

Abiturprüfung 2018

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

Informatik

Aufgabe Inf1.I

Aufgabe	BE	Hinweise
1. a)	5	Z. B. Beschreibung des Wasserfallmodells mit seinen Phasen
b)	5	<pre> classDiagram class PINNWAND class LISTENELEMENT { <abstract> } class ABSCHLUSS class KNOTEN class NACHRICHT PINNWAND "1" -- "1" LISTENELEMENT : erster > LISTENELEMENT < -- ABSCHLUSS LISTENELEMENT < -- KNOTEN KNOTEN "1" -- "1" NACHRICHT : inhalt v KNOTEN "1" -- "1" KNOTEN : nachfolger </pre>
c)	2	Mögliche Argumente betreffen z. B. die Anzahl der Nachrichten, das Löschen einer Nachricht oder den benötigten Speicherplatz.
d)	4	<pre> sequenceDiagram participant pinnwand as pinnwand : PINNWAND participant k1 as k1 : KNOTEN participant k2 as k2 : KNOTEN participant k3 as k3 : KNOTEN participant a1 as a1 : ABSCHLUSS participant n1 as n1 : NACHRICHT participant n2 as n2 : NACHRICHT participant n3 as n3 : NACHRICHT pinnwand->>k1: erster k1->>n1: inhalt k1->>k2: nachfolger k2->>n2: inhalt k2->>k3: nachfolger k3->>n3: inhalt k3->>a1: nachfolger </pre>
e)	8	<pre> class Pinnwand{ private Listenelement erster; int anzahlNachrichtenGeben(){ return erster.anzahlNachrichtenGeben(); } } </pre>

```

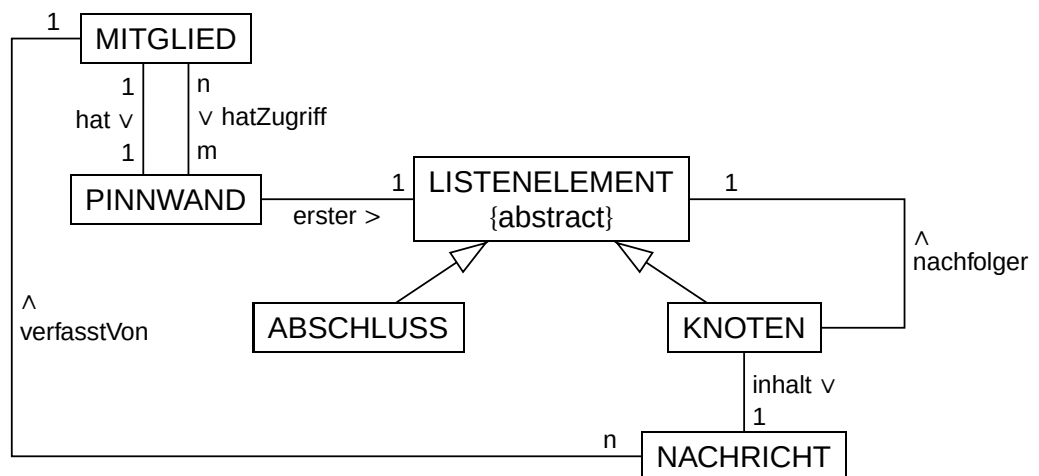
abstract class Listenelement{
    abstract int anzahlNachrichtenGeben();
}

class Knoten extends Listenelement{
    private Listenelement nachfolger;
    int anzahlNachrichtenGeben(){
        return 1 + nachfolger.anzahlNachrichtenGeben();
    }
}

class Abschluss extends Listenelement{
    int anzahlNachrichtenGeben(){
        return 0;
    }
}
    
```

f)

4



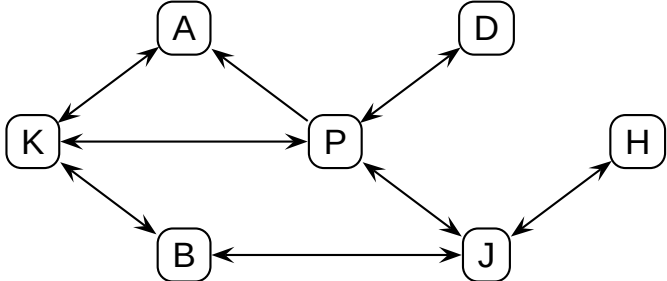
Hinweis: Auch eine textuelle Beschreibung ist möglich.

2. a)

2

Werden die Knoten mit aufsteigenden Mitgliedsnummern eingefügt, so entspricht die Struktur des dabei entstehenden Baums einer einfach verketteten Liste. Der Vorteil der schnellen Suche in einem ausbalancierten Baum ist somit nicht gegeben.

b)	4	Die neue Mitgliedsnummer wird mit nummer bezeichnet. nummer mit neuer Zufalls- zahl belegen wiederhole solange nummer im Baum vorhanden ist
c)	3	<pre> graph TD 3257 --> 116 3257 --> 66013 116 --> 2485 66013 --> 5485 66013 --> 833271 833271 --> 73680 </pre>
d)	3	2485 – 116 – 5485 – 73680 – 833271 – 66013 – 3257
e)	3	$\log_2(250000 + 1) \approx 17,9$. Es genügen also 18 Ebenen.
f)	12	<pre> class Baum{ private Baumelement wurzel; void mitgliederAusgeben(String suchnachname, String suchvorname){ wurzel.mitgliederAusgeben(suchnachname, suchvorname); } } abstract class Baumelement{ abstract void mitgliederAusgeben(String suchnachname, String suchvorname); } </pre>

		<pre> class Knoten extends Baumelement{ private Baumelement nachfolgerLinks; private Baumelement nachfolgerRechts; private Mitglied inhalt; void mitgliederAusgeben(String suchnachname, String suchvorname){ nachfolgerLinks.mitgliederAusgeben(suchnachname, suchvorname); if (inhalt.istNachnameGleich(suchnachname) && inhalt.istVornameGleich(suchvorname)){ inhalt.alleDatenAusgeben(); } nachfolgerRechts.mitgliederAusgeben(suchnachname, suchvorname); } } class Abschluss extends Baumelement{ void mitgliederAusgeben(String suchnachname, String suchvorname){ } } </pre>
g)	3	<pre> SELECT * FROM mitglieder WHERE nachname = "Meier" AND vorname = "Laura"; </pre>
3. a)	4	 In einem ungerichteten Graphen könnte man die Situation einer unbestätigten Freundschaftsanfrage nicht darstellen.

b)

3

	A	B	D	H	J	K	P
A						X	
B					X	X	
D							X
H					X		
J		X		X			X
K	X	X					X
P	X		X		X	X	

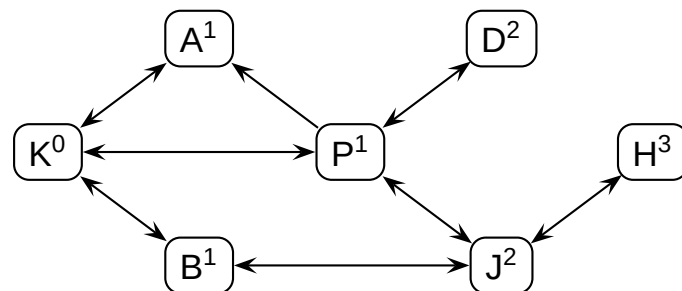
c)

5

Die Methode zählt für einen festen Knoten kn , zu wie vielen Nachbarknoten eine beidseitige Beziehung besteht. Der Rückgabewert ist also die Anzahl der Freunde des Mitglieds, mit dessen Knoten die Methode aufgerufen wurde.

d)

2



e)	8	<pre>Methode freundeVonFreunden(kn) i = Index von kn im Feld knoten wiederhole für j von 0 bis n-1 wenn matrix[j] [i] den Wert wahr hat und matrix[i] [j] den Wert wahr hat wiederhole für k von 0 bis n-1 wenn (matrix[k] [j] den Wert wahr hat und matrix[j] [k] den Wert wahr hat) und (matrix[k] [i] den Wert falsch hat oder matrix[i] [k] den Wert falsch hat) und k ≠ i gib k aus endeWenn endeWiederhole endeWenn endeWiederhole endeMethode</pre>
	80	

Informatik

Aufgabe Inf1.II

Aufgabe	BE	Hinweise
1. a)	5	<pre> classDiagram class SCHIFF { name motorleistung tiefgang ... } class SEE { name gröÙe ... } class ANLEGESTELLE { ort wassertiefe ... } SCHIFF "n" -- "1" SEE : befährt > SEE "1" -- "n" ANLEGESTELLE : hat > </pre>
b)	5	<pre> classDiagram class SCHIFF { <<abstract>> ... +restbetriebsstundenBerechnen()* ... } class ELEKTROBOOT { ... +restbetriebsstundenBerechnen() ... } class DIESELMOTORSCHIFF { ... +restbetriebsstundenBerechnen() ... } class SCHAUFELRADDAMPFER { } SCHIFF < -- ELEKTROBOOT SCHIFF < -- DIESELMOTORSCHIFF DIESELMOTORSCHIFF < -- SCHAUFELRADDAMPFER </pre>
c)	5	<pre> int anzahlTage(Datum saisonstart, Datum saisonende){ if (wartungstermin.istNach(saisonende)){ return saisonstart.dauer(saisonende); }else{ if (wartungstermin.istVor(saisonstart)){ return 0; }else{ return saisonstart.dauer(wartungstermin)-1; } } } </pre>

d)	3	Ein Softwaremuster ist ein wiederverwendbarer Lösungsansatz, der sich bereits bei ähnlichen Problemstellungen bewährt hat und deshalb bei der Entwicklung von Softwaresystemen immer wieder genutzt werden kann. Er beschleunigt somit die Umsetzung neuer Systeme.
e)	5	<pre> classDiagram class SCHIFFSLISTE class LISTENELEMENT { <abstract> } class ABSCHLUSS class KNOTEN class SCHIFF SCHIFFSLISTE --> "1" LISTENELEMENT : erster LISTENELEMENT < -- ABSCHLUSS LISTENELEMENT < -- KNOTEN LISTENELEMENT --> "1" KNOTEN : naechster KNOTEN --> "1" SCHIFF : inhalt v </pre>
f)	14	Methoden in Klasse SCHIFFSLISTE: <pre> int anzahlSchiffeBestimmen(){ return erster.anzahlBestimmen(); } void einfuegen(Schiff schiff){ erster = erster.einfuegen(schiff); } </pre> Methoden in Klasse LISTENELEMENT: <pre> abstract int anzahlBestimmen(); abstract Listenelement einfuegen(Schiff schiff); </pre> Methoden in Klasse ABSCHLUSS: <pre> int anzahlBestimmen(){ return 0; } Listenelement einfuegen(Schiff schiff){ return new Knoten(schiff, this); } </pre>

Methoden in Klasse KNOTEN:

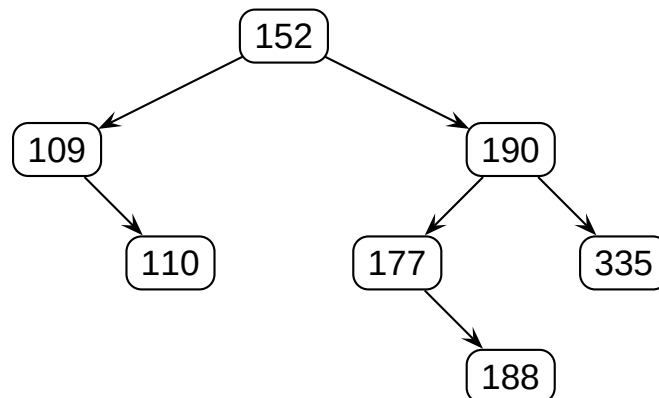
```

Knoten(Schiff schiff, Listenelement l){
    inhalt = schiff;
    naechster = l;
}
int anzahlBestimmen(){
    return naechster.anzahlBestimmen() + 1;
}
Listenelement einfuegen(Schiff schiff){
    if (inhalt.wartungsterminGeben().
        istVor(schiff.wartungsterminGeben())){
        naechster = naechster.einfuegen(schiff);
        return this;
    }else{
        return new Knoten(schiff, this);
    }
}

```

2. a) 3 SELECT persNr, name, vorname
FROM mitarbeiter
WHERE dienstort = "Königssee";

b) 6



Die Postorder-Traversierung liefert: 110 – 109 – 188 – 177 – 335 – 190 – 152

c)	4	Wenn der sortierte Binärbaum elementweise aus einer nach dem Ordnungskriterium sortierten Tabelle aufgebaut wird, entartet der Baum vollständig zu einer Liste. Eine mögliche Vorgehensweise zur Vermeidung dieses Problems wäre, die ursprüngliche Tabelle vorher nach einem anderen, möglichst stark von der Personalnummer abweichenden Kriterium zu sortieren.																																																																																																				
3. a)	5	Der vorliegende Graph ist gerichtet, gewichtet, zusammenhängend und nicht zyklensfrei. Adjazenzmatrix: <table><tr><td></td><td>Sb</td><td>Ch</td><td>H</td><td>N</td><td>Gs</td><td>Hr</td><td>Üb</td><td>Pr</td><td>Be</td></tr><tr><td>Seebruck</td><td></td><td>30</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Chieming</td><td></td><td></td><td>30</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Hauptsteg</td><td>30</td><td></td><td></td><td></td><td>10</td><td>10</td><td>25</td><td></td><td></td></tr><tr><td>Nordsteg</td><td></td><td></td><td></td><td></td><td>5</td><td></td><td></td><td></td><td></td></tr><tr><td>Gstadt</td><td></td><td></td><td>10</td><td>5</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Herreninsel</td><td></td><td></td><td>10</td><td></td><td></td><td></td><td></td><td>15</td><td>23</td></tr><tr><td>Übersee/Feldwies</td><td></td><td></td><td>25</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Prien/Stock</td><td></td><td></td><td></td><td></td><td></td><td>15</td><td></td><td></td><td></td></tr><tr><td>Bernau/Felden</td><td></td><td></td><td></td><td></td><td></td><td>23</td><td></td><td></td><td></td></tr></table>		Sb	Ch	H	N	Gs	Hr	Üb	Pr	Be	Seebruck		30								Chieming			30							Hauptsteg	30				10	10	25			Nordsteg					5					Gstadt			10	5						Herreninsel			10					15	23	Übersee/Feldwies			25							Prien/Stock						15				Bernau/Felden						23			
	Sb	Ch	H	N	Gs	Hr	Üb	Pr	Be																																																																																													
Seebruck		30																																																																																																				
Chieming			30																																																																																																			
Hauptsteg	30				10	10	25																																																																																															
Nordsteg					5																																																																																																	
Gstadt			10	5																																																																																																		
Herreninsel			10					15	23																																																																																													
Übersee/Feldwies			25																																																																																																			
Prien/Stock						15																																																																																																
Bernau/Felden						23																																																																																																
b)	4	<pre>graph LR; Be((Be)) -- 23 min --> Hr1((Hr)); Hr1 -- 10 min --> H2((H)); H2 -- 30 min --> Sb((Sb)); Sb -- 30 min --> Ch((Ch)); Ch -- 30 min --> H3((H)); H3 -- 10 min --> Gs((Gs)); Gs -- 10 min --> H4((H)); H4 -- 25 min --> Üb((Üb)); Üb -- 25 min --> H5((H)); H5 -- 10 min --> Hr2((Hr)); Hr2 -- 15 min --> Pr((Pr)); Pr -- 15 min --> Hr3((Hr)); Hr3 -- 23 min --> Be;</pre> Die Dauer T zum Durchlaufen dieses Zyklus beträgt unter Berücksichtigung von zwölf Zwischenaufenthalten 6 h 16 min.																																																																																																				

c)	6	<pre>private Knoten[] knoten; private int[][] adjazenzmatrix; private int anzahl; Liniennetz(int n){ knoten = new Knoten[n]; adjazenzmatrix = new int[n][n]; for(int i=0; i<n; i=i+1){ for(int j=0; j<n; j=j+1){ adjazenzmatrix[i][j] = -1; } } anzahl = 0; }</pre>
d)	5	<pre>int indexSuchen(Knoten k){ for(int i = 0; i < anzahl; i = i+1){ if (knoten[i] == k){ return i; } } return -1; }</pre>

e)	10	In der Klasse KNOTEN werden ein Attribut <i>besucht</i> sowie die zugehörige Gebe- und Setzemethode ergänzt. In der Klasse LINIENNETZ: <pre>void traversieren(Knoten k){ tiefensucheStarten(indexSuchen(k)); } void tiefensucheStarten(int startNr){ initialisierungGraphTiefensuche(); tiefensuche(startNr); } void tiefensuche(int index){ knoten[index].besuchtSetzen(true); System.out.println(knoten[index].nameGeben()); for (int i = 0; i < anzahl; i++){ if (adjazenzmatrix[index][i] != -1 && !knoten[i].besuchtGeben()){ tiefensuche(i); } } } void initialisierungGraphTiefensuche(){ for (int i = 0; i < anzahl; i++){ knoten[i].besuchtSetzen(false); } }</pre>
	80	

Informatik

Aufgabe Inf2.III

Aufgabe	BE	Hinweise
1.	4	R1: Rechenausdruck = Strichoperand { ('+' '-') Strichoperand }; R2: Strichoperand = Punktoperand { ('*' '/') Punktoperand }; R3: Punktoperand = Zahl Variable '(' Rechenausdruck ')'; R4: Variable = 'x' 'y' 'z'; R5: Zahl = Ziffer { Ziffer }; R6: Ziffer = '0' '1' '2' '3' '4' '5' '6' '7' '8' '9';
2. a)	2	(i) akzeptiert (ii) nicht akzeptiert (iii) nicht akzeptiert (iv) akzeptiert
b)	2	Die Zeichenketten enden auf 0 und besitzen an erster oder auch zweiter Stelle das Zeichen 1.
c)	6	<pre> class Automat{ private int zustand; boolean binaerzahlPruefen(String eingabe){ zustand = 0; for (int i=0; i<eingabe.length(); i++){ zeichenVerarbeiten(eingabe.charAt(i)); } return (zustand == 5); } </pre>

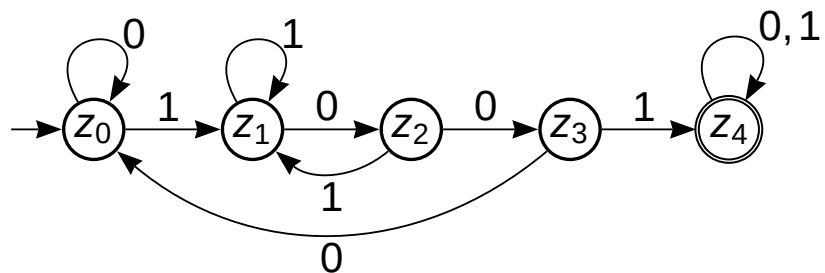
```

void zeichenVerarbeiten(char zeichen){
    switch (zustand){
        case 2:
            switch (zeichen){
                case '0':
                    zustand = 6;
                    break;
                case '1':
                    zustand = 3;
                    break;
            }
            break;
        ...
    }
}

```

3.

5



4. a)

3

1. Opcode aus der Speicherzelle holen, deren Adresse im Befehlszähler steht; Befehlszähler um 1 erhöhen
2. Opcode dekodieren (hier in "sub")
3. Operand aus der Speicherzelle holen, deren Adresse im Befehlszähler steht (hier "100"); Befehlszähler um 1 erhöhen
4. Differenz aus Akkumulatorinhalt und Inhalt der Speicherzelle 100 berechnen, Ergebnis in den Akkumulator schreiben

b)

1

Im Speicher liegen das Programm und die Daten.

c)	4		BZ	A	SR	100	101	102	nach Befehl
		0				4	5		
		2	4			4	5		load 100
		4	4	N		4	5		cmp 101
		6	4	N		4	5		jmpnn 12
		8	5	N		4	5		load 101
		10	5	N		4	5	5	store 102
		14	5	N		4	5	5	jmp 14
		16	5	N		4	5	5	hold
		Das Maximum beider Startwerte wird in die Speicherzelle 102 kopiert.							
d)	6	<i>n</i> steht in Speicherzelle 100 wdh: load 100 cmpi 3 jmpn ende modi 4 jmpnz sonst1 load 100 divi 4 store 100 jmp wdh sonst1: load 100 modi 2 jmpnz sonst2 load 100 divi 2 addi 1 store 100 jmp wdh sonst2: load 100 addi 1 divi 2 store 100 jmp wdh ende: hold							

e)	2	Für $n = 2$ wäre der Wert von n nach Ausführung der bedingten Anweisungen immer noch 2, weshalb die Abbruchbedingung der Wiederholung nie erfüllt werden würde; der Algorithmus würde nicht terminieren.
5. a)	3	Fahrer 1 muss auf Fahrer 3 (Linksabbiegerregel), Fahrer 3 muss auf Fahrer 2 („Rechts vor links“) und Fahrer 2 auf Fahrer 1 („Rechts vor links“) warten. Es muss also jeder Fahrer auf jeweils einen anderen Fahrer warten, so dass niemand fahren kann (Verklemmung). Eine Möglichkeit, die Verklemmung zu beheben, wäre beispielsweise, dass sich die Verkehrsteilnehmer per Handzeichen einigen, wer zuerst fahren darf.
b)	2	Ein kritischer Abschnitt ist ein Programmbereich, bei dessen Ausführung die Veränderung einer Ressource durch nebenläufige Prozesse zu Inkonsistenzen führen kann. Ein Monitor stellt einen geschützten Bereich zur Verfügung, der den Zugriff immer nur jeweils einem Prozess erlaubt.
	40	

Informatik

Aufgabe Inf2.IV

Aufgabe	BE	Hinweise
1. a)	5	ReKz = Buchstabe Buchstabe '-' ZifferOhneNull {Ziffer} Versicherungsart '-' ZifferOhneNull {Ziffer}; Versicherungsart = 'P' 'G' '0' ZifferOhneNull 'G' '1' '0' 'G' '1' '1' 'G' '1' '2'; Buchstabe = 'A' 'B' 'C' ... 'X' 'Y' 'Z'; ZifferOhneNull = '1' '2' '3' ... '8' '9'; Ziffer = '0' ZifferOhneNull;
b)	6	
c)	2	Z. B. über die Folge der durchlaufenen Zustände oder über einen Ableitungsbaum

d)	6	<pre> class Kennzahltester{ private int zustand; void zustandWechseln(char zeichen){ switch(zustand){ case 6: switch(zeichen){ case '0': zustand = 7; break; case '1': zustand = 9; break; default: zustand = -1; break; } break; // weitere Zustände ... } } boolean kennzahlTesten(String kennzahl){ zustand = 0; for (int i=0; i<kennzahl.length(); i++){ zustandWechseln(kennzahl.charAt(i)); } return zustand == 12; } } </pre>
2. a)	4	Eine Menge von Prozessen ist in einer Verklemmung, wenn jeder Prozess auf ein Ereignis wartet, das nur von einem anderen Prozess ausgelöst werden kann. Wenn bei einer Kreuzung aus allen Richtungen gleichzeitig ein Auto kommt, dann muss jedes Fahrzeug dem jeweils rechts stehenden die Vorfahrt über die Kreuzung gewähren. Es kommt daher zu einer Verklemmung.
b)	3	Eine Ampelanlage sorgt dafür, dass maximal zwei Fahrzeuge auf der Kreuzung in Konkurrenz treten können. In dem in Teilaufgabe 2a beschriebenen Fall würde die Ampelanlage somit dafür sorgen, dass es zu keiner Verklemmung mehr kommen kann.

3. a)	3	1. Befehl mit Operand (hier "102") aus der Speicherzelle holen, deren Adresse im Befehlszähler steht; Befehlszähler um 1 erhöhen 2. Befehl dekodieren (hier in "sub") 3. Differenz aus Akkumulatorinhalt und Inhalt der Speicherzelle 102 berechnen, Ergebnis in den Akkumulator schreiben
b)	5	Wert in Speicherzelle 104: 23 Werte des Befehlszählers: 1 – 2 – 3 – 10 – 11 – 12 – 13 – 14 – 15 – 18. Allgemein steht das Maximum der drei Zahlen in Speicherzelle 104.
c)	6	Die Methode berechnet die Summe der Quadrate der Zahlen von 1 bis 10. 1: dload 0 2: store 101 -- Summe in Speicherzelle 101 speichern 3: dload 10 4: store 102 -- i in Speicherzelle 102 speichern 5: dload 1 6: store 103 -- Zahl 1 in Speicherzelle 103 speichern 7: load 102 -- Beginn der Wiederholung 8: jle 16 9: mul 102 -- i^2 berechnen 10: add 101 11: store 101 -- $\text{summe} = \text{summe} + i^2$ 12: load 102 13: sub 103 14: store 102 -- $i = i - 1$ 15: jmp 7 16: end -- Programmende Der Rückgabewert steht in Speicherzelle 101.
	40	