**IN-G 2013 HT Lö**

Saarland
Ministerium
für Bildung und Kultur

## Lösungshinweise: Nur für die Hand der Lehrperson

# Schriftliche Abiturprüfung 2013

| | |
|---|---|
| Fach: | **Informatik** |
| Kurstyp: | G-Kurs |
| Bearbeitungszeit: | 3 Zeitstunden |
| Hilfsmittel: | nicht-programmierbarer Taschenrechner |
| Seitenzahl: | Die Lösungshinweise umfassen mit Deckblatt, Punkteverteilungstabelle und Bewertungstabelle 9 Seiten. |

## 1. Aufgabe

**1.1** (Lehrplan: „Objektorientiertes Modellieren und Programmieren")

Der ergänzte Ablauf lautet:

```
baca
  caab
    aba
      abca
        cabca
          bcaa
            aaab
              abbca
                bcabca
```

**1.2** (Lehrplan: „Objektorientiertes Modellieren und Programmieren")

Java:

```
public String fuehreAus(String wort) {
    band.setzeInhalt(wort);
    while (!band.ende()) {
        char anfang = band.gibZeichen();
        String anhang = programm.anhang(anfang);
        band.aktualisiere(anhang);
    }
    return band.gibInhalt();
}
```

Delphi:

```
function fuehreAus(wort: String): String;
var anfang: char;
    anhang: String;
begin
  band.setzeInhalt(wort);
  while not(band.ende()) do
  begin
    anfang := band.gibZeichen();
    anhang := programm.anhang(anfang);
    band.aktualisiere(anhang);
  end;
  fuehreAus := band.gibInhalt();
end;
```

**1.3** (Lehrplan: „Datentypen und Datenstrukturen“)

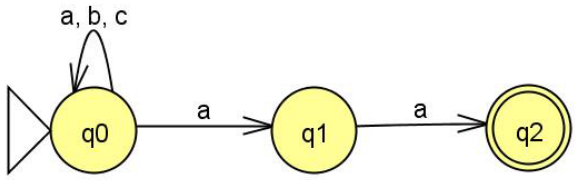
Der abstrakte Datentyp *Schlange* stellt die Operationen `enqueue` und `dequeue` zur Verfügung. Diese können in der Klasse `Speicherband` genutzt werden, um einerseits die beiden Zeichen am 'Anfang' des Bandes zu löschen (zweifaches `dequeue`) und andererseits die Zeichen des Anhangs der Reihe nach am 'Ende' des Bandes einzufügen (ggf. mehrfaches `enqueue`).
Um das Programm beenden zu können, muss die Schlange allerdings zusätzlich eine Möglichkeit bieten, ihre Länge abzufragen.

**1.4** (Lehrplan: „Grenzen der Berechenbarkeit“)

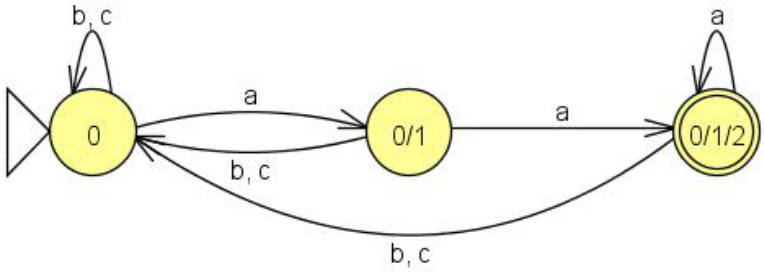
Dass Tag-Maschinen im Allgemeinen dem Halteproblem unterliegen, bedeutet, dass es keinen Algorithmus gibt, der feststellt, ob ein gegebenes Tag-Maschinen-Programm bei einem gegebenen Bandinhalt hält oder nicht.

**1.5** (Lehrplan: „Automaten und formale Sprachen“)

Nicht-deterministischer Übergangsgraph:




Deterministischer Übergangsgraph:




[Die Darstellung der Teilmengenkonstruktion selbst ist abhängig von der im Unterricht verwendeten Darstellung.]

## 2. Aufgabe

**2.1** (Lehrplan: „Strukturiertes Programmieren“)

Java:

```
    public int potenz1(int a, int b) {
        if (b == 0) {
            return 1;
        } else {
            return a * potenz1(a, b - 1);
        }
    }


    public int potenz2(int a, int b) {
        if (b == 0) {
            return 1;
        } else if (b % 2 == 0) {
            return square(potenz2(a, b / 2));
        } else {
            return a * square(potenz2(a, (b - 1) / 2));
        }
    }
```

Delphi:

```
function potenz1(a,b: integer): integer;
begin
  if (b = 0) then
     potenz1 := 1
  else
     potenz1 := a * potenz1(a, b - 1);
end;


function potenz2(a,b: integer): integer;
begin
  if (b = 0) then
     potenz2 := 1
  else
    if (b mod 2 = 0) then
     potenz2 := square(potenz2(a, b div 2))
    else
     potenz2 := a * square(potenz2(a,(b - 1) div 2);
end;
```

**2.2** (Lehrplan: „Algorithmenanalyse“)

Nach dem Aufruf `potenz2(2, 18)` folgen fünf weitere Aufrufe:

```
potenz2(2, 9)
potenz2(2, 4)
potenz2(2, 2)
potenz2(2, 1)
potenz2(2, 0)
```

**2.3** (Lehrplan: „Algorithmenanalyse“)

Die Laufzeit der Methode `potenz1` ist linear bezüglich des Parameters *b*.
Da *b* in jedem Rekursionsschritt um 1 erniedrigt wird, wird die Methode insgesamt (*b* + 1) mal aufgerufen.
[ Landau-Notation: *O(b)* ]

Die Laufzeit der Methode `potenz2` ist logarithmisch bezüglich des Parameters *b*.
Da *b* (bzw. *b*-1) in jedem Rekursionsschritt halbiert wird, wird die Methode insgesamt höchstens $\log_2 b$ mal aufgerufen.
[ Landau-Notation: *O(log b)* ]

**2.4** (Lehrplan: „Strukturiertes Programmieren“ / „Funktionsweise von Computersystemen“)

Java:

```
public int potenz3(int a, int b) {
    int result = 1;
    for (int i = b; i > 0; i--) {
        result = result * a;
    }
    return result;
}
```

Delphi:

```
function potenz3(a,b: integer): integer;
var i: integer;
begin
  potenz3 := 1;
  for i := b downto 1 do
    potenz3 := potenz3 * a;
end;
```

**2.5** (Lehrplan: „Funktionsweise von Computersystemen“)

```
0  JMP 4
1  DEF 0      ; Variable a
2  DEF 0      ; Variable b
3  DEF 1      ; Variable result (a^b)
4  INM 1      ; input a
5  INM 2      ; input b
6  LDA 2      ; Lade b in den Akkumulator
7  JZE 14     ; Falls b = 0: Gehe zu Zeile 14 (Output)
8  DEC        ; Erniedrige b
9  STA 2      ; Speichere b
10 LDA 3      ; Lade result in den Akkumulator
11 MUL 1      ; Multipliziere result mit a
12 STA 3      ; Speichere result
13 JMP 6      ; repeat: Springe zu Zeile 6
14 OUT 3      ; output result
15 END
```

Nach der Deklaration der Variablen *a*, *b* und *result* (Zeilen 1 bis 3) sorgen Zeilen 4 und 5 für die Eingabe der Werte für *a* und *b*. In Zeile 6 beginnt eine Schleife, die abbricht, falls *b* = 0 (Zeile 7). Innerhalb der Schleife wird einerseits *b* um 1 erniedrigt (Zeile 8) und andererseits *result* mit *a* multipliziert (Zeilen 10/11).
Das Ergebnis *result* wird schließlich nach Abbruch der Schleife in Zeile 14 ausgegeben.

## 3. Aufgabe

**3.1** (Lehrplan: „Kryptographie“)

$n = p \cdot q = 17 \cdot 19 = 323$
$\phi(n) = (p - 1) \cdot (q - 1) = 16 \cdot 18 = 288$

**3.2** (Lehrplan: „Kryptographie“)

Schritt 1:
$288 = 12 \cdot 23 + 12$
$23 = 1 \cdot 12 + 11$
$12 = 1 \cdot 11 + 1$
$11 = 9 \cdot 1 + 0$

(d.h. $ggT(288; 23) = 1$)

Schritt 2:
$1 = 12 - 1 \cdot 11$
$= 12 - 1 \cdot (23 - 1 \cdot 12)$
$= 2 \cdot 12 - 1 \cdot 23$
$= 2 \cdot (288 - 12 \cdot 23) - 1 \cdot 23$
$= 2 \cdot 288 - 25 \cdot 23$

Demnach: $d = -25 = 263$ (*mod* 288)

**3.3** (Lehrplan: „Kryptographie“)

$c = m^e \bmod n = 4^{23} \bmod 323 = ((4^{11} \bmod 323)^2 \cdot 4) \bmod 323 = (149^2 \cdot 4) \bmod 323 = 302$

**3.4** (Lehrplan: „Kryptographie“)

i) [ Alice berechnet $A = m^a \bmod p$. Sie sendet $A$ an Bob. ] *(vorgegeben)*
ii) Bob berechnet $B = A^b \bmod p$. Er sendet $B$ an Alice.
iii) Alice berechnet $C = B^{a'} \bmod p$. Sie sendet $C$ an Bob.
iv) Bob berechnet $D = C^{b'} \bmod p$.

**3.5** (Lehrplan: „Kryptographie“)

Im letzten Schritt des Verfahrens berechnet Bob

$$D = C^{b'} mod p = (((m^a)^b)^{a'})^{b'} mod p = m^{a \cdot b \cdot a' \cdot b'} mod p = (m^{a \cdot a'})^{b \cdot b'} mod p = m^{b \cdot b'} mod p = m$$

.

**3.6**

Obwohl die Werte $a$, $a'$, $b$ und $b'$ als 'Schlüssel' bezeichnet werden könnten, müssen Alice und Bob weder einen Schlüssel *austauschen*, noch einen Schlüssel *veröffentlichen*, noch – wie etwa im Diffie-Hellman-Merkle-Verfahren – einen gemeinsamen Schlüssel *generieren*. Daher der Name *'No-Key-Verfahren'*.

## Punkteverteilungstabelle

### Aufgabe 1

| Teilaufgaben | Summe | Punkte in Anforderungsbereichen | | |
|---|---|---|---|---|
| | | I | II | III |
| 1.1 | 3 | 3 | | |
| 1.2 | 6 | | 4 | 2 |
| 1.3 | 3 | | 3 | |
| 1.4 | 2 | | 2 | |
| 1.5 | 6 | | 6 | |
| | **20** | **3** | **15** | **2** |
| | | **15 %** | **75 %** | **10 %** |

### Aufgabe 2

| Teilaufgaben | Summe | Punkte in Anforderungsbereichen | | |
|---|---|---|---|---|
| | | I | II | III |
| 2.1 | 4 | 3 | 1 | |
| 2.2 | 2 | | 2 | |
| 2.3 | 4 | | 2 | 2 |
| 2.4 | 2 | | 2 | |
| 2.5 | 8 | 2 | 4 | 2 |
| | **20** | **5** | **11** | **4** |
| | | **25 %** | **55 %** | **20 %** |

### Aufgabe 3

| Teilaufgaben | Summe | Punkte in Anforderungsbereichen | | |
|---|---|---|---|---|
| | | I | II | III |
| 3.1 | 2 | 2 | | |
| 3.2 | 4 | | 4 | |
| 3.3 | 4 | 2 | 2 | |
| 3.4 | 4 | | 2 | 2 |
| 3.5 | 4 | | 2 | 2 |
| 3.6 | 2 | | 2 | |
| | **20** | **4** | **12** | **4** |
| | | **20 %** | **60 %** | **20 %** |

**Punkteverteilungstabelle (insgesamt)**

| Aufgabe | Summe | Punkte in den Anforderungsbereichen | | |
|---|---|---|---|---|
| | | I | II | III |
| 1 | 20 | 3 | 15 | 2 |
| 2 | 20 | 5 | 11 | 4 |
| 3 | 20 | 4 | 12 | 4 |
| **Summe** | **60** | **12** | **38** | **10** |
| | | **20 %** | **63,33 %** | **16,67 %** |

## 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 -**