naechsterLinks.emailDrucken(); inhalt.emailadresseAusgeben(); naechsterRechts.emailDrucken(); }</pre>

5. a)	6	Der Graph ist ungerichtet, bewertet, unzusammenhängend und zyklisch. <table><tr><td></td><td>B.</td><td>D.</td><td>H.</td><td>M.</td><td>R.</td><td>S.</td></tr><tr><td>Bananen</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>Dinkelflocken</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>Haferflocken</td><td>0</td><td>1</td><td>0</td><td>3</td><td>1</td><td>2</td></tr><tr><td>Mandeln</td><td>0</td><td>0</td><td>3</td><td>0</td><td>1</td><td>2</td></tr><tr><td>Rosinen</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>Schokolade</td><td>0</td><td>0</td><td>2</td><td>2</td><td>0</td><td>0</td></tr></table>		B.	D.	H.	M.	R.	S.	Bananen	0	0	0	0	0	0	Dinkelflocken	0	0	1	0	0	0	Haferflocken	0	1	0	3	1	2	Mandeln	0	0	3	0	1	2	Rosinen	0	0	1	1	0	0	Schokolade	0	0	2	2	0	0
	B.	D.	H.	M.	R.	S.																																													
Bananen	0	0	0	0	0	0																																													
Dinkelflocken	0	0	1	0	0	0																																													
Haferflocken	0	1	0	3	1	2																																													
Mandeln	0	0	3	0	1	2																																													
Rosinen	0	0	1	1	0	0																																													
Schokolade	0	0	2	2	0	0																																													
b)	6	<pre>private String[] zutaten; private int[] [] adjazenzmatrix; Mueslimatrix(){ zutaten = new String[6]; zutaten[0] = "Bananen"; zutaten[1] = "Dinkelflocken"; zutaten[2] = "Haferflocken"; zutaten[3] = "Mandeln"; zutaten[4] = "Rosinen"; zutaten[5] = "Schokolade"; adjazenzmatrix = new int[6][6]; for(int i=0; i<6; i=i+1){ for(int j=0; j<6; j=j+1){ adjazenzmatrix[i][j] = 0; } } }</pre>																																																	

c)	5	Methode dazuEmpfehlen(i) zähle von j=0 bis j=5 falls adjazenzmatrix[i] [j] >1 wahr Ausgeben von „Viele Kunden haben zu Ihrer Zutat noch “+ zutaten [j] +„ gewählt.“ falsch
d)	8	Eine mögliche Lösung ist die Tiefensuche: Die Klasse MUESLIMATRIX benötigt zusätzlich das Attribut besucht als ein Feld von Wahrheitswerten. Methode tiefensuche (startknoten) zähle i von 0 bis knotenanzahl – 1 besucht [i] = falsch endeZähle besuchen(startknoten) endeMethode Methode besuchen(knoten) wenn nicht besucht [knoten] dann besucht [knoten] = wahr ausgeben(knoten) zähle i von 0 bis knotenanzahl – 1 wenn adjazenzmatrix [knoten] [i] > 0 dann besuchen(i) endeWenn endeZähle endeWenn endeMethode
e)	3	SELECT bezeichner FROM zutaten WHERE menge < verbrauch;
f)	2	Verbindung mit der Datenbank, Übermittlung der Abfrage, Auswertung der Ergebnistabelle, Schließen der Verbindung
	80	

Aufgabe	BE	Hinweise
1. a)	4	<pre> classDiagram class EINHEIT { laenge laengeGeben() } class LOKOMOTIVE { leistung } class PERSONENWAGEN { sitzplatzanzahl } class GEPAECKWAGEN { lademassee } EINHEIT < -- LOKOMOTIVE EINHEIT < -- PERSONENWAGEN EINHEIT < -- GEPAECKWAGEN </pre>
b)	12	<pre> class Zug{ private int zugnummer; private int anzahl; private Einheit[] einheiten; Zug(int zugnr, Lokomotive lok){ zugnummer = zugnr; anzahl = 1; einheiten = new Einheit[20]; einheiten[0] = lok; } void hinzufuegen(Einheit einheit){ if(anzahl < 20){ einheiten[anzahl] = einheit; anzahl++; } } } </pre>

		<pre> void umkehren(){ Einheit[] einheitenNeu = new Einheit[20]; einheitenNeu[0] = einheiten[0]; for(int i = 1; i < anzahl; i++){ einheitenNeu[i] = einheiten[anzahl-i]; } einheiten = einheitenNeu; } </pre>
2. a)	5	<pre> graph TD b["b: BUCHUNGSLISTE"] -- anfang --> k1["k1: KNOTEN"] k1 -- nachfolger --> k2["k2: KNOTEN"] k2 -- nachfolger --> k3["k3: KNOTEN"] k3 -- nachfolger --> a1["a1: ABSCHLUSS"] k1 -- daten --> b1["b1: BUCHUNG"] k2 -- daten --> b2["b2: BUCHUNG"] k3 -- daten --> b3["b3: BUCHUNG"] </pre>
b)	12	In BUCHUNGSLISTE: <pre> int anzahlReisendeGeben(int bahnhofsnummer){ return anfang.personenanzahlGeben(bahnhofsnummer); } </pre> In LISTENELEMENT: <pre> abstract int personenanzahlGeben(int bahnhofsnummer); </pre> In KNOTEN: <pre> int personenanzahlGeben(int bahnhofsnummer){ return daten.personenanzahlGeben(bahnhofsnummer) + nachfolger.personenanzahlGeben(bahnhofsnummer); } </pre>

In ABSCHLUSS:

```
int personenanzahlGeben(int bahnhofsnummer){
    return 0;
}
```

In BUCHUNG:

```
int personenanzahlGeben(int bahnhofsnummer){
    if ((bahnhof1 <= bahnhofsnummer)
        && (bahnhofsnummer < bahnhof2)){
        return personenanzahl;
    }else{
        return 0;
    }
}
```

c)

2

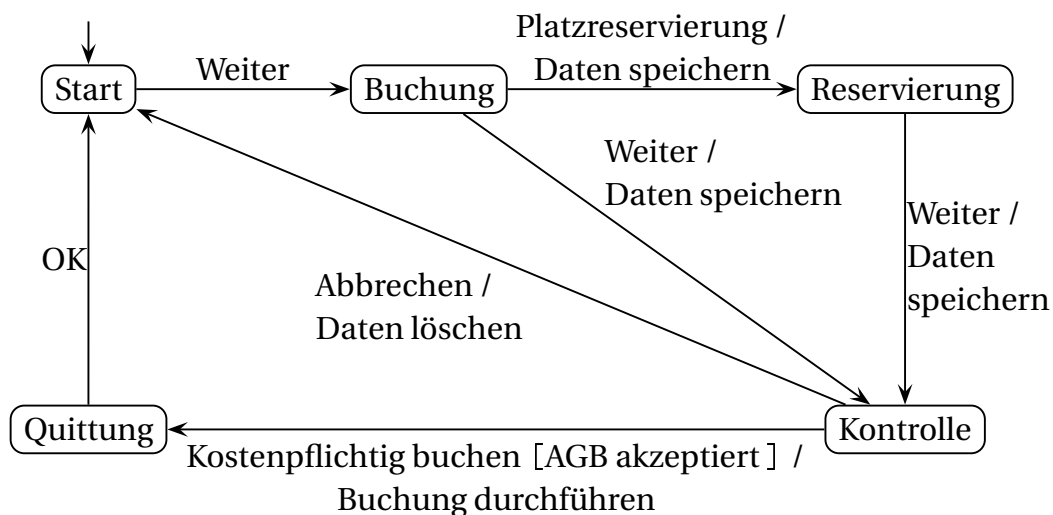
Mögliche Argumente sind z. B.:

Da die Anzahl der Wagen eines Zuges veränderlich ist, hat man sich für den Einsatz einer verketteten Liste entschieden.

Da die Anzahl der Sitzplätze in einem Personenwagen sich normalerweise nicht ändert, empfiehlt sich der Einsatz eines Feldes.

d)

8

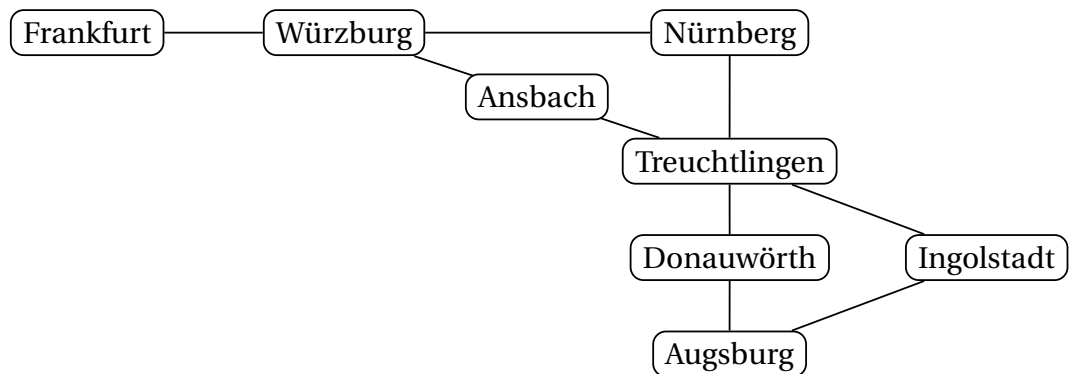


3. a)	3	SELECT kundenNr FROM kunden WHERE vorname = ‘Herbert’ AND nachname = ‘Huber’;
b)	3	Beispielsweise kann man durch Angabe eines fremden Namens bei diesem Verfahren Kenntnis über die zugehörige Kundennummer erhalten. Damit erhält man z. B. auch Kenntnis darüber, ob die genannte Person Kunde des Unternehmens ist. Das Verfahren ließe sich durch die zusätzliche Abfrage von Informationen verbessern, die tatsächlich nur der jeweilige Kunde selbst kennt.
c)	4	<pre> graph TD Adenauer --> Erhard Erhard --> Brandt Erhard --> Kiesinger Kiesinger --> Schmidt Schmidt --> Kohl Schmidt --> Schröder Kohl --> Merkel </pre>
d)	6	Günstigster Fall: $\log_2 2\,000\,001 \approx 20,9$. Es werden also nur 21 Ebenen benötigt. Ungünstigster Fall: Für jeden Knoten wird eine eigene Ebene benötigt. Es werden also 2 000 000 Ebenen benötigt. Bei der Suche im geordneten Binärbaum ist die maximale Anzahl an Vergleichen gleich der Anzahl der Ebenen. Die Unterschiede in der Anzahl der Ebenen machen deutlich, dass ein Binärbaum nur dann für schnelle Suchvorgänge geeignet ist, wenn er einigermaßen gut ausbalanciert ist.
4. a)	2	Da in der Liste der Strecken immer beide Richtungen vorkommen, ist der Graph ungerichtet. Da keine Entfernungen oder Ähnliches angegeben sind, ist der Graph unbewertet.

b)

6

Graph:



Adjazenzmatrix:

	Au.	Do.	In.	Tr.	Nü.	An.	Wü.	Fr.
Augsburg		x	x					
Donauwörth	x			x				
Ingolstadt	x			x				
Treuchtlingen		x	x		x	x		
Nürnberg				x			x	
Ansbach				x			x	
Würzburg					x	x		x
Frankfurt							x	

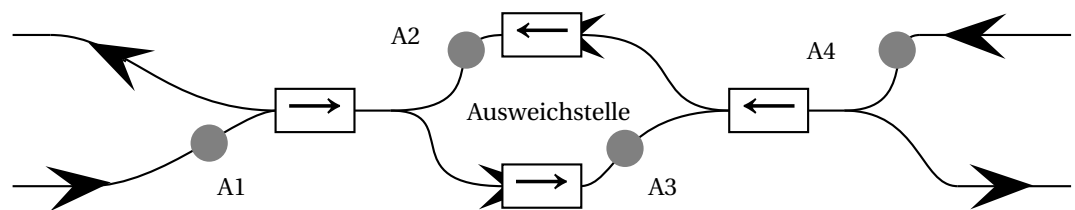
c)

3

Eine Argumentation kann z. B. mit Hilfe der Tiefensuche erfolgen.

d)	10	<pre> class Streckengraph{ private int knotenzahl; private String[] bahnhoefe; private boolean[] [] strecken; Streckengraph(){ knotenzahl = 0; bahnhoefe = new String[100]; strecken = new boolean[100][100]; } void bahnhofHinzufuegen(String bahnhofsname){ bahnhoefe[knotenzahl] = bahnhofsname; knotenzahl++; } void streckeHinzufuegen(String bahnhofsname1, String bahnhofsname2){ int bahnhofsnummer1 = bahnhofsnummerGeben(bahnhofsname1); int bahnhofsnummer2 = bahnhofsnummerGeben(bahnhofsname2); strecken[bahnhofsnummer1][bahnhofsnummer2] = true; strecken[bahnhofsnummer2][bahnhofsnummer1] = true; } } </pre>
	80	

Aufgabe	BE	Hinweise
1. a)	5	<pre> sequenceDiagram participant n1 as n1:NUTZER participant rs as rs:RESERVIERUNGSSYSTEM participant n2 as n2:NUTZER n1->>rs: plaetzeGeben(„K1“) rs-->n1: 7 n2->>rs: plaetzeGeben(„K1“) rs-->n2: 7 n1->>rs: reservieren(„K1“, 4) rs-->n1: „ok“ n2->>rs: reservieren(„K1“, 5) rs-->n2: „ok“ </pre>
b)	3	Monitorkonzept: Eine Menge von Methoden, die nur von einem Prozess zur selben Zeit benutzt werden dürfen, werden zum Monitor zusammengefasst. Nutzt ein Prozess den Monitor, so müssen andere Prozesse, die ebenfalls diesen Monitor nutzen wollen, warten. Im Zusammenhang mit Teilaufgabe 1a können die Methoden <i>plaetzeGeben</i> und <i>reservieren</i> z. B. zu einer Methode zusammengefasst werden; diese muss dann zu einem Monitor erklärt werden.
2.	4	Eine Menge von Prozessen $\{P_1, \dots, P_n\}$ ist in einer Verklemmung, wenn jeder Prozess auf ein Ereignis wartet, das nur von einem anderen Prozess dieser Menge ausgelöst werden kann. Beispiel: Zwischen A1 und A4 befindet sich kein Fahrzeug. A1 und A4 zeigen „grün“; A2 und A3 zeigen „rot“. Eine lange Fahrzeugkolonne fährt jeweils bei A1 und A4 ein und kommt bei A3 bzw. A2 zum Stehen. Aufgrund ihrer Länge stauen sich die Fahrzeuge in den einspurigen Bereich zurück. A1 und A4 schalten dann zwar auf „rot“; da die einspurigen Bereiche aber nicht frei sind, zeigen A2 und A3 ebenfalls weiter „rot“. Der Verkehr steht dauerhaft.



3. a)

5

 $[PC] = 6; [Z] = 1; [N] = 0; [A] = 0; [100] = 19; [101] = 0$

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

b entspricht dem Inhalt der Speicherzelle 101 und a dem Inhalt der Speicherzelle 100.

Hinweis: Abhängig vom verwendeten System kann der Befehlszähler (PC) auch andere Werte annehmen, beispielsweise 4 oder 5.

b)

5

```

0:  load 101
2:  jmp 18
4:  jmpz 30
6:  inc
8:  store 101
10: load 100
12: dec
14: store 100
16: jmp 0
18: dec
20: store 101
22: load 100
24: inc
26: store 100
28: jmp 0
30: hold

```

4. a)	6	$gtk = b[b][b] ("-" " ") [b[b] " "] zzz\{z\} " " bb.$ $b = "A" "B" \dots "Z".$ $z = "0" "1" \dots "9".$ Erläuterung an einem Beispiel: Die Kennung XYZ-1370 EG ist syntaktisch, aber nicht semantisch korrekt, da XYZ keinen Staat der EU repräsentiert.
b)	5	

c)	7	<pre>class Automat{ private int zustand; private void zeichenVerarbeiten(char c){ switch (zustand){ ... case 2: switch (c){ case 'E': zustand = 3; break; case ' ': zustand = 4; break; case '-': zustand = 4; break; default: zustand = -1; } break; ... } } boolean kennungTesten(String w){ zustand = 1; for (int i=0; i<w.length(); i++){ zeichenVerarbeiten(w.charAt(i)); } return (zustand == 13); } }</pre>
	40	

Informatik

Aufgabe Inf2.IV

Aufgabe	BE	Hinweise
1. a)	2	i) Es ist ein Crystalball-Stab, da die oberste Kugel wie auch die vorletzte Kugel schwarz ist und da er aus mindestens zwei Kugeln besteht. ii) Es ist kein Crystalball-Stab, da die oberste Kugel schwarz, der Rest der Kugeln aber weiß ist.
b)	8	
c)	3	$\langle \text{Stab} \rangle \xrightarrow{R1} s \langle S \rangle \xrightarrow{R2} sw \langle Z \rangle \xrightarrow{R4} sws$ ist die einzig mögliche Ableitung für sws. Das vierte Zeichen s des zu L gehörenden Wortes swss kann damit nicht mehr erzeugt werden.
d)	3	Es genügt eine Ergänzung in (R4): $\langle Z \rangle \rightarrow s \langle Z \rangle \mid w \langle Z \rangle \mid s \mid w$
e)	4	Eine Menge von Prozessen ist in einer Verklemmung, wenn jeder Prozess auf ein Ereignis wartet, das nur von einem anderen Prozess dieser Menge ausgelöst werden kann. Zwei Gärtner benötigen zum Pflanzen von Bäumen jeweils einen Spaten zum Ausheben eines Pflanzlochs und eine Schubkarre zum Sammeln und Abtransport des Aushubs. Es steht aber nur ein Spaten und eine Schubkarre zur Verfügung. Der erste Gärtner nimmt sich den Spaten, der andere die Schubkarre. Beide warten nun darauf, dass der jeweils andere sein Arbeitsgerät wieder zurückgibt.

2. a)	3	<pre>int anzahlGeben(int n){ if(n == 0){ return 1; }else{ return 2 * anzahlGeben(n-1); } }</pre>																																													
b)	2	$\begin{aligned} \text{anzahlGeben}(3) &= 2 \cdot \text{anzahlGeben}(2) \\ &= 2 \cdot (2 \cdot \text{anzahlGeben}(1)) \\ &= 2 \cdot (2 \cdot (2 \cdot \text{anzahlGeben}(0))) \\ &= 2 \cdot 2 \cdot 2 \cdot 1 \\ &= 8 \end{aligned}$ Die Methode <i>anzahlGeben</i> wird 4-mal aufgerufen.																																													
c)	3	<table><tr><td>n</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>Gesamtanzahl der Aufrufe</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr></table> Die Gesamtanzahl der Aufrufe bei n ist n + 1, folglich ist der Laufzeit-aufwand linear.	n	0	1	2	3	4	Gesamtanzahl der Aufrufe	1	2	3	4	5																																	
n	0	1	2	3	4																																										
Gesamtanzahl der Aufrufe	1	2	3	4	5																																										
3. a)	5	<table><tr><td>Speicherzelle:</td><td>100</td><td>101</td><td>102</td><td></td></tr><tr><td>Werte:</td><td>undef.</td><td>25</td><td>3</td><td></td></tr><tr><td></td><td>-1</td><td>25</td><td>3</td><td></td></tr><tr><td></td><td>0</td><td>25</td><td>3</td><td></td></tr><tr><td></td><td>0</td><td>8</td><td>3</td><td>100: Ausgabewert des Programms</td></tr><tr><td></td><td>1</td><td>8</td><td>3</td><td></td></tr><tr><td></td><td>1</td><td>2</td><td>3</td><td>101: Eingabewert n</td></tr><tr><td></td><td>2</td><td>2</td><td>3</td><td>102: Eingabewert b</td></tr><tr><td></td><td>2</td><td>0</td><td>3</td><td></td></tr></table>	Speicherzelle:	100	101	102		Werte:	undef.	25	3			-1	25	3			0	25	3			0	8	3	100: Ausgabewert des Programms		1	8	3			1	2	3	101: Eingabewert n		2	2	3	102: Eingabewert b		2	0	3	
Speicherzelle:	100	101	102																																												
Werte:	undef.	25	3																																												
	-1	25	3																																												
	0	25	3																																												
	0	8	3	100: Ausgabewert des Programms																																											
	1	8	3																																												
	1	2	3	101: Eingabewert n																																											
	2	2	3	102: Eingabewert b																																											
	2	0	3																																												

b)	5	<table><tr><td colspan="2">ergebnis = -1</td></tr><tr><td colspan="2">wiederhole solange n-1 größer gleich 0</td></tr><tr><td></td><td>ergebnis = ergebnis +1</td></tr><tr><td></td><td>n = n / b</td></tr></table>	ergebnis = -1		wiederhole solange n-1 größer gleich 0			ergebnis = ergebnis +1		n = n / b
ergebnis = -1										
wiederhole solange n-1 größer gleich 0										
	ergebnis = ergebnis +1									
	n = n / b									
c)	2	Eine mögliche Reihenfolge ist: 5, 2, 6, 1, 7, 4, 3.								
	40									