

Abiturprüfung 2014

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*.

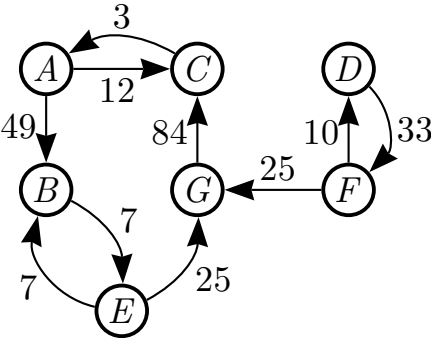
Die in den Lösungshinweisen gewählte objektorientierte Programmiersprache ist Java.

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. a)	6	 Die Pfeilrichtung $A \rightarrow C$ entspricht hierbei der Beziehung „A besitzt von C ... Prozent der Anteile“. Ein ungerichteter Graph würde eine gegenseitige Abhängigkeit der Firmen prinzipiell zu gleichen Anteilen unterstellen.																																																																
b)	4	<table><tr><th></th><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th><th>F</th><th>G</th></tr><tr><th>A</th><td></td><td>49</td><td>12</td><td></td><td></td><td></td><td></td></tr><tr><th>B</th><td></td><td></td><td></td><td></td><td>7</td><td></td><td></td></tr><tr><th>C</th><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><th>D</th><td></td><td></td><td></td><td></td><td></td><td>33</td><td></td></tr><tr><th>E</th><td></td><td>7</td><td></td><td></td><td></td><td></td><td>25</td></tr><tr><th>F</th><td></td><td></td><td></td><td>10</td><td></td><td></td><td>25</td></tr><tr><th>G</th><td></td><td></td><td>84</td><td></td><td></td><td></td><td></td></tr></table>		A	B	C	D	E	F	G	A		49	12					B					7			C	3							D						33		E		7					25	F				10			25	G			84				
	A	B	C	D	E	F	G																																																											
A		49	12																																																															
B					7																																																													
C	3																																																																	
D						33																																																												
E		7					25																																																											
F				10			25																																																											
G			84																																																															
c)	10	Methode tiefsucheStarten(startKnotenNr) wiederhole für alle Knoten setze Markierung des Knotens auf den Wert für unbesucht endeWiederhole tiefsucheDurchfuehren(startKnotenNr) endeMethode Methode tiefsucheDurchfuehren(knotenNr) markiere den Knoten mit der Nummer knotenNr als besucht wiederhole von $j = 0$ bis $j = \text{anzahlKnoten} - 1$ wenn $\text{adjazenzmatrix}[\text{knotenNr}][j] > 0$ und der Knoten mit der Nummer j wurde noch nicht besucht dann tiefsucheDurchfuehren(j) endeWenn endeWiederhole endeMethode Die Knoten werden in folgender Reihenfolge besucht: $A - B - E - G - C$. In diesem Zusammenhang bedeutet dies, dass Firma A direkt oder indirekt an den Firmen B, C, E und G beteiligt ist.																																																																

2. a)	4	<pre> graph TD CH325(CH 325) --> AT74(AT 74) CH325 --> RJ86(RJ 86) AT74 --> BU290(BU 290) RJ86 --> MV255(MV 255) RJ86 --> SV131(SV 131) SV131 --> SB11(SB 11) SB11 --> SK905(SK 905) </pre>
b)	3	Postorder: BU 290, AT 74, MV 255, SK 905, SB 11, SV 131, RJ 86, CH 325
c)	2	In diesem Fall muss nur die Referenz, die von Knoten RJ 86 auf Knoten SV 131 führt, auf den Knoten SB 11 „umgelenkt“ werden.
d)	6	Der geordnete Binärbaum hat möglichst wenig Ebenen. Die Anzahl der Knoten in einem solchen Baum mit n Ebenen ist ungefähr 2^n . Wegen $2^{22} < 7\,000\,000 < 2^{23}$ genügen 23 Ebenen, um sämtliche Knoten unterzubringen.
3. a)	16	Mögliches Klassendiagramm: <pre> classDiagram class INTERNET_SCHNITTSTELLE class ZENTRALE { +taxiHinzufuegen(kennzeichen) +fahrerHinzufuegen(persnr, name) +aktuellePosZeigen(taxi) +karteHolen(pos) } class TAXI { +kennzeichen +datenZeigen() +positionSenden() } class FAHRT { +datum +startzeit +ankunftszeit +startort +zielort +datenZeigen() } class FAHRER { +persnr +name +datenZeigen() +fahrtStarten() +fahrtBeenden() } INTERNET_SCHNITTSTELLE "1" -- "1" ZENTRALE : < verbunden mit ZENTRALE "1" -- "n" TAXI : < verwaltet ZENTRALE "1" -- "n" FAHRT : verwaltet ZENTRALE "1" -- "n" FAHRER : verwaltet TAXI "1" -- "n" FAHRT : < nutzt FAHRT "n" -- "1" FAHRER : < macht </pre>
b)	4	Grundlegende Idee: Der Übergang zur nächsten Phase findet erst statt, wenn die aktuelle Phase abgeschlossen ist. Typische Phasen sind: Analyse, Entwurf, Implementierung, Test, Bewertung und Abnahme.

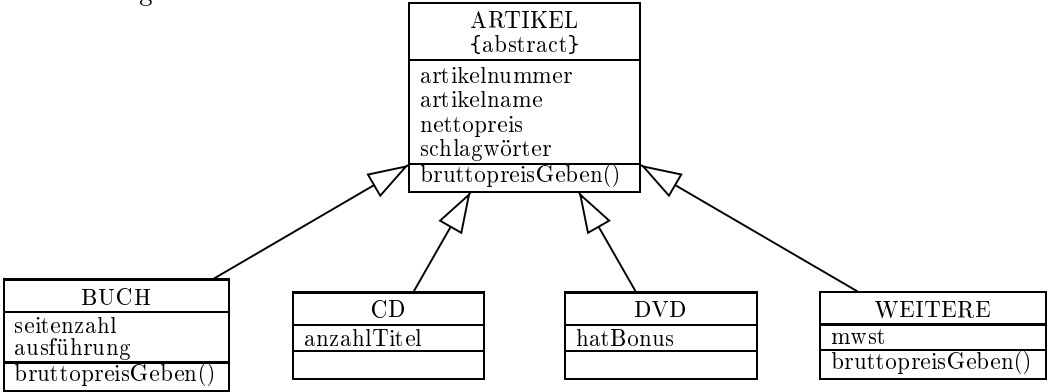
4. a)	5	<pre> graph TD m1["m1 : MESSREIHE"] -- erster --> k1["k1 : KNOTEN"] k1 -- nächster --> k2["k2 : KNOTEN"] k2 -- nächster --> k3["k3 : KNOTEN"] k3 -- nächster --> a1["a1 : ABSCHLUSS"] k1 -- inhalt --> z1["z1 : ZERFALL"] k2 -- inhalt --> z2["z2 : ZERFALL"] k3 -- inhalt --> z3["z3 : ZERFALL"] </pre>
b)	20	<pre> class Messreihe{ private Listenelement erster; Messreihe(){ erster = new Abschluss(); } double maxEnergieGeben(){ return erster.maxEnergieGeben(0.0); } double mittlereZerfallszeitGeben(){ return erster.summeZeitenGeben() / erster.anzahlKnotenGeben(); //Das Abfangen einer Division durch Null wird hier nicht verlangt... } } abstract class Listenelement{ abstract double maxEnergieGeben(double temp); abstract double summeZeitenGeben(); abstract int anzahlKnotenGeben(); } class Abschluss extends Listenelement{ double maxEnergieGeben(double temp){ return temp; } double summeZeitenGeben(){ return 0; } int anzahlKnotenGeben(){ return 0; } } </pre>

```
class Datenknoten extends Listenelement{
    private Listenelement naechster;
    private Zerfall inhalt;
    Datenknoten(Listenelement le, Zerfall z){
        naechster = le;
        inhalt = z;
    }
    double maxEnergieGeben(double temp){
        if(temp < inhalt.energieGeben()){
            return naechster.maxEnergieGeben(inhalt.energieGeben());
        }
        else{
            return naechster.maxEnergieGeben(temp);
        }
    }
    double summeZeitenGeben(){
        return inhalt.zerfallszeitGeben() + naechster.summeZeitenGeben();
    }
    int anzahlKnotenGeben(){
        return 1 + naechster.anzahlKnotenGeben();
    }
}

class Zerfall{
    private double zerfallszeit;
    private double energie;
    private int anzahlGammaquanten;
    Zerfall(double z, double e, int aq){
        zerfallszeit = z;
        energie = e;
        anzahlGammaquanten = aq;
    }
    double zerfallszeitGeben(){
        return zerfallszeit;
    }
    double energieGeben(){
        return energie;
    }
}
```

Aufgabe	BE	Hinweise
1. a)	5	<pre> sequenceDiagram actor neukunde participant verwaltung as verwaltung:KUNDENVERWALTUNG neukunde->>verwaltung: registrieren(name, adresse, password) activate verwaltung verwaltung->>verwaltung: kundennummerErzeugen() verwaltung-->>verwaltung: nummer deactivate verwaltung verwaltung->>verwaltung: new(nummer, password, name, adresse) verwaltung-->>verwaltung: kunde:KUNDE deactivate verwaltung verwaltung-->>neukunde: neuer Kunde deactivate verwaltung </pre>
b)	8	<pre> classDiagram class KUNDE { kundennummer password name adresse bestellen() } class KUNDENVERWALTUNG { kundennummerErzeugen() } class WARENKORB { einfügen(eintrag) entfernen(eintrag) } class WARENKORBEINTRAG { anzahl anzahlSetzen(anzahlNeu) } class ARTIKEL { artikelnummer } KUNDE "1" --> "n" KUNDENVERWALTUNG : verwaltet KUNDE "1" -- "1" WARENKORB : hat WARENKORB "1" --> "n" WARENKORBEINTRAG : verwaltet WARENKORBEINTRAG "n" --> "n" ARTIKEL : bezieht sich auf </pre>
c)	8	<pre> graph LR Start((Startseite)) -- "Registrieren" --> Start Start -- "Anmelden" --> WA(Warenauswahl) WA -- "Neuer Artikel" --> WA WA -- "Artikel löschen" --> WA WA -- "Anzahl ändern" --> WA WA -- "Bestellung abschließen" --> ZZW(Zahlungsart wählen) ZZW -- "Zahlungsart festlegen" --> UDD(Überprüfung der Daten) UDD -- "Endgültig bestätigen" --> Fertig((Fertig)) </pre>

d)	12	In der Klasse WARENKORB: <pre>float gesamtbruttopreisGeben() { return anfang.bruttopreisAbHierGeben(); }</pre> Warenkorbeintrag artikelSuchen(int nummer) { <pre> return anfang.artikelSuchen(nummer); }</pre> In der Klasse LISTENELEMENT: <pre>abstract float bruttopreisAbHierGeben(); abstract Warenkorbeintrag artikelSuchen(int nummer);</pre> In der Klasse ABSCHLUSS: <pre>float bruttopreisAbHierGeben() { return 0.0f; }</pre> Warenkorbeintrag artikelSuchen(int nummer) { <pre> return null; }</pre> In der Klasse KNOTEN: <pre>float bruttopreisAbHierGeben() { return nachfolger.bruttopreisAbHierGeben() + daten.artikelGeben().bruttopreisGeben() * daten.anzahlGeben(); }</pre> Warenkorbeintrag artikelSuchen(int nummer) { <pre> if (daten.artikelGeben().artikelnummerGeben() == nummer) { return daten; } else { return nachfolger.artikelSuchen(nummer); } }</pre>
----	----	--

2. a)	7	Alle Artikel haben mehrere Attribute und Methoden gemeinsam, sind aber spezialisierte Varianten der Klasse ARTIKEL, da sie zusätzliche Attribute benötigen und die Ausführung einzelner Methoden modifizieren müssen. Mögliches Klassendiagramm:  <pre> classDiagram class ARTIKEL { <<abstract>> artikelnummer artikelname nettopreis schlagwörter bruttopreisGeben() } class BUCH { seitenzahl ausführung bruttopreisGeben() } class CD { anzahlTitel } class DVD { hatBonus } class WEITERE { mwst bruttopreisGeben() } ARTIKEL < -- BUCH ARTIKEL < -- CD ARTIKEL < -- DVD ARTIKEL < -- WEITERE </pre> Da der Bruttopreis für CDs und DVDs gleich berechnet wird, ist die Methode <i>bruttopreisGeben</i> primär in der Klasse ARTIKEL implementiert. In den Klassen BUCH und WEITERE wird sie überschrieben, um den dort geltenden Bedingungen gerecht zu werden.
b)	5	In der Klasse ARTIKEL: <pre>float bruttopreisGeben() { return nettopreis * 1.19f; }</pre> In der Klasse BUCH: <pre>float bruttopreisGeben() { return nettopreis * 1.07f; }</pre> In der Klasse WEITERE: <pre>float bruttopreisGeben() { return nettopreis * (1.0f + mwst / 100.0f); }</pre>

3. a)	8	Ein mögliches Modell ist das Wasserfallmodell. Seine typischen Phasen sind: Anforderungsanalyse: Hier werden die verlangten Leistungen festgestellt (Lastenheft) und es wird festgelegt, wie diese Leistungen erbracht werden (Pflichtenheft). Systementwurf: Das zu erstellende System wird modelliert. Implementierung: Das im Systementwurf erstellte Modell wird am Computer realisiert. Dabei werden die einzelnen Teile getestet. Zusammenführung und Abnahme: Die einzelnen Teile werden zusammengeführt. Nach dem Gesamttest wird überprüft, ob das System alle Anforderungen des Pflichtenhefts erfüllt (Abnahme). Installation und Wartung: Das fertige System wird in Betrieb genommen. In der Folge werden eventuell übersehene Fehler beseitigt; das System wird gegebenenfalls an veränderte Anforderungen angepasst (Wartung). <i>Hinweis:</i> Die Wartung gehört nicht mehr zur eigentlichen Erstellung und muss daher nicht aufgeführt sein. Meilensteine legen fest, zu welchem Zeitpunkt eines Projekts bestimmte Teile erstellt sein müssen. Typische Meilensteine des Wasserfallmodells sind die Übergänge der Phasen, also z. B. Fertigstellung des Systementwurfs.
b)	4	Kritische Abschnitte entstehen durch die Bearbeitung der Klasse GESPRAECHSVERWALTUNG, da diese Ressource während der Änderung durch ein Team nicht auch durch andere Teams verändert werden darf. Ein mögliches Verfahren zur Koordination sind Semaphore, d. h. Merker, die zu Beginn einer Änderung auf den Wert „belegt“ und nach der Änderung wieder auf „frei“ gesetzt werden. Andere Teams können erst wieder zugreifen, wenn der Semaphor auf „frei“ steht.
4. a)	8	Da der Baum nach Artikelnummern sortiert ist, liefert ein Durchlauf nach Inorder-Strategie die Artikel in der gewünschten Reihenfolge. Die bedingte Anweisung stellt sicher, dass nur Artikel ausgegeben werden, die die Suchbedingung erfüllen. In der Klasse ARTIKELBAUM: <pre>void artikelAuflisten(String schlagwort){ anfang.artikelAuflisten(schlagwort); }</pre> In der Klasse BAUMELEMENT: <pre>abstract void artikelAuflisten(String schlagwort);</pre> In der Klasse ABSCHLUSS: <pre>void artikelAuflisten(String schlagwort){ // nichts zu tun }</pre> In der Klasse KNOTEN: <pre>void artikelAuflisten(String schlagwort){ linksNachfolger.artikelAuflisten(schlagwort); if (daten.istSchlagwortVorhanden(schlagwort)){ daten.nameAusgeben(); } rechtsNachfolger.artikelAuflisten(schlagwort); }</pre>
b)	2	<pre>SELECT * FROM artikel WHERE artikelnummer=123456</pre>

5. a)	3	<table><tr><td></td><td>L</td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td></tr><tr><td>L</td><td></td><td>40</td><td>50</td><td>30</td><td></td><td></td></tr><tr><td>A</td><td>40</td><td></td><td>70</td><td></td><td>60</td><td></td></tr><tr><td>B</td><td>50</td><td>70</td><td></td><td>50</td><td></td><td></td></tr><tr><td>C</td><td>30</td><td></td><td>50</td><td></td><td>30</td><td>20</td></tr><tr><td>D</td><td></td><td>60</td><td></td><td>30</td><td></td><td></td></tr><tr><td>E</td><td></td><td></td><td></td><td>20</td><td></td><td></td></tr></table>		L	A	B	C	D	E	L		40	50	30			A	40		70		60		B	50	70		50			C	30		50		30	20	D		60		30			E				20		
	L	A	B	C	D	E																																													
L		40	50	30																																															
A	40		70		60																																														
B	50	70		50																																															
C	30		50		30	20																																													
D		60		30																																															
E				20																																															
b)	4	<pre>class Graph { int [][] matrix; Knoten [] knoten; Graph(){ matrix = new int [6][6]; knoten = new Knoten [6]; } }</pre>																																																	
c)	6	Ein möglicher Algorithmus ist die Tiefensuche. Er benötigt als weiteres Attribut z. B. ein Feld von Wahrheitswerten, das für jeden Knoten angibt, ob er schon besucht wurde oder nicht. Jedes Feldelement ist zu Beginn mit <i>falsch</i> vorbesetzt. Für den Aufruf mit einem aktuellen Knoten k wird zuerst dieser auf „besucht“ (<i>wahr</i>) gesetzt. Anschließend wird die zugeordnete Zeile der Adjazenzmatrix durchgegangen; für jeden noch nicht besuchten Knoten wird die gleiche Methode rekursiv aufgerufen. Der erste Aufruf ruft die Methode mit dem Startknoten als Parameter auf.																																																	
	80																																																		

Aufgabe	BE	Hinweise
1. a)	2	$\langle S \rangle \rightarrow \langle \text{Paar} \rangle \langle \text{Paar} \rangle \langle \text{Paar} \rangle \langle \text{Schwarz} \rangle$ $\rightarrow \langle \text{Paar} \rangle \langle \text{Paar} \rangle \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle$ $\rightarrow \langle \text{Paar} \rangle \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle$ $\rightarrow \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle$ $\rightarrow \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle s$ $\rightarrow \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle w w w s$ $\rightarrow \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle s w w w s$ $\rightarrow \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle \langle \text{Schwarz} \rangle w w s w w w s$ $\rightarrow \langle \text{Schwarz} \rangle \langle \text{Weiß} \rangle s s s w w s w w w s$ $\rightarrow \langle \text{Schwarz} \rangle w s s s w w s w w w s$ $\rightarrow s s w s s s w w s w w w s$ <i>Hinweis:</i> Sinnvolle Zusammenfassungen der Ableitungsschritte sind zulässig.
b)	4	Mögliche Argumente: Da die Grammatik keine rekursiven Produktionsregeln enthält, sind nur Worte begrenzter Länge möglich. Die erste Produktionsregel erzeugt mindestens ein Schwarz-Weiß-Paar, gefolgt von einem schwarzen Streifen. Ein einzelner schwarzer Streifen ist also nicht möglich. Die folgende Veränderung lässt beliebig lange Strichcodes zu, im Minimalfall auch den aus nur einem einzigen schwarzen Streifen bestehenden: $\langle S \rangle \rightarrow \langle \text{Schwarz} \rangle \mid \langle \text{Paar} \rangle \langle S \rangle$
c)	6	
2. a)	3	Zwei Mitarbeiter können kurz nacheinander einen Anruf entgegennehmen, so dass der zweite mit der Arbeit beginnt, bevor der erste fertig ist. Die Arbeitsvorgänge der beiden Mitarbeiter stellen nebenläufige Prozesse dar. Z. B.: Der auf der Preistafel angezeigte Preis stellt ein gemeinsam genutztes Betriebsmittel dar.

b)	5	Während Mitarbeiter X eine Preiserhöhung um 3 Cent bearbeitet, schreibt Mitarbeiter Y noch den alten Preis ab und berechnet davon ausgehend eine Preissenkung um 2 Cent. Die vorhergehende Preiserhöhung ist somit unwirksam. Der zuletzt aufgehängte Preis ist damit falsch. <pre> sequenceDiagram participant X as Mitarbeiter X participant PT as Preistafel participant Y as Mitarbeiter Y X->>PT: ändern(+3) X->>PT: preisLesen() PT-->>X: X->>X: berechnen() X->>PT: aushängen() Y->>PT: ändern(-2) Y->>PT: preisLesen() PT-->>Y: Y->>Y: berechnen() Y->>PT: aushängen() PT-->>Y: </pre>
c)	4	Ein geeignetes informatisches Konzept wäre z. B. der Monitor, in dem eine Menge von Attributen und Methoden (hier wären das die Preistafel und die Methode <i>ändern</i>) durch einen Überwachungsmechanismus so geschützt sind, dass keine zwei Prozesse gleichzeitig auf diese Menge zugreifen können. Im Beispiel könnte der Überwachungsmechanismus ein Chef sein, dem Bearbeitungsvorgänge gemeldet werden und der diese koordiniert. Auch mit Hilfe von Semaphoren wäre ein Schutz möglich. Ein Semaphore ist eine Markierung, die der Benutzer einer gemeinsamen Ressource setzt, bevor er sie belegt. Die Markierung wird von anderen potentiellen Nutzern als Sperre akzeptiert. Sie wird wieder entfernt, sobald die Ressource wieder freigegeben wird. Der Mitarbeiter, der den Preis ändern will, muss also ein Hinweisschild an die Preistafel hängen und wieder entfernen sobald er die neue Tafel aufgehängt hat.
3. a)	3	Der Superrechner benötigt ca. 95 a
b)	3	Verwendet man die Daten aus Teilaufgabe 3a, so erscheint ein Zeitraum von 95 a zum Brechen eines Schlüssels und damit das Verfahren selbst ausreichend sicher. Wenn die Rechengeschwindigkeit in Zukunft z. B. auf das 1 000 000-fache steigen würde, wäre mit den heutigen Verfahren die Faktorisierungszeit für das Produkt 20-stelliger Primzahlen bereits kritisch kurz und der RSA-Algorithmus nicht mehr ausreichend sicher. Mit größeren Primzahlen könnte man diese Probleme aber wieder in den Griff bekommen. Auch die parallele Verwendung mehrerer Rechner könnte die Faktorisierungszeit weiter verkürzen. Falls allerdings Verfahren entwickelt würden, bei denen die Zeit nicht exponentiell mit der Stellenzahl steigt, wäre die Faktorisierung auch bei größeren Primzahlen eventuell möglich. Unter diesen Umständen wäre dann der RSA-Algorithmus nicht mehr sicher. <i>Hinweise:</i> Nicht alle diese Argumente müssen genannt werden. Die Bewertung hängt davon ab, inwiefern die Einschätzung insgesamt schlüssig ist. Auf die Entwicklung der Rechengeschwindigkeit und die Möglichkeit größerer Primzahlen muss aber eingegangen werden; hier könnte auch das Gesetz von Moore genannt werden.

4. a)	2	Am Ende hat w den Wert 3 und r weiterhin den Wert 11.
b)	8	<pre> loadi 1 anfw: store w mul w sub r // w*w-r jmpnn endew // falls w*w-r >= 0 load w // falls w*w-r < 0 addi 1 // w+1 jmp anfw endew: sub w // (w*w-r)-w jmpnp endeb // falls (w*w-r)-w <= 0 load w // falls (w*w-r)-w > 0 subi 1 store w // w = w-1 endeb: hold r: word 11 // Speicherzelle für die Variable r w: word 0 // Speicherzelle für die Variable w </pre> <i>Hinweis:</i> Auch eine Lösung ohne symbolische Adressen ist zulässig. Dann ist die Angabe der Speicherzellen für die Variablen unbedingt erforderlich.
	40	

Aufgabe	BE	Hinweise
1. a)	4	In Zeile 99 wird 0 abgelegt, wenn a gerade ist, ansonsten 1.
b)	9	<pre> 1: load 100 2: sub 103 3: jle 18 // falls a-3 ≤ 0 4: load 100 5: div 102 6: mult 102 7: sub 100 8: jne 13 // falls a ungerade 9: load 100 // a ist gerade 10: div 102 11: store 100 // a=a/2 12: jump 1 13: load 100 // a ist ungerade 14: add 103 15: div 102 16: store 100 // a=(a+3)/2 17: jump 1 18: hold </pre>
2. a)	3	Terminale $T = \{\text{"Der", "Die", "Schüler", "Schülerin", "Abiturient", "Abiturientin", "lernt", "schreibt", "denkt", "übt", "feiert", ":", ",", "}"$ Nichtterminale $N = \{\text{Satz, Subjekt, Prädikat, Artikel, Substantiv}\}$ Startsymbol $S = \text{Satz}$
b)	1	Z. B. Der Schülerin schreibt.
c)	3	R2: Subjekt $\rightarrow$ "Der" SubstantivM "Die" SubstantivW R5a: SubstantivM $\rightarrow$ "Schüler" "Abiturient" R5b: SubstantivW $\rightarrow$ "Schülerin" "Abiturientin" R6: Satzzeichen $\rightarrow$ ":"
3. a)	2	Es liegt keine Verklemmung vor. Der Roboter B muss nur warten, bis der Roboter A mit dem Befestigen des Armaturenbretts fertig ist und die Ressourcen frei gibt. Dann kann B die Schrauben erhalten.
b)	5	<pre> graph LR Sieb[Sieb] --> Fritz(Fritz) Fritz -.-> Puderzucker[Puderzucker] Puderzucker --> Emil(Emil) Emil -.-> Messbecher[Messbecher] Messbecher --> Heinz(Heinz) Heinz -.-> Mehl(Mehl) Mehl --> Gerd(Gerd) Gerd -.-> Sieb </pre> Der Graph ist gerichtet, ungewichtet und zyklisch. Aus dem Graph ist ersichtlich, dass jeder Koch eine Ressource belegt und auf eine andere wartet. Es liegt also eine Verklemmung vor.
4. a)	2	Die Wörter enthalten eine ungerade Anzahl von Einsen.

b)	4	Mögliche Lösung:
c)	1	Der Automat akzeptiert z. B. 00001.
d)	6	
	40	