

Abiturprüfung 2020

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.

Umsetzung der erreichten Bewertungseinheiten in Notenpunkte

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

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

INF1. MODELLIERUNG UND PROGRAMMIERUNG

I.

Aufgabe	BE	Hinweise
1. a)	11	
b)	2	Zur Verwaltung der Spieltermine eignet sich eine einfach verkettete Liste, da die Anzahl der Spieltermine variabel ist. Für die Verwaltung der Kinosäle eignet sich ein Feld, da die Anzahl der Kinosäle feststeht.
c)	7	
2. a)	3	Konzept: Trennung von Struktur und Daten Vorteile: • Wiederverwendbarkeit • leichtere Wartung

b)	5	<pre> classDiagram class BEWERTUNGSLISTE class LISTENELEMENT { <<abstract>> } class ABSCHLUSS class KNOTEN class BEWERTUNG BEWERTUNGSLISTE "1" -- "1" LISTENELEMENT : anfang LISTENELEMENT < -- ABSCHLUSS LISTENELEMENT < -- KNOTEN KNOTEN "1" -- "1" LISTENELEMENT : nachfolger KNOTEN "1" -- "1" KNOTEN : nachfolger KNOTEN "1" -- "1" BEWERTUNG : inhalt </pre> Bei der Implementierung ergibt sich der Vorteil, dass nicht geprüft werden muss, ob ein Anfangs- bzw. Nachfolgerobjekt vorhanden ist.
c)	20	<pre> class BEWERTUNGSLISTE { private LISTENELEMENT anfang; void einfuegen(BEWERTUNG bewertungNeu) { KNOTEN knotenNeu = new KNOTEN(anfang, bewertungNeu); anfang = knotenNeu; } void ausgeben(int sterneMin, int sterneMax) { anfang.ausgeben(sterneMin, sterneMax); } double summeGeben(String filmtitel) { return anfang.summeGeben(filmtitel); } double durchschnittGeben(String filmtitel) { return summeGeben(filmtitel)/anzahlGeben(filmtitel); } // anzahlGeben bereits implementiert } abstract class LISTENELEMENT { abstract void ausgeben(int sterneMin, int sterneMax); abstract double summeGeben(String filmtitel); } class ABSCHLUSS extends LISTENELEMENT { void ausgeben(int sterneMin, int sterneMax) { } double summeGeben(String filmtitel) { return 0.0; } } </pre>

```

class KNOTEN extends LISTENELEMENT {
    private LISTENELEMENT nachfolger;
    private BEWERTUNG inhalt;

    KNOTEN(LISTENELEMENT n, BEWERTUNG b) {
        nachfolger = n;
        inhalt = b;
    }

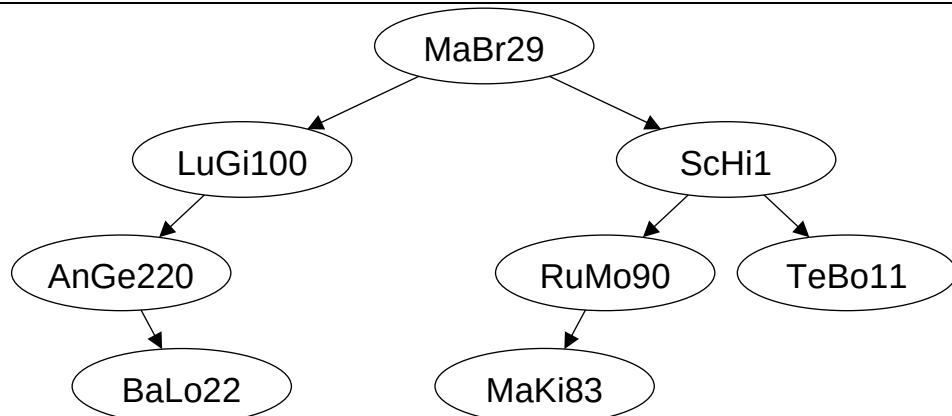
    void ausgeben(int sterneMin, int sterneMax) {
        if (inhalt.anzahlSterneGeben() >= sterneMin &&
            inhalt.anzahlSterneGeben() <= sterneMax) {
            inhalt.ausgeben();
        }
        nachfolger.ausgeben(sterneMin, sterneMax);
    }

    double summeGeben(String filmtitel) {
        if (inhalt.gehoertZuFilm(filmtitel)) {
            return nachfolger.summeGeben(filmtitel) +
                inhalt.anzahlSterneGeben();
        }
        else {
            return nachfolger.summeGeben(filmtitel);
        }
    }
}

```

3. a)

4



Einfügereihenfolge: MaBr29, LuGi100, AnGe220, BaLo22, ScHi1, RuMo90, MaKi83, TeBo11

b)

4

Maximale Knotenanzahl: $2^n - 1$
 $2^9 - 1 - 295 = 216$ Kunden haben noch Platz.
 Term: $2^n - 1 - k$

c)

5

Je nachdem, ob die Kundennummer des einzufügenden Kunden kleiner oder größer als die des verwalteten Kunden ist, wird der linke oder rechte Nachfolger des Knotens mit dem Einfügen beauftragt. Der Knoten erhält eine Referenz zurück, die er als seinen linken bzw. rechten Nachfolger setzt. Er gibt eine Referenz auf sich selbst zurück.

Der Abschluss erzeugt einen neuen Knoten für den einzufügenden Kunden und gibt eine Referenz auf diesen zurück.

4. a)

5

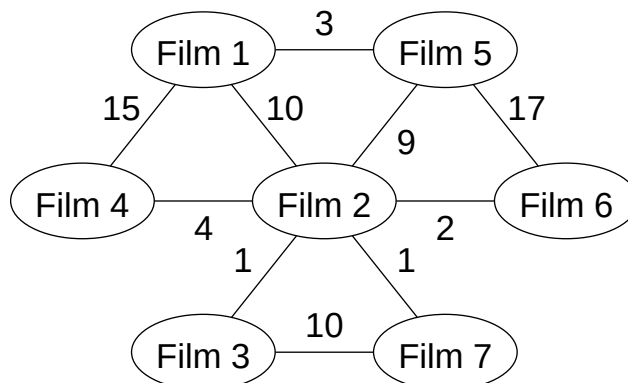
	Film 1	Film 2	Film 3	Film 4	Film 5	Film 6	Film 7
Film 1		10		15			
Film 2	10			4	6	2	
Film 3							9
Film 4	15	4					
Film 5		6				17	
Film 6		2			17		
Film 7			9				

Außerhalb der Diagonalen haben die unbesetzten Stellen sinnvollerweise den Eintrag 0, was dafür steht, dass die betreffenden Filme noch nie beide von demselben Kunden angesehen wurden.

Die Werte auf der Diagonalen haben im Sachzusammenhang keine Bedeutung. Aus programmiertechnischen Gründen eignet sich ebenfalls der Wert 0 (da dann ggf. ein Vergleich der Indizes nicht nötig ist, um den jeweiligen Eintrag zu ignorieren).

b)

4



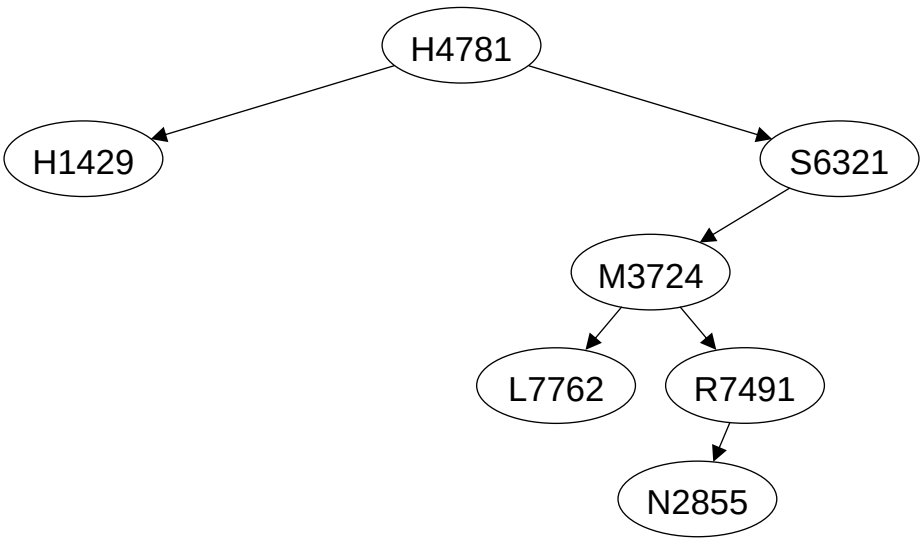
c)	8	Methode besteEmpfehlungSuchen(i) indexBeste = -1 max = 0 zähle j von 0 bis anzahlKnoten-1 wenn matrix[i][j]>max dann max = matrix[i][j] indexBeste = j endeWenn endeZähle wenn indexBeste>=0 dann gib knoten[indexBeste] aus endeWenn endeMethode <i>Hinweis:</i> Es wird hier davon ausgegangen, dass in der Diagonalen der Adjazenzmatrix überall der Wert 0 eingetragen ist.
d)	2	Es wird nicht berücksichtigt, • ob ein Kunde den vorgeschlagenen Film bereits gesehen hat, • ob dem Kunden die gesehenen Filme gefallen haben, • wie lange ein Film bereits läuft.
	80	

INF1. MODELLIERUNG UND PROGRAMMIERUNG

II.

Aufgabe	BE	Hinweise
1.	4	Nennung und Beschreibung von vier der folgenden fünf Phasen: Analyse: Festlegung der Anforderungen an die Software unter Einbeziehung des Kunden Entwurf: Modellierung der Komponenten der Software Implementierung: Erstellen von Programmcode Gesamttest: Zusammenführung der Einzelkomponenten und Test der gesamten Software Wartung: Anpassung der Software an neue Bedingungen
2. a)	6	<pre> classDiagram class SPIELEVERLEIH class SPIEL { name neupreis } class KUNDE class AUSLEIHVORGANG { ausleihdatum } class EXEMPLAR { nummer abnutzungsgrad } SPIELEVERLEIH --> SPIEL : verwaltet > n SPIELEVERLEIH --> KUNDE : verwaltet v n KUNDE --> AUSLEIHVORGANG : tätig v n AUSLEIHVORGANG --> EXEMPLAR : betrifft > n 1 SPIEL --> EXEMPLAR : ist_von ^ 1 n </pre>
b)	4	<pre> classDiagram class AUSLEIHLISTE class LISTENELEMENT { <abstract> } class ABSCHLUSS class KNOTEN class AUSLEIHVORGANG AUSLEIHLISTE --> LISTENELEMENT : erster > 1 LISTENELEMENT < -- ABSCHLUSS LISTENELEMENT < -- KNOTEN KNOTEN --> LISTENELEMENT : < nachfolger 1 KNOTEN --> AUSLEIHVORGANG : inhalt v 1 </pre>

c)	7	<pre> classDiagram class liste1["liste1: AUSLEIHLISTE"] class k1["k1: KNOTEN"] class k2["k2: KNOTEN"] class a["a: ABSCHLUSS"] class v1["v1: AUSLEIHVORGANG
ausleihdatum = 03.04.2020"] class v2["v2: AUSLEIHVORGANG
ausleihdatum = 03.04.2020"] class e1["e1: EXEMPLAR
abnutzungsgrad = 1"] class e2["e2: EXEMPLAR
abnutzungsgrad = 4"] class s1["s1: SPIEL
name = „Boccia“
neupreis = 35 €"] liste1 --> k1 : erster k1 --> k2 : nachfolger k1 --> v1 : inhalt k2 --> v2 : inhalt v1 --> e1 : betrifft v2 --> e2 : betrifft e1 --> s1 : ist_von e2 --> s1 : ist_von </pre>
d)	5	<pre> sequenceDiagram participant v2 as v2: AUSLEIHVORGANG participant e2 as e2: EXEMPLAR participant s1 as s1: SPIEL v2->>v2: gebuehrBerechnen() activate v2 v2->>e2: restwertBerechnen() activate e2 e2->>s1: neupreisGeben() activate s1 s1-->>e2: 35 € deactivate s1 e2-->>v2: 14 € deactivate e2 v2-->>: 3,50 € deactivate v2 </pre>
e)	7	in der Klasse AUSLEIHLISTE: <pre> void entfernen(Exemplar exemplar) { erster = erster.entfernen(exemplar); } </pre> in der abstrakten Klasse LISTENELEMENT: <pre> abstract Listenelement entfernen(Exemplar exemplar); </pre> in der Klasse KNOTEN: <pre> Listenelement entfernen(Exemplar exemplar) { if (exemplar==inhalt.exemplarGeben()) { return nachfolger; } else { nachfolger = nachfolger.entfernen(exemplar); return this; } } </pre> in der Klasse ABSCHLUSS: <pre> Listenelement entfernen(Exemplar exemplar) { return this; } </pre>

3. a)	5	H4781, H1429, S6321, M3724 H4781, S6321, H1429, M3724 H4781, S6321, M3724, H1429  <pre> graph TD H4781((H4781)) --> H1429((H1429)) H4781 --> S6321((S6321)) S6321 --> M3724((M3724)) M3724 --> L7762((L7762)) M3724 --> R7491((R7491)) R7491 --> N2855((N2855)) </pre>
b)	4	Inorder-Durchlauf: Von jedem Knoten wird zuerst rekursiv der linke Teilbaum abgearbeitet, dann der Inhalt dieses Knotens ausgegeben und anschließend rekursiv der rechte Teilbaum abgearbeitet. Eine nach den Nachnamen sortierte Ausgabe kann damit im Allgemeinen nicht erzeugt werden, da die vierstellige Zahl unabhängig von der alphabetischen Ordnung sein kann.
c)	2	Für die Anzahl n der Ebenen muss gelten: $2^n - 1 \geq 2500$; also mindestens 12 Ebenen.
d)	3	Vorteil: Bei einem Suchvorgang sind im Mittel weniger Vergleiche notwendig. Nachteil: Beim Einfügen eines neuen Knotens muss der Baum unter Umständen aufwendig reorganisiert werden.
e)	2	Anstelle der Beziehung <i>nachfolger</i> existieren zwei Beziehungen <i>nachfolgerLinks</i> und <i>nachfolgerRechts</i>.

f)	10	in der Klasse KUNDENBAUM: <pre>boolean passwortPruefen(String kdnr, String pw) { return wurzel.passwortPruefen(kdnr, pw); }</pre> in der abstrakten Klasse BAUMELEMENT: <pre>abstract boolean passwortPruefen(String kdnr, String pw);</pre> in der Klasse KNOTEN: <pre>boolean passwortPruefen(String kdnr, String pw) { int vgl = inhalt.kundennummerVergleichen(kdnr); if (vgl==0) { return inhalt.istPasswortGleich(pw); } else if (vgl<0) { return nachfolgerLinks.passwortPruefen(kdnr, pw); } else { return nachfolgerRechts.passwortPruefen(kdnr, pw); } }</pre> in der Klasse ABSCHLUSS: <pre>boolean passwortPruefen(String kdnr, String pw) { return false; }</pre>																																																																
4. a)	6	<table><tr><td></td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td><td>G</td></tr><tr><td>A</td><td></td><td>6</td><td></td><td></td><td></td><td>4</td><td>5</td></tr><tr><td>B</td><td>6</td><td></td><td>3</td><td></td><td></td><td></td><td></td></tr><tr><td>C</td><td></td><td>3</td><td></td><td>4</td><td></td><td></td><td>4</td></tr><tr><td>D</td><td></td><td></td><td>4</td><td></td><td>5</td><td></td><td></td></tr><tr><td>E</td><td></td><td></td><td></td><td>5</td><td></td><td>3</td><td>6</td></tr><tr><td>F</td><td>4</td><td></td><td></td><td></td><td>3</td><td></td><td></td></tr><tr><td>G</td><td>5</td><td></td><td>4</td><td></td><td>6</td><td></td><td></td></tr></table> zwei der folgenden Eigenschaften: gewichtet, ungerichtet, zusammenhängend, zyklisch		A	B	C	D	E	F	G	A		6				4	5	B	6		3					C		3		4			4	D			4		5			E				5		3	6	F	4				3			G	5		4		6		
	A	B	C	D	E	F	G																																																											
A		6				4	5																																																											
B	6		3																																																															
C		3		4			4																																																											
D			4		5																																																													
E				5		3	6																																																											
F	4				3																																																													
G	5		4		6																																																													
b)	5	<table><tr><th>Spielzug</th><th>Rest</th></tr><tr><td>(A</td><td>12</td></tr><tr><td>A – B</td><td>6</td></tr><tr><td>A – B – C</td><td>3</td></tr><tr><td>A – F</td><td>8</td></tr><tr><td>A – F – E</td><td>5</td></tr><tr><td>A – F – E – D</td><td>0</td></tr><tr><td>A – G</td><td>7</td></tr><tr><td>A – G – C</td><td>3</td></tr><tr><td>A – G – C – B</td><td>0</td></tr><tr><td>A – G – E</td><td>1</td></tr></table>) je nachdem, ob keine Bewegung der Spielfigur als Zug interpretiert wird	Spielzug	Rest	(A	12	A – B	6	A – B – C	3	A – F	8	A – F – E	5	A – F – E – D	0	A – G	7	A – G – C	3	A – G – C – B	0	A – G – E	1																																										
Spielzug	Rest																																																																	
(A	12																																																																	
A – B	6																																																																	
A – B – C	3																																																																	
A – F	8																																																																	
A – F – E	5																																																																	
A – F – E – D	0																																																																	
A – G	7																																																																	
A – G – C	3																																																																	
A – G – C – B	0																																																																	
A – G – E	1																																																																	

c)	10	Methode spielzug(indexStart, punktestand) zähle i von 0 bis anzahlKnoten-1 knoten[i].besuchtSetzen(falsch) endeZähle pfadeAusgeben(indexStart, knoten[indexStart].bezeichnungGeben(), punktestand) endeMethode Methode pfadeAusgeben(index, pfad, punktestand) gib (pfad+", "+punktestand) aus knoten[index].besuchtSetzen(wahr) zähle i von 0 bis anzahlKnoten-1 wenn matrix[index][i]>0 und nicht knoten[i].besuchtGeben() und matrix[index][i]<=punktestand dann pfadNeu = pfad+" – "+knoten[i].bezeichnungGeben() punktestandNeu = punktestand-matrix[index][i] pfadeAusgeben(i, pfadNeu, punktestandNeu) endeWenn endeZähle knoten[index].besuchtSetzen(falsch) endeMethode
	80	

III.

Aufgabe	BE	Hinweise
1. a)	5	$\langle \text{Raum} \rangle \rightarrow \langle \text{Gebäude} \rangle - \langle \text{Nummer} \rangle \rightarrow A - \langle \text{Nummer} \rangle \rightarrow$ $A - \langle Z \rangle \langle \text{Nummer} \rangle \rightarrow A - \langle Z \rangle \langle Z \rangle \langle \text{Nummer} \rangle \rightarrow$ $A - \langle Z \rangle \langle Z \rangle \langle Z \rangle \langle Z^n N \rangle \rightarrow A - 0 \langle Z \rangle \langle Z \rangle \langle Z^n N \rangle \rightarrow$ $A - 00 \langle Z \rangle \langle Z^n N \rangle \rightarrow A - 000 \langle Z^n N \rangle \rightarrow A - 0007$ $A - 100$ wird nicht von G erzeugt, gehört aber zu L.
b)	6	<pre> graph LR start(()) --> z0((z0)) z0 -- "A, B, C" --> z1((z1)) z1 -- "-" --> z2((z2)) z2 -- "0" --> z3((z3)) z3 -- "0" --> z4((z4)) z2 -- "1, ..., 9" --> z5((z5)) z3 -- "1, ..., 9" --> z6(((z6))) z4 -- "1, ..., 9" --> z7(((z7))) z5 -- "0, ..., 9" --> z6 z6 -- "0, ..., 9" --> z7 style start fill:none,stroke:none style z6 fill:none,stroke:double style z7 fill:none,stroke:double </pre>
c)	2	Es kann syntaktisch korrekte Raumbezeichnungen geben, zu denen kein Raum existiert und die damit semantisch nicht sinnvoll sind.
2. a)	4	Zwei Lehrkräfte wollen für die dritte Stunde am Dienstag Koffer 3 buchen. Wenn die beiden Buchungsprozesse nebenläufig ausgeführt werden, kann es vorkommen, dass Prozess 1 nach der (zweiten) Feststellung der Verfügbarkeit unterbrochen und erst wieder fortgesetzt wird, nachdem Prozess 2 bereits die Buchungsbestätigung erhalten hat. Prozess 1 erhält im Anschluss ebenfalls eine Buchungsbestätigung.
b)	3	Kritischer Abschnitt: (zweite) Feststellung der Verfügbarkeit und Vornehmen der Buchung Monitorkonzept: Sobald ein Prozess eine als geschützt markierte Methode eines Objekts ausführt, sorgt der Monitor des Objekts dafür, dass es als belegt markiert wird. Weitere Prozesse, die ebenfalls eine solche Methode des Objekts ausführen möchten, müssen in einer Warteschlange warten.

c)	5	<i>Beispiel:</i> Ausgangssituation: Die Lehrkräfte L1 und L2 haben jeweils einen Koffer für die kommende Stunde reserviert; genau diese Koffer sind derzeit von den Lehrkräften L3 und L4 ausgeliehen. Die Lehrkräfte L1 und L2 erreichen den Hausmeister vor den Lehrkräften L3 und L4, die die entsprechenden Koffer zurückbringen wollen. L1 und L2 erhalten jeweils einen Schlüssel und gehen zum Medienraum. Dort warten sie, weil die reservierten Koffer nicht vorhanden sind. L3 und L4 warten beim Hausmeister auf einen Schlüssel. Da die Gruppen L1/L2 und L3/L4 gegenseitig aufeinander warten, tritt eine Verklemmung auf. Veränderung: Falls der reservierte Koffer nicht vorhanden ist, nicht warten, sondern trotzdem (S4) und (S5) ausführen und wieder bei (S1) beginnen.						
3. a)	8	<pre>1: load 202 2: muli 5 3: subi 500 4: store 203 5: subi 1000 6: jle 10 7: loadi 2 8: store 201 9: jmp 1 10: load 203 11: subi 500 12: jle 16 13: loadi 1 14: store 201 15: jmp 1 16: loadi 0 17: store 201 18: jmp 1</pre>						
b)	4	Startwert	1	2	3	10	100	n
		Befehle	6	11	16	51	501	5n+1
c)	3	$\frac{2s}{10^{-7} s} = 20\,000\,000$ Der Startwert n sollte etwa 4 000 000 sein.						
	40							

IV.

Aufgabe	BE	Hinweise
1. a)	4	Alphabet: $\Sigma = \{A, B, C, D, 0, 1, -\}$ EBNF: Produktcode = Präfix Ziffernfolge; Präfix = 'A' 'B' 'C' 'D'; Ziffernfolge = '0' Z0 '-' '0' '0' Z1 '-' '1'; Z0 = '0' [Z0] '1' Z1; Z1 = '1' [Z0] '0' Z1;
b)	4	„C01011–1“ ist der korrekt gebildete Produktcode. <Produktcode> $\rightarrow$ <Präfix> <Ziffernfolge> $\rightarrow$ C <Ziffernfolge> $\rightarrow$ C0 <Z1> –1 $\rightarrow$ C01 <Z0> –1 $\rightarrow$ C010 <Z0> –1 $\rightarrow$ C0101 <Z1> –1 $\rightarrow$ C01011–1 Die Anzahl der Einsen ist in jedem Fall gerade.
c)	5	
2. a)	4	

b)	5	Die Methoden <i>anzahlGeben</i> und <i>anzahlSetzen</i> müssen durch eine einzige als geschützt markierte Methode zum Erhöhen des Zählers ersetzt werden. Diese kann dann immer nur von einem Prozess zur selben Zeit ausgeführt werden. Zu einer Verklemmung kann es in der beschriebenen Situation nicht kommen, da es nur eine gemeinsam genutzte Ressource gibt und daher jeder Prozess während der Ausführung der Methode <i>abrufeErhöhen</i> auf keine weiteren Ressourcen warten muss, sodass die Ausführung auf jeden Fall zu einem Ende kommt. <i>Hinweis:</i> Die Synchronisation der Methode <i>abrufeErhöhen</i> bei den einzelnen Prozessen genügt nicht, da das zentrale Objekt <i>zähler</i> dadurch nicht geschützt wäre.						
c)	3	Der Zentralserver kann maximal $100 \cdot 60 \cdot 60 \cdot 24 = 8640000$ Meldungen pro Tag verarbeiten. Einzelmeldung: Zu viele eintreffende Meldungen können den Zentralserver überlasten. Sammelmeldung: Der Zählerstand ist ggf. nicht aktuell.						
3. a)	3	<table border="1"><tr><td colspan="2" style="text-align: center;">$x < 0$</td></tr><tr><td style="text-align: center;">wahr</td><td style="text-align: center;">falsch</td></tr><tr><td style="text-align: center;">$y = -x$</td><td style="text-align: center;">$y = x$</td></tr></table> Bedeutung von y: Betrag der Zahl x	$x < 0$		wahr	falsch	$y = -x$	$y = x$
$x < 0$								
wahr	falsch							
$y = -x$	$y = x$							
b)	4	Folge der rekursiven Methodenaufrufe: $\text{potenz}(2, 5) \rightarrow \text{potenz}(2, 4) \cdot 2 \rightarrow \text{potenz}(4, 2) \cdot 2 \rightarrow$ $\text{potenz}(16, 1) \cdot 2 \rightarrow \text{potenz}(16, 0) \cdot 32 \rightarrow 32$ Bei Verdoppelung des positiven Exponenten erhöht sich die Zahl der rekursiven Aufrufe um 1.						

c)	8	Die Adressen der Speicherzellen, in denen Basis, Exponent und Ergebnis abgelegt sind, werden mit den Symbolen <i>a</i>, <i>n</i> bzw. <i>erg</i> bezeichnet. <pre> loadi 1 store erg -- Ergebnis mit 1 vorbelegen start: load n -- Exponent untersuchen jmpz ende -- Abbruchbedingung modi 2 -- Exponent gerade? jmpnz sonst load n -- Exponent halbieren, wenn gerade divi 2 store n load a -- Basis quadrieren, wenn Exponent gerade mul a store a jmp start sonst: load n -- Exponent dekrementieren, wenn ungerade subi 1 store n load erg -- Ergebnis mit Basis multiplizieren, mul a -- wenn Exponent ungerade store erg jmp start ende: hold </pre>
	40	