Ministerium für Bildung und Kultur
SAARLAND

Lösungshinweise:

Nur für die Hand der Lehrperson

Schriftliche Abiturprüfung 2018

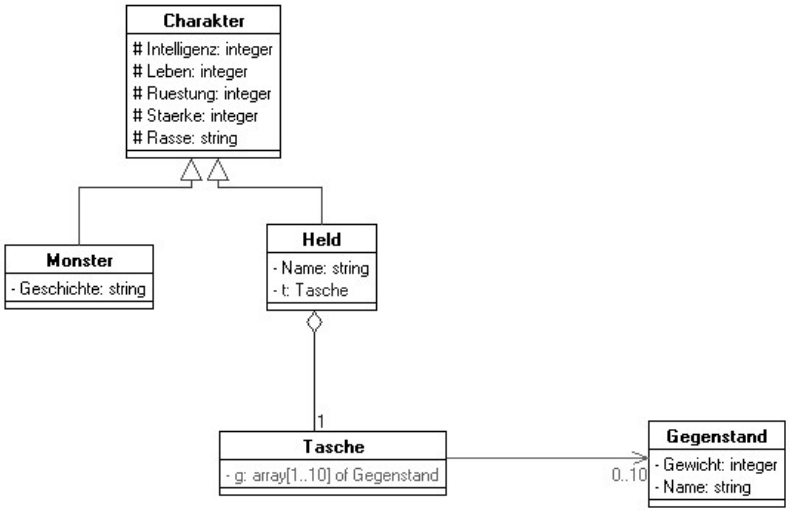
Fach:	**Informatik**
Kurstyp:	G-Kurs
Datum:	17. April 2018
Bearbeitungszeit:	3 Zeitstunden
Hilfsmittel:	nicht-programmierbarer Taschenrechner
Seitenzahl:	Die Lösungshinweise umfassen mit Deckblatt, Punkteverteilungstabelle und Bewertungstabelle 13 Seiten.

Trierer Str. 33 | 66111 Saarbrücken
www.saarland.de

1. Aufgabe (20 Punkte)

1.1. Strukturierte Programmierung und Objektorientierung (13 Punkte)

1.1.1. (6 Punkte)



1.1.2. (4 Punkte)

Java Code:

```
public class Gegenstand
{
    private String name;
    private int gewicht;

    public Gegenstand(String pName, int pGewicht)
    {
        name=pName;
        gewicht=pGewicht;
    }

    public String getName()
    {
        return name;
    }

    public int getGewicht()
    {
        return gewicht;
    }

    public void setName(String pName)
    {
        name=pName;
    }

    public void setGewicht(int pGewicht)
    {
        gewicht=pGewicht;
    }


}
```

ObjectPascal Code:

```
type
 TGegenstand = class
  private
    name: String;
    gewicht: integer;
  public
    constructor create(pname: String; pgewicht: integer);
    function getName: String;
    function getGewicht: integer;
    procedure setGewicht(pGewicht:integer);
    procedure setName(pName:string);
  end;

implementation

  constructor TGegenstand.create(pname:String;pgewicht:integer);
  begin
    name:= pname;
    gewicht:= pgewicht;
  end;

  function TGegenstand.getName: String;
  begin
    result:= name;
  end;

  function TGegenstand.getGewicht: integer;
  begin
    result:= gewicht;
  end;

  procedure TGegenstand.setGewicht(pGewicht:integer);
  begin
    gewicht:= pgewicht;
  end;

  procedure TGegenstand.setName(pname:string);
  begin
    name:=pname;
  end;
```

1.1.3. (3 Punkte)

Java Code:

```
public void angreifen(Monster monster)
{
    int angriff=this.staerke- monster.getRuestung();
    if (angriff>0)
    {
        monster.setLeben(monster.getLeben()-angriff);
        if (monster.getLeben()<0)
        {
            monster.setLeben(0);
        }
    }
}
```

ObjectPascal Code:

```
procedure TCharakter.angreifen(Monster: TMonster);
var angriff: integer;
begin
   angriff:= staerke- Monster.getRuestung;
   if (angriff > 0) then
   begin
     Monster.setLeben(Monster.getLeben - angriff);
     if (Monster.getLeben < 0) then
        Monster.setLeben(0);
   end;
end;
```

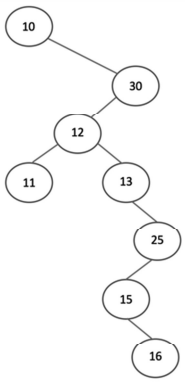
1.2 Binäre Suchbäume (7 Punkte) (Lehrplan: Datentypen und Datenstrukturen)

1.2.1. (1 Punkt)

$h \geq \log_2 130 = \frac{\ln(130)}{\ln(2)} \approx 7{,}02$

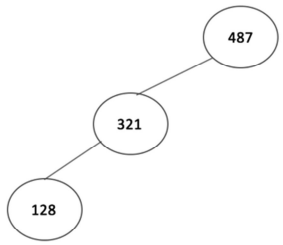
Es gilt daher $h = 8$.

1.2.2. (3 Punkte)

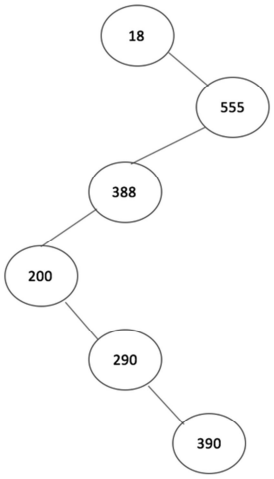


Der resultierende Baum ist als Suchbaum nicht effektiv, da er entartet ist.
Im schlechtesten Fall kann dies zu linearem Aufwand von Einfüge-, Lösch- und Suchoperationen führen.

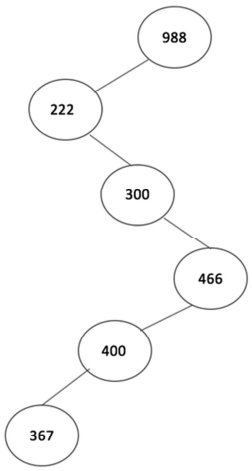
1.2.3. (3 Punkte)



Nach dem Knoten mit dem Schlüssel 321 müsste bei der Suche nach dem Knoten 367 ein Knoten mit größerem Schlüssel in der Sequenz folgen.



Widerspruch zur Suchbaumeigenschaft:
390 steht im linken Teilbaum von 388.



Mögliche korrekte Knotenfolge.

2. Aufgabe (20 Punkte)

2.1. (13 Punkte)
(Lehrplan: Funktionsweise von Computersystemen, Automaten und Formale Sprachen)

2.1.1. (1 Punkt)

(8+9) mod 3 = 17 mod 3 = 2
Die vollständige Kreditkartennummer mit Prüfziffer lautet also 892.

2.1.2. (5 Punkte – davon 1 für Kommentare)

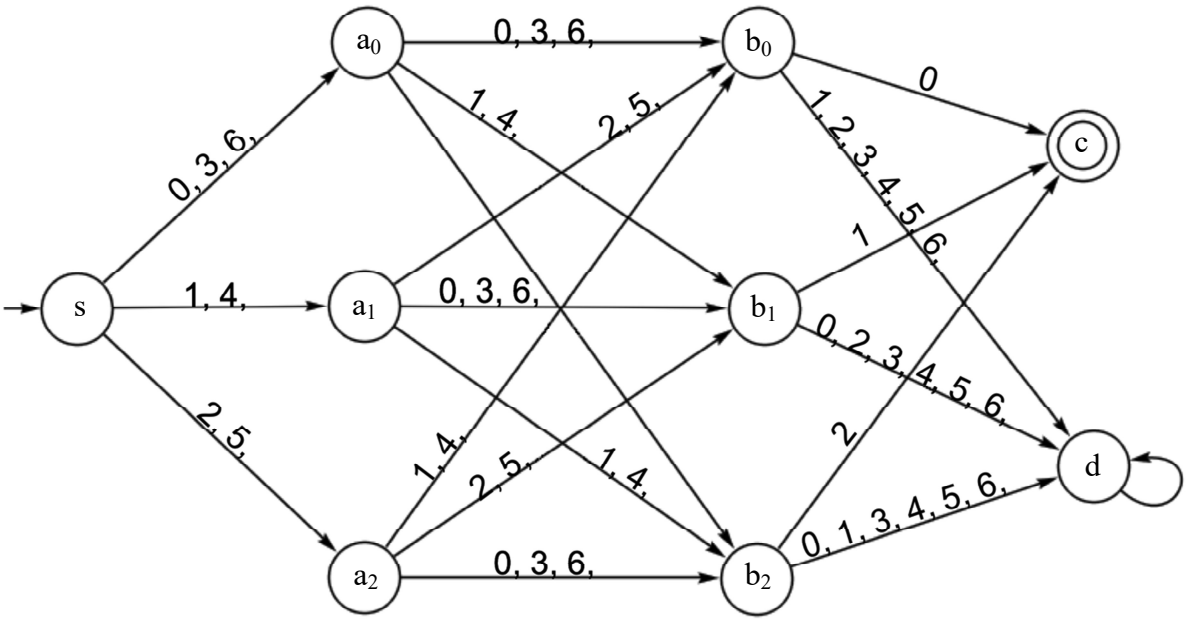
0	JMP 5	Sprung zum Programmstart
1	DEF 0	1.Ziffer
2	DEF 0	2.Ziffer
3	DEF 3	Zahl 3
4	DEF 0	(Zwischen-)Ergebnis
5	INM 1	Einlesen der 1. Ziffer
6	INM 2	Einlesen der 2. Ziffer
7	LDA 1	
8	ADD 2	Summe der Ziffern bilden
9	STA 4	und in Speicherplatz 4 speichern
10	LDA 4	
11	SUB 3	von der Summe die Zahl 3 (Speicher 3) subtrahieren
12	JNM 11	solange das Zwischenergebnis >= 0 ist
13	ADD 3	zuviel subtrahierte 3 wieder dazuaddieren
14	STA 4	Ergebnis auf Speicherplatz 4 ablegen
15	OUT 4	Ausgabe
16	END	

2.1.3. (4 Punkte)

Hinweis. Ein korrekter Übergangsgraph genügt.

$DEA = (Z, \sum, \delta, s, E)$ mit

- $Z = \{s, a_0, a_1, a_2, b_0, b_1, b_2, c, d\}$
- $\sum = \{0,1,2,3,4,5,6,7,8,9\}$
- $q_0 = S$
- $E = \{c\}$
- δ: siehe Zustandsübergangsgraph



2.1.4. (3 Punkte)

2.1.4.1. (2 Punkte)

Die Ziffernfolge 641 gehört zur Sprache K, denn:

$$S \rightarrow B1 \rightarrow DE1 \rightarrow D41 \rightarrow 641$$

Die Ziffernfolge 830 gehört hingegen nicht zur Sprache K:

$$S \rightarrow A0 \rightarrow DD0 \rightarrow D30$$

An dieser Stelle gibt es keine Produktionsregel, die zur Ziffernfolge 830 führt.

2.1.4.2. (1 Punkt)

Die angegebene Grammatik ist nicht regulär, da auf der rechten Seite der Produktionsregeln maximal ein Nichtterminalsymbol stehen dürfte und es in dieser Grammatik mehrere Produktionsregeln ((2), (3), (4)) gibt, die diese Voraussetzung nicht erfüllen.

2.2. (7 Punkte)

2.2.1. (2 Punkte)

$G = (N, T, P, S)$ mit

$N = \{S, A, B, C\}$

$T = \{0, 1, 2\}$

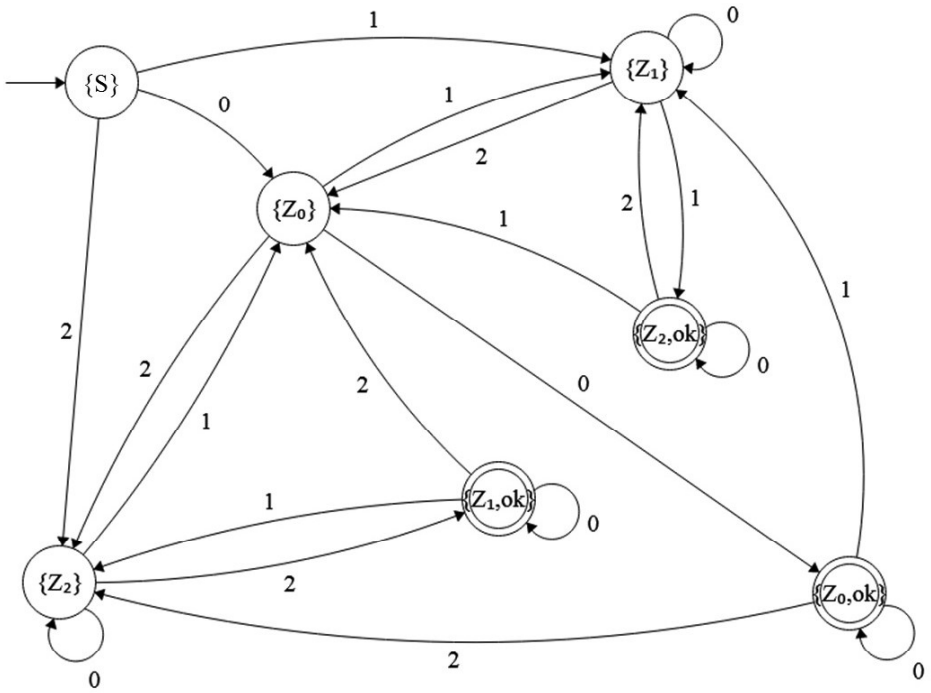
$$P = \begin{Bmatrix} S \rightarrow 0A \mid 1B \mid 2C \\ A \rightarrow 0A \mid 1B \mid 2C \mid 0 \\ B \rightarrow 0B \mid 1C \mid 2A \mid 1 \\ C \rightarrow 0C \mid 1A \mid 2B \mid 2 \end{Bmatrix}$$

2.2.2. (5 Punkte)

Teilmengenkonstruktion:

	0	1	2
$Z' := \{S\}$	$\{Z_0\}$	$\{Z_1\}$	$\{Z_2\}$
$Z'_0 := \{Z_0\}$	$\{Z_0, ok\}$	$\{Z_1\}$	$\{Z_2\}$
$Z'_1 := \{Z_1\}$	$\{Z_1\}$	$\{Z_2, ok\}$	$\{Z_0\}$
$Z'_2 := \{Z_2\}$	$\{Z_2\}$	$\{Z_0\}$	$\{Z_1, ok\}$
$Z'_3 := \{Z_0, ok\}$	$\{Z_0, ok\}$	$\{Z_1\}$	$\{Z_2\}$
$Z'_4 := \{Z_1, ok\}$	$\{Z_1, ok\}$	$\{Z_2\}$	$\{Z_0\}$
$Z'_5 := \{Z_2, ok\}$	$\{Z_2, ok\}$	$\{Z_0\}$	$\{Z_1\}$

Zugehöriger Zustandsübergangsgraph:



3. Aufgabe (20 Punkte)

3.1. (11 Punkte) (Lehrplan: "Kryptographie")

3.1.1. (5 Punkte)

Es gilt:

$$n = p \cdot q = 13 \cdot 23 = 299$$
$$\varphi(n) = (p-1) \cdot (q-1) = 12 \cdot 22 = 264$$

Da e kein Teiler von $\varphi(n)$ sein darf, kommt $(33{,}299)$ nicht in Frage, da 33 Teiler von 264 ist.

(43,299)	(53,299)
$264 = 6 \cdot 43 + 6$ $43 = 7 \cdot 6 + 1$ $1 = 43 - 7 \cdot 6$ $\Leftrightarrow = 43 - 7 \cdot (264 - 6 \cdot 43)$ $\Leftrightarrow = -7 \cdot 264 + 43 \cdot 43$	$264 = 4 \cdot 53 + 52$ $53 = 1 \cdot 52 + 1$ $1 = 53 - 1 \cdot 52$ $\Leftrightarrow = 53 - 1 \cdot (264 - 4 \cdot 53)$ $\Leftrightarrow = -1 \cdot 264 + 5 \cdot 53$
Da hier (d,n) = (43,299) = (e,n) gilt, ist diese Wahl im Sinne der asymmetrischen Kryptographie nicht günstig	Also ist (d,n) = (5,299). Dieser Schlüssel ist somit geeignet.

3.1.2. (4 Punkte)

3.1.2.1. (1 Punkt)

Der Schlüssel muss so gewählt sein, dass $n > m$ ist. Diese Eigenschaft ist in diesem Fall nicht gegeben.

3.1.2.2. (2 Punkte)

$$13^7 \; mod \; 299 \equiv 62.748.517 \; mod \; 299$$
$$\equiv 78 \; mod \; 299$$

Somit gilt $c' = 78$

3.1.2.3. (1 Punkt)

$$m' = 78^{151} \; mod \; 299$$

3.1.3. (2 Punkte)

Durch Verschlüsseln einer Nachricht mit dem privaten Schlüssel des Absenders und mit dem öffentlichen Schlüssel des Empfängers werden alle drei Sicherheitsziele erreicht.

3.2 (9 Punkte) (Lehrplan: "Rekursion")

3.2.1 (2 Punkte)

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_n	0	1	1	2	1	3	2	3	1	4	3	5	2	5	3	4

3.2.2 (3 Punkte)

Lösungsvorschlag delphi

```
function Stern_Brocot(n:integer):integer;
var h:integer;
begin
  if n<2 then result := n
    else if (n mod 2 = 0) then result := Stern_Brocot(n div 2)
    else
          begin
            h:= (n-1) div 2;
            result := Stern_Brocot(h)  + Stern_Brocot(h+1)
          end;
end;
```

Lösungsvorschlag java

```
static int Stern_Brocot(int n)
{         int h;
          if (n<2) return n;
                    else if ( n % 2 == 0) return Stern_Brocot(n / 2)
                    else {
                              h = (n-1) / 2;
                              return Stern_Brocot(h) + Stern_Brocot(h+1);
                         }
}
```

3.2.3 (4 *Punkte)*

3.2.3.1 (1 Punkt)

$i = 2^0$	1							
$i = 2^1$	1	2						
$i = 2^2$	1	3	2	3				
$i = 2^3$	1	4	3	5	2	5	3	4
…								

Summe 4. Zeile = 1+4+3+5+2+5+3+4 =27 = 3^{4-1}

3.2.3.2 (3 Punkte)

Lösungsvorschlag Delphi

```
function istDreierpotenz(k:integer):Boolean;
var anfang,ende,i,summe:integer;
begin
  summe := 0;
  anfang := power(2,k);
  ende := power(2,k+1) – 1;
  for i := anfang to ende do summe:=summe + Stern_Brocot(i);
  result := ( summe = power(3,k-1) );
end;
```

Lösungsvorschlag JAVA

```
static Boolean istDreierPotenz(int k);
{       int anfang, ende, summe, i;
        summe = 0;
        anfang = Math.pow(2,n);
        ende = Math.pow(2,k+1) – 1;
        for (int i=anfang; i <= ende; i++) {
                summe = summe +Stern_Brocot(i);
        }
        return ( summe = Math.pow(3,k-1) );
}
```

Punkteverteilungstabellen:

Aufgabe 1:

Teilaufgaben	Summe	Punkte in den Anforderungsbereichen		
		I	II	III
1.1	13	2	10	1
1.2	7	2	3	2
	20	**4**	**13**	**3**
		20%	**65%**	**15%**

Aufgabe 2:

Teilaufgaben	Summe	Punkte in den Anforderungsbereichen		
		I	II	III
2.1	13	4	6	3
2.2	7	1	6	0
	20	**5**	**12**	**3**
		25%	**60%**	**15%**

Aufgabe 3:

Teilaufgaben	Summe	Punkte in den Anforderungsbereichen		
		I	II	III
3.1	11	3	5	3
3.2	9	0	7	2
	20	**3**	**12**	**5**
		15%	**60%**	**25%**

Punkteverteilungstabelle (insgesamt):

Aufgaben	Summe	Punkte in den Anforderungsbereichen		
		I	II	III
1	20	4	13	3
2	20	5	12	3
3	20	3	12	5
	60	**12**	**37**	**11**
		20%	**62%**	**18%**

Bewertungstabelle

Prozent der maximal erreichbaren Roh-punktzahl	Note	Punktzahl
ab 97% bis 100% der max. Punktzahl	sehr gut	15 P
ab 93% bis weniger als 97%		14 P
ab 90% bis weniger als 93%		13 P
ab 85% bis weniger als 90%	gut	12 P
ab 80% bis weniger als 85%		11 P
ab 75% bis weniger als 80%		10 P
ab 70% bis weniger als 75%	befriedigend	09 P
ab 65% bis weniger als 70%		08 P
ab 60% bis weniger als 65%		07 P
ab 55% bis weniger als 60%	ausreichend	06 P
ab 50% bis weniger als 55%		05 P
ab 45% bis weniger als 50%		04 P
ab 38% bis weniger als 45%	mangelhaft	03 P
ab 32% bis weniger als 38%		02 P
ab 25% bis weniger als 32%		01 P
weniger als 25% der max. Punktzahl	ungenügend	00 P

- Ende der Lösungshinweise -