

Abiturprüfung 2019

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

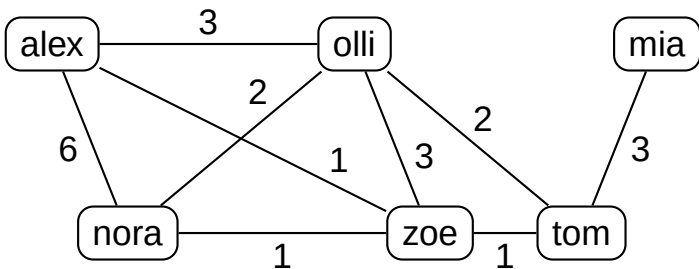
Aufgabe Inf1.I

Aufgabe	BE	Hinweise
1. a)	5	<pre> class SPIELER{ private String benutzername; private int punktestand, xpos, ypos; SPIELER(String neuerBenutzername){ benutzername = neuerBenutzername; punktestand = 100; xpos = zufallszahlErzeugen(0,999); ypos = zufallszahlErzeugen(0,999); } void punktestandSetzen(int p){ punktestand = p; } } </pre> <i>Hinweis: zufallszahlErzeugen(0,1000) ist ebenfalls zulässig.</i>
b)	4	<pre> graph LR liste([liste:SPIELERLISTE]) -- anfang --> k1([k1:KNOTEN]) k1 -- nachfolger --> k2([k2:KNOTEN]) k1 -- daten --> s1([s1:SPIELER]) k2 -- nachfolger --> a([a:ABSCHLUSS]) k2 -- daten --> s2([s2:SPIELER]) </pre>

c)	17	in SPIELERLISTE: <pre>void punktestandAendern(String benutzername, int wert){ anfang.punktestandAendern(benutzername, wert); } int maximum(){ return anfang.maximum(); }</pre> in LISTENELEMENT: <pre>abstract void punktestandAendern(String benutzername, int wert); abstract int maximum();</pre> in KNOTEN: <pre>void punktestandAendern(String benutzername, int wert){ if(daten.benutzernameGeben().equals(benutzername)){ daten.punktestandSetzen(daten.punktestandGeben()+wert); }else{ nachfolger.punktestandAendern(benutzername, wert); } } int maximum(){ int a = daten.punktestandGeben(); int b = nachfolger.maximum(); if(a>b){ return a; } else{ return b; } }</pre> in ABSCHLUSS: <pre>void punktestandAendern(String benutzername, int wert){ } int maximum(){ return 0; // da keine negativen Punktestände möglich }</pre>
----	----	---

d)	5	Die Fallen werden durch Objekte der Klasse FALLE repräsentiert. FALLE hat mindestens die vier ganzzahligen Attribute <i>xpos</i>, <i>ypos</i>, <i>komplexitaet</i> und <i>schaden</i>. Die Klasse SPIELSYSTEM besitzt ein FALLE-Feld der Länge 100. Ein Feld ist gut geeignet, da die Anzahl der Fallen bekannt und konstant ist.
2. a)	3	<pre> graph TD lena --> benni lena --> tom benni --> alex tom --> olli tom --> zoe olli --> mia olli --> sammy mia --> nora </pre>
b)	4	Zum Auffinden eines Knotens der untersten Ebene benötigt man mehr Vergleiche als bei einem Knoten einer höheren Ebene. Deshalb sollten möglichst wenige Knoten in der untersten Ebene sein. Als Wurzel des geforderten Baums kommen mia, nora und olli infrage. Dann enthalten beide Teilbäume drei bis fünf Knoten und die zweitunterste Ebene kann vollständig gefüllt werden.
c)	5	$\log_2(150\,000 + 1) \approx 17,2$ Wenn der Baum für die Suche optimal aufgebaut ist, hat er 18 Ebenen. Im schlechtesten Fall sind somit 18 Vergleiche nötig, diese dauern 90 Mikrosekunden. Eine solche Verzögerung ist für den neuen Nutzer beim Eintippen nicht wahrnehmbar.

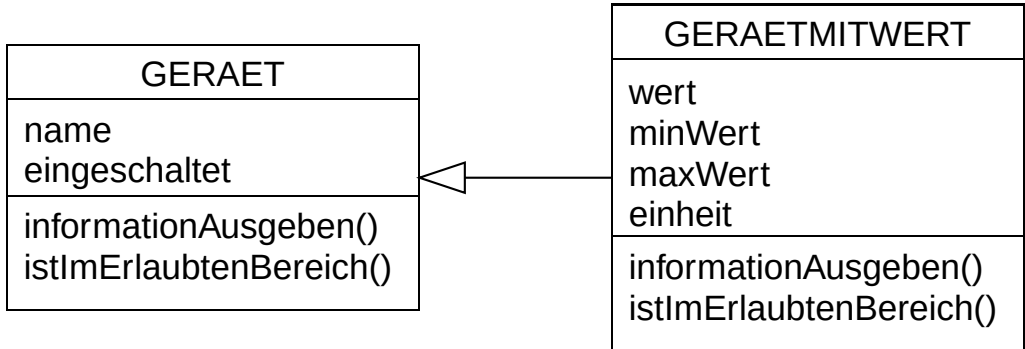
d)	8	in NUTZERBAUM: <pre>boolean istFrei(String neuerBenutzername){ return wurzel.istFrei(neuerBenutzername); }</pre> in BAUMELEMENT: <pre>abstract boolean istFrei(String neuerBenutzername);</pre> in KNOTEN: <pre>boolean istFrei(String neuerBenutzername){ int x = benutzername.compareTo(neuerBenutzername); if(x>0){ return liNachfolger.istFrei(neuerBenutzername); } else if(x<0){ return reNachfolger.istFrei(neuerBenutzername); } else return false; }</pre> in ABSCHLUSS: <pre>boolean istFrei(String neuerBenutzername){ return true; }</pre>
e)	5	Beim Methodenauf <i>m1('s')</i> wird eine Liste zurückgegeben, welche alle Benutzernamen, die mit s beginnen, in lexikographischer Reihenfolge enthält.

3. a)	5	Der Graph ist ungerichtet, gewichtet, unzusammenhängend und zyklisch. <table><tr><td></td><td>alex</td><td>mia</td><td>nora</td><td>olli</td><td>tom</td><td>zoe</td></tr><tr><td>alex</td><td>0</td><td>0</td><td>5</td><td>2</td><td>0</td><td>0</td></tr><tr><td>mia</td><td>0</td><td>0</td><td>0</td><td>0</td><td>3</td><td>0</td></tr><tr><td>nora</td><td>5</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>olli</td><td>2</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>tom</td><td>0</td><td>3</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>zoe</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr></table>		alex	mia	nora	olli	tom	zoe	alex	0	0	5	2	0	0	mia	0	0	0	0	3	0	nora	5	0	0	1	0	0	olli	2	0	1	0	0	1	tom	0	3	0	0	0	0	zoe	0	0	0	1	0	0
	alex	mia	nora	olli	tom	zoe																																													
alex	0	0	5	2	0	0																																													
mia	0	0	0	0	3	0																																													
nora	5	0	0	1	0	0																																													
olli	2	0	1	0	0	1																																													
tom	0	3	0	0	0	0																																													
zoe	0	0	0	1	0	0																																													
b)	4																																																		
c)	6	Methode kantenanzahlGeben(index) <table><tr><td colspan="2">x=0</td></tr><tr><td colspan="2">zähle k von 0 bis anzahl-1</td></tr><tr><td colspan="2">matrix[index][k]>0</td></tr><tr><td>wahr</td><td>falsch</td></tr><tr><td>x=x+1</td><td></td></tr><tr><td colspan="2">gib x zurück</td></tr></table>	x=0		zähle k von 0 bis anzahl-1		matrix[index][k]>0		wahr	falsch	x=x+1		gib x zurück																																						
x=0																																																			
zähle k von 0 bis anzahl-1																																																			
matrix[index][k]>0																																																			
wahr	falsch																																																		
x=x+1																																																			
gib x zurück																																																			

d)	9	<pre>Methode besterKooperatorGeben() setze bester = 0 setze max = 0 zähle i von 0 bis anzahl-1 setze aktuell = kantenanzahlGeben(i) wenn aktuell > max dann setze max = aktuell setze bester = i endeWenn endeZähle gib bester zurück endeMethode</pre> Der Algorithmus ermittelt Olli als besten Kooperator. Beispielsweise könnte sich Alex als kooperativer ansehen, weil er häufiger (siebenmal) mit anderen zusammengearbeitet hat als Olli (viermal).
	80	

Informatik

Aufgabe Inf1.II

Aufgabe	BE	Hinweise
1.	4	z. B. Analyse, Entwurf, Implementierung, Test Erstellen des Pflichtenhefts: Analyse Einsatz von Softwaremustern: Entwurf Aufwandsabschätzung: Analyse/Entwurf Modellieren mithilfe von Diagrammen: Analyse/Entwurf
2. a)	5	 <pre> classDiagram class GERAET { name eingeschaltet informationAusgeben() istImErlaubtenBereich() } class GERAETMITWERT { wert minWert maxWert einheit informationAusgeben() istImErlaubtenBereich() } GERAETMITWERT < -- GERAET </pre>
b)	13	<pre> class Geraet{ protected String name; protected boolean eingeschaltet; Geraet(String s){ name = s; eingeschaltet = false; } void informationAusgeben(){ System.out.print(name+": "); if(eingeschaltet){ System.out.println("an"); }else{ System.out.println("aus"); } } boolean istImErlaubtenBereich(){ return true; } } </pre>

```

class GeraetMitWert extends Geraet{
    private int wert;
    private int minWert;
    private int maxWert;
    private String einheit;
    GeraetMitWert(String n, int min, int max, String e){
        super(n);
        minWert = min;
        maxWert = max;
        wert = 0;    // willkürlich, nicht gefordert
        einheit = e;
    }
    boolean istImErlaubtenBereich(){
        return wert<=maxWert && wert>=minWert;
    }
    void informationAusgeben(){
        System.out.print(name+": ");
        if(eingeschaltet){
            System.out.println(wert+" "+einheit);
        }else{
            System.out.println("aus");
        }
    }
}

```

c)

6

In LISTE:

```

void handlungsbedarfAnzeigen(){
    erster.handlungsbedarfAnzeigen();
}

```

In LISTENELEMENT:

```

abstract void handlungsbedarfAnzeigen();

```

In ABSCHLUSS:

```

void handlungsbedarfAnzeigen() {}

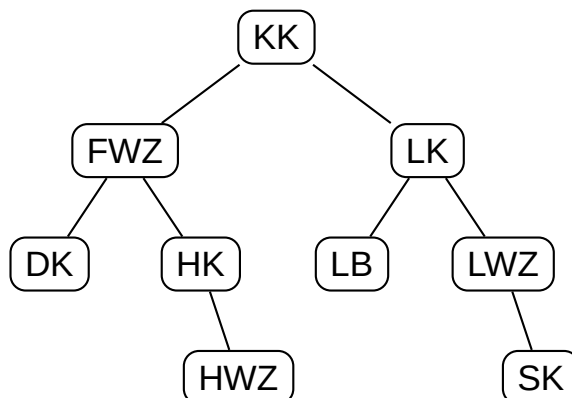
```

		In KNOTEN: <pre> void handlungsbedarfAnzeigen(){ if(!inhalt.istImErlaubtenBereich()){ inhalt.informationAusgeben(); } naechster.handlungsbedarfAnzeigen(); } </pre>
d)	7	<pre> Listenelement vorruecken(Geraet geraet){ if(naechster.inhaltGeben() != geraet){ naechster = naechster.vorruecken(geraet); return this; }else{ Listenelement k = naechster; naechster = naechster.naechsterGeben(); k.naechsterSetzen(this); return k; } } </pre>
3. a)	6	<pre> graph TD b1([b1:BAUM]) -- wurzel --> k1([k1:KNOTEN]) k1 -- nachfolgerLinks --> ab1([ab1:ABSCHLUSS]) k1 -- inhalt --> g1([g1:GERAET name="DK"]) k1 -- nachfolgerRechts --> k2([k2:KNOTEN]) k2 -- nachfolgerLinks --> ab2([ab2:ABSCHLUSS]) k2 -- inhalt --> g2([g2:GERAET name="KK"]) k2 -- nachfolgerRechts --> ab3([ab3:ABSCHLUSS]) </pre>
b)	4	Falls die Geräte in alphabetischer Reihenfolge in den Baum eingefügt werden, entartet dieser zu einer Liste. Er hat dann also 534 Ebenen. Da $\log_2(534 + 1) \approx 9,1$ ist, hat der Baum mit geringster Höhe 10 Ebenen.

c)

7

Z. B.: KK, FWZ, LK, DK, HK, HWZ, LB, LWZ, SK

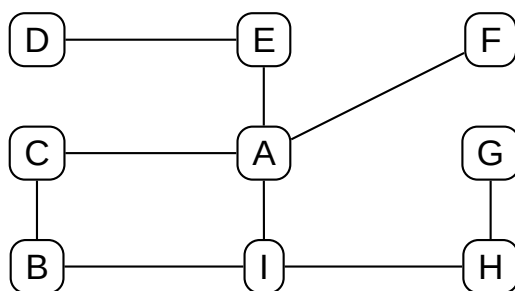


Preorder: KK FWZ DK HK HWZ LK LB LWZ SK

Postorder: DK HWZ HK FWZ LB SK LWZ LK KK

4. a)

5



	A	B	C	D	E	F	G	H	I
A			1		1	1			1
B			1						1
C	1	1							
D					1				
E	1			1					
F	1								
G								1	
H							1		1
I	1	1						1	

b)

9

z. B. Tiefensuche

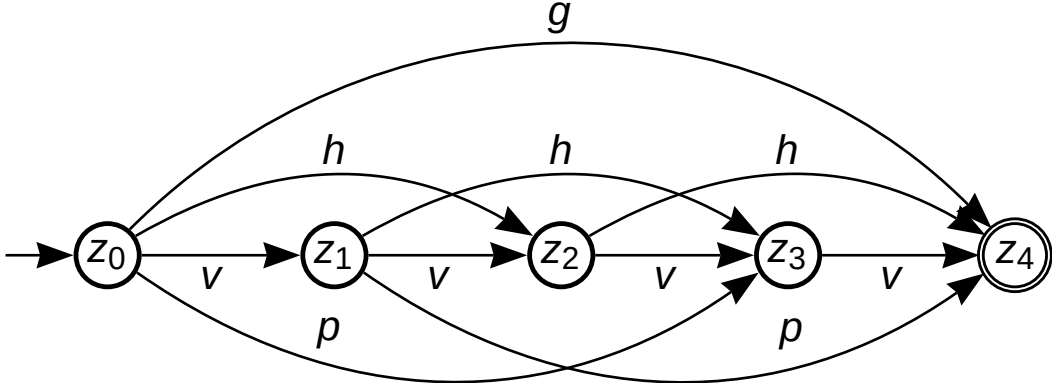
Mögliche Reihenfolge: A – C – B – I – H – G – E – D – F

Der vollständige Durchlauf ist möglich, da der Graph zusammenhängend ist.

c)	7	Methode sackgassenAusgeben() zähle i von 0 bis anzahl-1 setze x=0 zähle k von 0 bis anzahl-1 wenn matrix [i] [k] > 0 dann setze x=x+1 endeWenn endeZähle wenn x gleich 1 dann gib Name des Knotens mit Index i aus endeWenn endeZähle endeMethode
5. a)	4	Beim MVC wird eine Trennung von Oberfläche (View) und Programmlogik (Model) vorgenommen. Diese Trennung ist sinnvoll, da die Ausgabe hier z. B. auf dem Smartphone verlangt ist und die Implementierung der Logik nicht von Details der Darstellungsmöglichkeiten des Smartphones abhängen sollte. Um Eingaben auf der Oberfläche zu verarbeiten, ist hier auch der Controller notwendig, um das Model, das die Geräte repräsentiert, zu steuern. Außerdem verwaltet der Controller den Zustand der Oberfläche.
b)	3	Bei Anklicken des „+1“-Buttons nimmt die Oberfläche die Eingabe entgegen und informiert den Controller darüber. Der Controller verarbeitet die Eingabe und steuert das Model mittels dessen bereitgestellter Methoden, d. h. er erhöht hier den Wert für die gewünschte Temperatur um 1 °C und informiert danach die Oberfläche, die die Anzeige entsprechend anpasst.
	80	

Informatik

Aufgabe Inf2.III

Aufgabe	BE	Hinweise
1. a)	3	 gehört zu S: <Takt> $\xrightarrow{(R1)}$ <3/4>  $\xrightarrow{(R2)}$ <1/2>  $\xrightarrow{(R3)}$   gehört nicht zu S: Die erste anwendbare Regel ist (R1). Bei der jetzt anzuwendenden Regel (R2) passt keine Alternative zur verbleibenden Notenfolge. Beispiel für ein weiteres Wort, das nicht zu S gehört: 
b)	3	R1: <Takt> $\rightarrow$  <3/4>   <3/4> <1/2> <1/2> R2: <3/4> $\rightarrow$  <1/2>   <1/2> R3: <1/2> $\rightarrow$  
c)	4	
d)	7	<pre> class Automat{ private int zustand; boolean istViervierteltakt(String zk){ zustand = 0; for(int i = 0; i < zk.length(); i++){ zustandWechseln(zk.charAt(i)); } return zustand == 4; } } </pre>

```

private void zustandWechseln(char c){
    switch(zustand){
        case 0: switch(c){
            case 'g': zustand = 4; break;
            case 'p': zustand = 3; break;
            case 'h': zustand = 2; break;
            case 'v': zustand = 1; break;
            default: zustand = -1;
        } break;
        ...
    }
}

```

2. a)

4

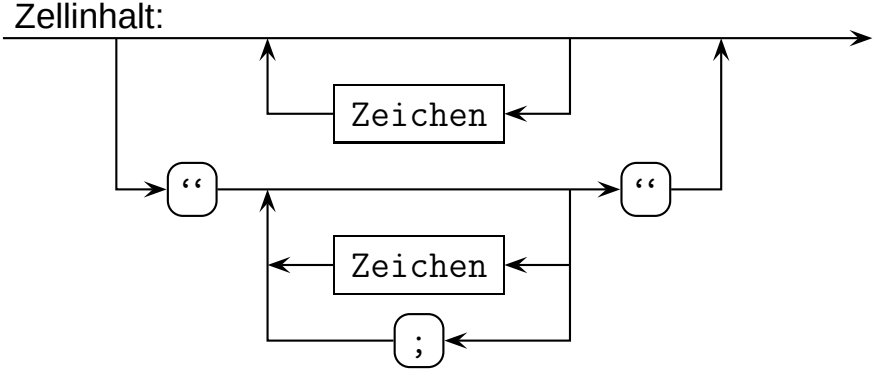
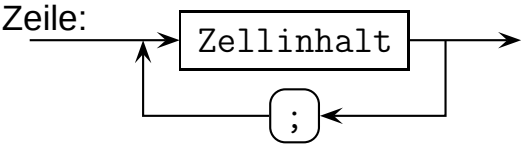
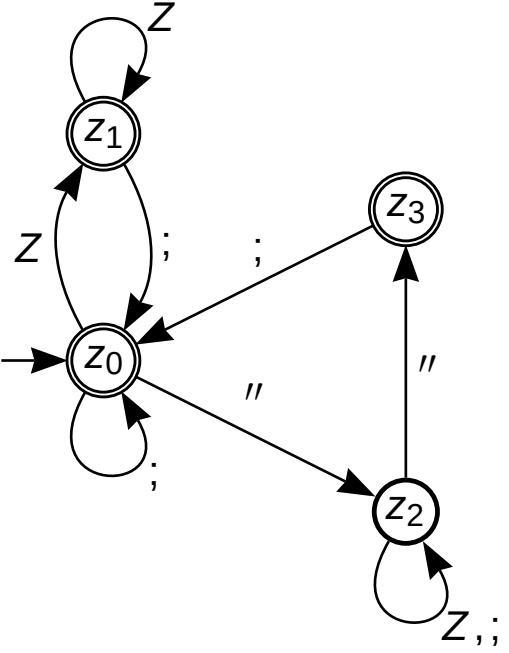
n	z	m
243		0
	3	3
24	4	4
2		
	2	
0		

Der Rückgabewert ist die größte Ziffer der gegebenen Zahl n.

b)	6	1: loadi 0 2: store 102 3: load 100 4: jle 15 5: modi 10 6: store 101 7: sub 102 8: jlt 11 9: load 101 10: store 102 11: load 100 12: divi 10 13: store 100 14: jmp 3 15: end
3. a)	2	z. B. Monitorkonzept
b)	4	Falls für die Methoden <i>buchen</i> und <i>buchenMit</i> wechselseitiger Ausschluss garantiert ist, kann es zu folgender Situation kommen: Thread A führt den Aufruf <i>platz1.buchenMit(platz2)</i> aus. Thread B führt den Aufruf <i>platz2.buchenMit(platz1)</i> aus, bevor Thread A <i>nachbar.buchen()</i> aufruft. Dann sind beide Threads nicht in der Lage <i>nachbar.buchen()</i> auszuführen und warten, bis der jeweils andere Thread den kritischen Abschnitt wieder verlässt. Die Threads befinden sich somit in einer Verklemmung.
4.	7	Für $n = 4$ erfolgt der Vergleich „ $a[i]$ gleich $b[j]$ “ im ungünstigsten Fall bei Algorithmus I 16-mal und bei Algorithmus II 7-mal. Allgemein benötigt Algorithmus I höchstens n^2 und Algorithmus II höchstens $2n-1$ Ausführungen des Vergleichs. An den Termen erkennt man, dass Algorithmus I ein quadratisches Laufzeitverhalten (Diagramm C) und Algorithmus II ein lineares Laufzeitverhalten (Diagramm B) aufweist.
	40	

Informatik

Aufgabe Inf2.IV

Aufgabe	BE	Hinweise
1. a)	5	Zellinhalt:  Zeile: 
b)	6	
2. a)	4	L hat zehn Elemente. Begründung durch Aufzählung (00111, 01011, 01101, 01110, 10011, 10101, 10110, 11001, 11010, 11100) oder mit Hilfe der Kombinatorik: $\binom{5}{3}$ Der Automat akzeptiert alle Wörter der Länge 5 mit genau zwei Zeichen 0 und drei Zeichen 1.

b)	2	Ein endlicher Automat erkennt genau dann nur endlich viele Wörter, wenn jeder Pfad vom Startzustand zu einem Endzustand zyklensfrei ist.
3.	5	Die Anzahl der möglichen Zugangs-codes, die generiert werden, beträgt bei 26 Buchstaben im Alphabet 26^{10}. Da der Angreifer $4 \cdot 10^9$ Zugangs-codes pro Sekunde ausprobieren kann, benötigt er höchstens $\frac{26^{10}}{4 \cdot 10^9} \text{ s} \approx 35292 \text{ s} = 9 \text{ h } 48 \text{ min } 12 \text{ s} < 24 \text{ h},$ um alle Zugangs-codes bei seinem Zugang auszuprobieren und damit sicher den Zugang zu hacken. Z. B. führt die Hinzunahme von Großbuchstaben zu 52^{10} möglichen Zugangs-codes. Die Brute-Force-Methode würde dann im schlechtesten Fall $\frac{52^{10}}{4 \cdot 10^9} \text{ s} \approx 36138776 \text{ s} = 10038 \text{ h } 32 \text{ min } 56 \text{ s} \gg 24 \text{ h}$ und damit meistens zu lange dauern. Die so generierten Zugangs-codes sind deutlich sicherer.
4. a)	2	7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1
b)	2	Geht die Division a/b ohne Rest auf, so liefert $(a/b) \cdot b$ wieder a , ansonsten ist $(a/b) \cdot b$ kleiner als a .

c)	7	<pre> 1 : loadi 0 2 : store 101 3 : load 100 4 : subi 1 5 : jeq 23 6 : load 101 7 : addi 1 8 : store 101 9 : load 100 10: divi 2 11: muli 2 12: sub 100 13: jlt 18 14: load 100 15: divi 2 16: store 100 17: jmp 3 18: load 100 19: muli 3 20: addi 1 21: store 100 22: jmp 3 23: hold </pre> - - Anzahl der Durchläufe in Zelle 101
5. a)	4	<pre> graph TD E1[E1] --> A((A)) S1[S1] --> A S2[S2] --> C((C)) E2[E2] --> B((B)) B -.-> S2 C -.-> E2 </pre> Ada hat einen Eimer und eine Schaufel und ist zufrieden. Bill hat den zweiten Eimer und möchte die zweite Schaufel haben, die aber Charles belegt, der wiederum auf den zweiten Eimer wartet.

b)	2	Eine Verklemmung erkennt man an einem Zyklus im Graphen.
c)	1	Z. B.: • Bill überlässt Charles seinen Eimer. • Bill wartet darauf, dass Ada ihre Burg fertiggestellt hat und bekommt dann ihre Schaufel. • Bill und Charles bauen gemeinsam eine Sandburg.
	40	