Abschlussklausur zu Programmierung 1 im Sommersemester 2013

Geschrieben am 06.09.2013

MUSTERLÖSUNG

Nachname _____

Matrikelnummer _____

Geburtsdatum _____

Studiengang _____

Klausur für 9 Credit Points werten? Ja ☐

Vom Prüfer auszufüllen:

Aufgabe	1	2	3	4	5	6	7	8	9	10	Σ
Punkte	10	10	10	10	10	10	10	10	10	10	100
Davon erreicht											

Klausurpunkte _____

\+ Bonuspunkte _____

Σ _____

Note _____

Generelle Klausurhinweise:

1. Geben Sie auf jedem Blatt (oben rechts) Ihre Matrikelnummer an. Blätter ohne Matrikelnummer können nicht gewertet werden.
2. Schreiben Sie bitte *leserlich*!
3. Kontrollieren Sie Ihre Klausur auf Vollständigkeit. Die Seitenzahlen befinden sich unten rechts.
4. Verwenden Sie die Rückseiten der Klausur ausschließlich für eigene Notizen - diese werden **nicht** gewertet. Die letzte Seite der Klausur ist als *„Schmierpapier“* vorgesehen, oder falls der Platz zum Beantworten einer Frage nicht ausreicht. Zur Benotung muss ein Verweis bei der Aufgabenstellung und eine deutliche Kennzeichnung auf dem Schmierblatt enthalten sein. Benötigen Sie weiteres Papier, melden Sie sich bei der Aufsicht. Selbst mit gebrachtes Papier wird als Täuschungsversuch gewertet!
5. Außer einem dokumentenechten Stift - kein Bleistift - (**nicht** Rot) sind keine weiteren Hilfsmittel zugelassen, wie Handy, Smartphone, Taschenrechner, Laptop etc. Ein betriebsbereites Handy oder Smartphone wird als Täuschungsversuch gewertet.
6. Die Prüflinge können während der Klausur einzeln die Toilette besuchen. Vor Verlassen des Klausurraumes haben diese bei der Aufsicht ihren Namen anzugeben.
7. Für die Bearbeitung der Klausur stehen 180 Minuten zur Verfügung. In der letzten halben Stunde (30 Minuten) vor Abgabe ist es den Prüflingen untersagt den Raum zu verlassen, um unnötige Unruhe zu vermeiden.

Aufgabe 1: **Multiple Choice** Punkte: ____ / 10

Eine Frage gilt dann als korrekt beantwortet, wenn alle richtigen und keine falschen Antworten angekreuzt sind. Mindestens eine Antwort ist immer richtig.

PRG 1.4
Skript V1
Folien V1

(a) (1 Punkt) **Algorithmus**
Welche Eigenschaften besitzt jeder *deterministische* Algorithmus?

- ☒ **Er ist Determiniert**
- ☐ Er kann beliebig viel Speicher anfordern
- ☒ **Zu jedem Ausführungzeitpunkt ist der nächste Schritt eindeutig**
- ☐ Er bricht nach einer endlichen Zeit ab

PRG 5.5
Skript V6
Folien V6

(b) (1 Punkt) **Formale Sprachen**
Welche der folgenden Elemente sind Bestandteil einer formalen Grammatik?

- ☐ Terminalregeln
- ☒ **Produktionsregeln**
- ☒ **Startsymbol**
- ☐ Endsymbole

Skript V1
Folien V1

(c) (1 Punkt) **Rechner-Architektur**
Welcher dieser Befehle führt zum *„Interpretieren der Befehle im Steuerwerk“*?

- ☒ `DECODE`
- ☐ `FETCH OPERANDS`
- ☐ `EXECUTE`
- ☐ `UPDATE INSTRUCTION POINTER`

Skript V3
Folien V3/4

(d) (1 Punkt) **Pythonanalyse**
Was berechnet die Python Funktion `unknown(n)`?

```
1 def unknown(n):
2     if n > 1:
3         return n + unknown(n - 1)
4     return 1
```

- ☐ Die Fakultät von n
- ☐ Die n-te Fibonacci Zahl
- ☒ **Die Summe der Zahlen** 1 **bis** n
- ☐ Die Produktsumme der Zahlen 1 bis n

Skript V3
Folien V3/4

(e) (1 Punkt) **Python Ausdrücke**
Welche der folgenden Ausdrücke werden von Python (Version 3.x) zu `True` ausgewertet?

- ☐ `bool(12 % 6.0)`
- ☒ `bool(1 << 2 << 3)`
- ☒ `bool(False and True or not False)`
- ☐ `bool(not True and 12 * 12 / 12)`

Skript V8
Folien V8

(f) (1 Punkt) **Build-In Datentypen**
Welche Eigenschaften haben Variablen vom Typ *Integer* in Python?

- ☐ Es können maximal 2^{32} unterschiedliche Werte abgebildet werden
- ☐ Es können maximal 2^{64} unterschiedliche Werte abgebildet werden
- ☒ **Sie sind *immutable***
- ☒ **Für die Variablen `i=1` und `j=1` gilt `id(i)==id(j)` ist `True`**

Skript V8
Folien V8

(g) (1 Punkt) **Objektorientierung**
Wie kann in Python aus der Klasse `B` auf die Methode `m` der Basisklasse `A` zugegriffen werden?

- ☐ `super(A, self).m()`
- ☒ `super(B, self).m()`
- ☒ `super().m()`
- ☒ `A.m(self)`

Skript V14
Folien V14

(h) (1 Punkt) **Prozesse**
Woraus besteht der Prozesskontext unter Anderem?

- ☐ Stack
- ☒ **Kernelstack**
- ☐ Programmdaten
- ☒ **Zugriffsrechte**

Skript V14
Folien V14

(i) (1 Punkt) **Synchronisation**
Welche notwendigen, beziehungsweise hinreichenden, Bedingungen müssen nach *Coffman et al.* für eine Verklemmung (*deadlock*) gegeben sein?

- ☐ busy wait
- ☒ **circular wait**
- ☒ **hold and wait**
- ☐ preemptive wait

Skript V15
Folien V15

(j) (1 Punkt) **Algorithmenentwurf**
Welche Technik zeichnet einen *Backtracking*-Algorithmen aus?

- ☐ loop-ahead
- ☐ loop-back
- ☒ **back-track**
- ☐ track-step

Matrikelnummer: _____

Aufgabe 2: **Zahlendarstellung** Punkte: ____ / 10

Folien V3
Skript V3/4

(a) **Hexadezimal ⇔ Oktal**

1. (1 Punkt) Konvertieren Sie die Dualzahl $1110\ 0011_2$ zur Zahlenbasis 8.

Lösung:

1_2	1_2	1_2	0_2	0_2	0_2	1_2	1_2
200_8	100_8	40_8	20_8	10_8	4_8	2_8	1_8

$$= 200_8 + 100_8 + 40_8 + 2_8 + 1_8$$
$$= \underline{\underline{343_8}}$$

2. (1 Punkt) Konvertieren Sie die Hexadezimalzahl $7D9_{16}$ zur Zahlenbasis 10.

Lösung:

$\ldots D_{16}$	C_{16}	B_{16}	A_{16}	9_{16}	8_{16}	$7_{16}\ldots$
$\ldots 13_{10}$	12_{10}	11_{10}	10_{10}	9_{10}	8_{10}	$7_{10}\ldots$

$$= 7_{10} \cdot 16^2 \quad +13_{10} \cdot 16 \quad +9_{10}$$
$$= 7_{10} \cdot 256_{10} \quad +208_{10} \quad +9_{10}$$
$$= 1792_{10} \quad +208_{10} \quad +9_{10}$$
$$= \underline{\underline{2009_{10}}}$$

Folien V3
Skript V3/4

(b) **Dezimal ⇔ Einer/Zweier-komplement**

1. (1 Punkt) Konvertieren Sie die Zahl $1010\ 1010_2$ des Einerkomplementes in eine vorzeichenbehaftete Dezimalzahl.

Lösung:

	128_{10}	64_{10}	32_{10}	16_{10}	8_{10}	4_{10}	2_{10}	1_{10}
	1_2	0_2	1_2	0_2	1_2	0_2	1_2	0_2
$\neg$	0_2	1_2	0_2	1_2	0_2	1_2	0_2	1_2

$$= 64_{10} + 16_{10} + 4_{10} + 1_{10}$$
$$= \underline{\underline{-85_{10}}}$$

2. (1 Punkt) Konvertieren Sie die Dezimalzahl $z = -78_{10}$ in das Zweierkomplement $\overline{z}$ für eine Wortlänge von 8 Bit.

Lösung:

	128_{10}	64_{10}	32_{10}	16_{10}	8_{10}	4_{10}	2_{10}	1_{10}
$z \to z_{bin} =$	0_2	1_2	0_2	0_2	1_2	1_2	1_2	0_2
$\neg z_{bin} =$	1_2	0_2	1_2	1_2	0_2	0_2	0_2	1_2
$\overline{z}_{bin} + 1 =$	1_2	0_2	1_2	1_2	0_2	0_2	1_2	0_2

$\overline{z} = \underline{\underline{1011\ 0010_2}}$

Folien V3
Skript V3/4

(c) (2 Punkte) Welche Dezimalzahl z wird mit folgendem Bitmuster codiert? Die Codierung entspricht dem *IEEE-754* Standard *b16*.
Mit $s = 1$ Bit, $e = 5$ Bit und $m = 10$ Bit. Der Bias ist 16.
Geben Sie den Rechenweg an.

0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0

z = + 5 . 3 7 5

Lösung:
0,5 Punkte pro s, e, m und z

$$s = 0_2 \rightarrow \underline{\mathbf{0_{10}}} \tag{1}$$
$$e = 10010_2 \rightarrow 18_{10} - 16_{10} = \underline{\mathbf{2_{10}}} \tag{2}$$
$$m = 0101100000_2 \rightarrow 1_{10} + \frac{1}{4_{10}} + \frac{1}{16_{10}} + \frac{1}{32_{10}} \tag{3}$$
$$= 1.0_{10} + 0.25_{10} + 0.0625_{10} + 0.03125_{10} = \underline{\mathbf{1.34375_{10}}} \tag{4}$$
$$z = (-1_{10})^s \cdot 2^e \cdot m \tag{5}$$
$$= (-1_{10})^0 \cdot 2_{10}^2 \cdot 1.34375_{10} \tag{6}$$
$$= 1_{10} \cdot 4_{10} \cdot 1.34375_{10} = \underline{+\mathbf{5.375_{10}}} \tag{7}$$

Folien V3
Skript V3/4

(d) (1 Punkt) Welche Dezimalzahl z wird mit folgendem Bitmuster codiert? Die Codierung orientiert sich am *IEEE-754* Standard *b32*.

1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

z = - ∞

Lösung:

$$s = 1_2 \rightarrow (-1_{10})^1 \underline{-\mathbf{1_{10}}} \tag{8}$$
$$e = 11111111_2 \Rightarrow = \underline{\infty \bigvee \text{NaN}} \tag{9}$$
$$m = 00000000000000000000000_2 \rightarrow 1_{10} \Rightarrow = \underline{\infty} \tag{10}$$
$$z = \underline{-\infty} \tag{11}$$

Folien V3
Skript V3/4

(e) (2 Punkte) Geben Sie das Bitmuster der Zahl $z = -112.0$ gemäß *IEEE-754* mit einfacher Genauigkeit (32 Bit) an. **Geben Sie den Rechenweg an.**

1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Lösung:

0,5 Punkte pro s, e, m und Bit-Pattern, bei Rechenfehlern Dezimal → Binär einmalig $0,5$ Punkte Abzug.

$$s = \begin{cases} 1, & \text{wenn negativ,} \\ 0, & \text{wenn positiv} \end{cases}$$

$\Rightarrow s = \mathbf{1}$

$$e = \begin{cases} \lfloor ln_2(|z|) \rfloor & \text{wenn } z \geq 1 \\ \lfloor ln_2(|z|) \cdot |z| \rfloor & \text{wenn } z < 1 \end{cases}$$

$\Rightarrow e = 6_{10} + 127_{10} = 133_{10} \rightarrow \mathbf{10000101_2}$

$m = \frac{|z|}{2^e} = \frac{112.0_{10}}{2^6_{10}} = \frac{112.0_{10}}{64_{10}} = \mathbf{1.75_{10}}$

$\Rightarrow 1.75 = 1.0_{10} + 0.5_{10} + 0.25_{10} = 1 + \frac{1}{2} + \frac{1}{4}$

$\Rightarrow \mathbf{11000000000000000000000_2}$

Folien V3
Skript V3/4

(f) (1 Punkt) Das halblogarithmische Verfahren nach IEEE-754 kompensiert die schlechte Speicherausnutzung im direkten Vergleich zur Festkommarithmetik. Zu welchem Preis wird dieser Vorteil erkauft?

Lösung:

Der Nachteil der halblogarithmischen Verfahren liegt in der Präzison ihrer Darstellung. Zwar findet eine bessere Ausnutzung des gegebenen Speichers statt, jedoch können manche Zahlen nicht mehr exakt repräsentiert werden; selbst bei theoretisch beliebig großem Speicher.

Matrikelnummer: _____

Aufgabe 3: **Elementare Datentypen** Punkte: ____ / 10

Handzettel
Folien V2
Skript V2

(a) (2 Punkte) Gegeben sind die folgenden Python Codezeilen:

```
1 a = 2 * 3 // 42 - 2
2 b = 2 % 42 + 15 / 3
3 c = 4 ^ 4 << 2 >> 4
4 d = 4 <= 1 - 2 & 3
```

Tragen Sie die Operatoren in der Reihenfolge in die Kästchen ein, in der Sie ausgewertet werden. Eine Zeile gilt als richtig gelöst, wenn alle drei Operatoren richtig eingetragen sind.

Der Operator wird als	1.	2.	3.	ausgewertet.
Zeile 1	*	//	-	
Zeile 2	%	/	+	
Zeile 3	<<	>>	^	
Zeile 4	-	&	<=	

Folien V3
Skript V3

(b) (2 Punkte) Erklären Sie in eignenen Worten die Begriffe *dynamische Typisierung* und *statische Typisierung*.

Lösung:
Bei der *dynamischen Typisierung* erfolgt die Typzuweisung einer Variablen zur Laufzeit des Programms, z.B. durch eine Zuweisung.
Bei der *statischen Typisierung* muss zur Übersetzungszeit der Datentyp einer Variablen bekannt sein. Dies erfolgt in der Regel durch die Deklaration.

Folien V3
Skript V3

(c) (2 Punkte) Erklären Sie in eignenen Worten die Begriffe *starke Typisierung* und *schwache Typisierung*.

Lösung:
Bei der *starken Typisierung* bleibt eine einmal durchgeführte Bindung zwischen Variable und Datentyp in jedem Fall bestehen.
Bei der *schwachen Typisierung* kann eine durchgeführte Bindung zwischen Variable und Datentyp „aufgebrochen“ werden.

Folien V3
Skript V3

(d) (1 Punkt) Was verstehen wir in der Informatik unter dem Begriff *„Datentyp“*?

Lösung:
Ein Datentyp in der Informatik ist die Zusammenfassung von Objektmengen mit den darauf definierten Operationen.

Folien V3
Skript V3

(e) (1 Punkt) Worin besteht der Unterscheid zwischen *Casting* und *Coertion*?

Lösung:
Coertion bezeichnet die implizite Typwandlung. *Casting* bezeichnet die explizite Typwandlung. Der Unterschied spiegelt sich daher darin wieder, ob eine Typwandlung explizit angegeben werden muss, oder ob diese implizit vom Compiler bzw. Interpreter vorgenommen wird.

Folien V3
Skript V3

(f) (2 Punkte) Mit welchen Befehlen können in Python (3.x) die folgenden Typkonvertierungen vorgenommen werden?

1. Den Wahrheitswert `x>=y` als String

   1. `str(x>=y)`

2. Der String `s="0xBAADF00D"` als Ganzzahl

   2. `int(s, 16)`

3. Die Oktalzahl `0o177` als Dezimalzahl

   3. `float(0o177)`

4. Den Buchstaben `c = "X"` als ganzzahligen Wert gemäß der ASCII Tabelle

   4. `ord(c)`

Aufgabe 4: **Konstrollstrukturen** Punkte: ____ / 10

(a) (2 Punkte) Formulieren Sie für den folgenden mathematischen Ausdruck eine Funktion `f(x)` in Python. Sie dürfen keine Funktionen aus der Python Klassenbibliothek verwenden.

$$f(x) := \frac{1}{2 \cdot x^3} - 4.5 \cdot \sqrt{x^6 + 7} - \frac{8}{9}$$

Lösung:

```
def f(x):
    return 1/(2*x**3) - 4.5 * (x**6+7)**0.5 - 8/9
```

(b) (1 Punkt) Was ist die Ausgabe des folgenden Python-Skripts:

```
L = []
for i in range(23, 1, -2):
    if i % 2 == 0:
        continue
    L.append(i)
print(L)
```

Lösung:

```
[23, 21, 19, 17, 15, 13, 11, 9, 7, 5, 3]
```

(c) (1 Punkt) Worin besteht der wesentliche Unterscheid zwischen einer `do/while` und einer `while/do` Anweisung?

Lösung:

Bei der `do/while`-Anweisung findet die Überprüfung des Abbruchkriteriums erst nach Durchlaufen des Schleifenrumpfes statt. Der Schleifenrumpf wird daher mindestens einmal ausgeführt.
Bei der `while/do`-Anweisung findet die Überprüfung vor Betreten des Schleifenrumpfes statt. Dieser muss daher nicht zwingend ausgeführt werden.

(d) (3 Punkte) Implementieren Sie die Funktion `hanoi(n, source, drain, temp)` zum Lösen der „Türme von Hanoi“.
Dabei bezeichnen `source`, `drain` und `temp` die drei Stapel. Mit `n` wird angegeben, wie viele Scheiben sich auf der Position `source` befinden.

Lösung:

```
def hanoi(n, source, drain, temp):
    if n==1:
        print("Move Slice from '" +source+ "' to '" +drain)
    else:
        hanoi(n-1, source, temp, drain)
        print("Move Slice from '" +source+ "' to '" +drain)
        hanoi(n-1, temp, drain, source)
```

(e) Gegeben sei folgendes Programm:

```
10 a = 2; b = 3;
20 if b > 2 goto 50
30 a = a-1
40 b = b+1; goto 20
50 if a>0 goto 30
60 print b
```

1. (1 Punkt) Welche Ausgabe erzeugt das Programm?

   Lösung:
   5

2. (2 Punkte) Schreiben Sie das Programm in Python neu.

   Lösung:

```
a = 2; b = 3
while b > 2 and a > 0:
    a -= 1
    b += 1
print(b)
```

Matrikelnummer: _____

Aufgabe 5: **Objektorientierung** Punkte: ____ / 10

Folien V8
Skript V8

(a) (1 Punkt) Worin besteht der wesentliche Unterscheid zwischen der objektorientierten und der prozeduralen Progammierung?

Lösung:
Der wesentliche Unterscheid besteht darin, dass in der objektorientierten Programierung Daten und Anweisungen zu „logischen“ Einheiten zusammengefasst sind, wärend sie im Kontext der prozeduralen Progammierung nebeneinander existieren.

Folien V8
Skript V8

(b) (1 Punkt) Was besagt das Geheimnisprinzip?

Lösung:
Das Geheimnisprinzip besagt, das der Benutzer vom Innenleben eines Objektes (Programmierer des aufrufenden Objekts) nichts wissen muss.

Folien V8
Skript V8

(c) (4 Punkte) Welche Arten der Sichtbarkeit (Zugriffskontrolle) gibt es in der Objektorientierung?
Welche Bedeutung haben diese jeweils?

Lösung:

- `public +`: Zugreifbar für alle Ausprägungen
- `private -`: Nur für Ausprägungen der eigenen Klasse zugreifbar
- `protected #`: Nur für Ausprägungen der eigenen Klasse und von Spezialisierungen der selben zugreifbar
- `package ~`: Erlaubt den Zugriff für alle Elemente innerhalb des eigenen Pakets

Folien V8
Skript V8

(d) (1 Punkt) Python unterstützt Mehrfvachvererbung. Mit welchem Konstrukt ist es möglich in Programiersprachen ohne Mehrfvachvererbung eine ähnliche Flexibilität zu erreichen?

Lösung:
Die Alternative zur Mehrfachvererbung ist das Konstrukt der Schnittstelle (Interface).

Folien V8
Skript V8

(e) (1 Punkt) Erklären Sie was mit „Operator überladen“ gemeint ist.

Lösung:
Das Überladen von Operatoren folgt dem Prinzip des Überladens einer Methode. Auf diese Weise ist es möglich, dass Methoden mit gleichem Name gleichzeitig existieren. Die Auswahl, welche dieser Methoden augeführt wird, entscheidet sich zur Laufzeit anhand des Kontexts.

Folien V8
Skript V8

(f) (2 Punkte) Was wird mit dem Begriff *Reflexion* bezeichnet?
Sind Programmierspachen die über Reflexion verfügen mächtiger als solche die kein Reflexion besitzen? **Begründen Sie ihre Antwort.**

Lösung:
Reflexion bedeutet, dass ein Programm Erkenntnisse über seine eigene Struktur gewinnen kann. Sie ermöglicht es, zur Laufzeit Informationen über Klassen oder deren Instanzen abfragen zu können. Die Mächtigkeit einer Programmierspachen ist unabhänig davon, ob sie reflexives programmieren erlaubt. Die Turing-Vollständigkeit ist hier das Mass der Dinge.

Aufgabe 6: **Aggregierte Datentypen** Punkte: ____ / 10

Folien V7

(a) (1 Punkt) Was verstehen wir unter dem Begriff *aggregierter Datentyp*?

Lösung:
Aggregierte Datentypen sind Datenkonstrukte, welche aus einfachen oder zusammengesetzten Datentypen bestehen.

Folien V7

(b) (2 Punkte) Was ist der Unterscheid zwischen einer Liste und einem Array? Wie sieht ihr Speicherverbrauch aus?

Lösung:
Ein Array, oder Feld, bezeichnet eine fest allokierte Größe des Speichers für einen zugrunde liegenden Datentyp.
Eine Liste bezeichnet hingegen eine dynamische Struktur eines beliebigen Typs. Der reale Speicherbedarf einer Liste kann variieren, während ein Array immer eine feste Größe besitzt.

Folien V7

(c) (2 Punkte) Für Wörterbücher ist eine effiziente Implementierung unerlässlich. Auf welche Datenstruktur würden Sie zurückgreifen, wenn Sie ein effizientes Wörterbuch selbst implementieren wollen?

Lösung:
Die Zeit zum Auffinden eines Eintrages ist entscheidend für die Effizientz eines Wörterbuchs. Je mehr Einträge ausgeschlossen werden können, um so schneller ist die Suche erfolgreich. Balancierte Bäume bieten sich an, da sie mit jedem Vergleich 50% des Suchraumes ausschließen, was zu einer logarithmischen Laufzeit führt. Alternativ bietet sich das Hashing an, bei dem auf der Grundlage des Schlüssels die „Speicherposition“ berechnet wird.

Folien V7

(d) (1 Punkt) In anderen Programmiersprachen gibt es die Möglichkeit Daten in einem Verbund zu strukturieren. Python kennt jedoch keinen solchen Verbund. Wie kann dennoch die gleich Funktionalität in Python erreicht werden?

Lösung:

Durch die Verwendung einer Klasse (`class`). Selbst in den Programmiersprachen, die dieses Konstrukt anbieten, ist ein struct im Grunde auch nur eine Klasse in der - per default - alle Member public sind.

Folien V7

(e) (1 Punkt) Was bedeutet *Dereferenzieren* im Zusammenhang mit Zeigern?

Lösung:

Zeiger erlauben es dynamische Strukturen aufzubauen. Auf einen Zeiger kann entweder „indirekt“ zugegriffen werden; also auf den Speicher auf den der Zeiger verweist. Oder *Dereferenziert* auf das Datum des Zeigers selbst.

Folien V7

(f) (1 Punkt) Wie ist es in Python möglich eine dynamische Datenstruktur aufzubauen, die sinngemäß dem Vorbild der Zeiger folgt?

Lösung:

Durch importieren von `ctyps` ☺, oder durch eine Klasse mit mindesten einem Attribut, das als Refrenz dient und vom Typ der Klasse selbst ist.

Folien V7

(g) (2 Punkte) Implementieren Sie eine rekursive Funktion `deep_copy(data)`, die eine tiefe Kopie von `data` zurückgibt, ohne das Modul `copy` zu importieren. Gehen Sie davon aus, dass `type(data) == type([])` gilt.

Lösung:

```
1 def deep_copy(data):
2     if type(data) != type([]):
3         return data
4     copy = [];
5     for item in data:
6         copy.append(deep_copy(item))
7     return copy
8     #return [deep_copy(item) for item in data]
```

Aufgabe 7: **Daten-Informationen-Wissen** Punkte: ____ / 10

Folien V6

(a) **Syntaxdiagramm & EBNF**

```
identifier ::= (letter|"_") (letter | digit | "_")*
letter ::= lowercase | uppercase
lowercase ::= "a"..."z"
uppercase ::= "A"..."Z"
digit ::= "0"..."9"
```

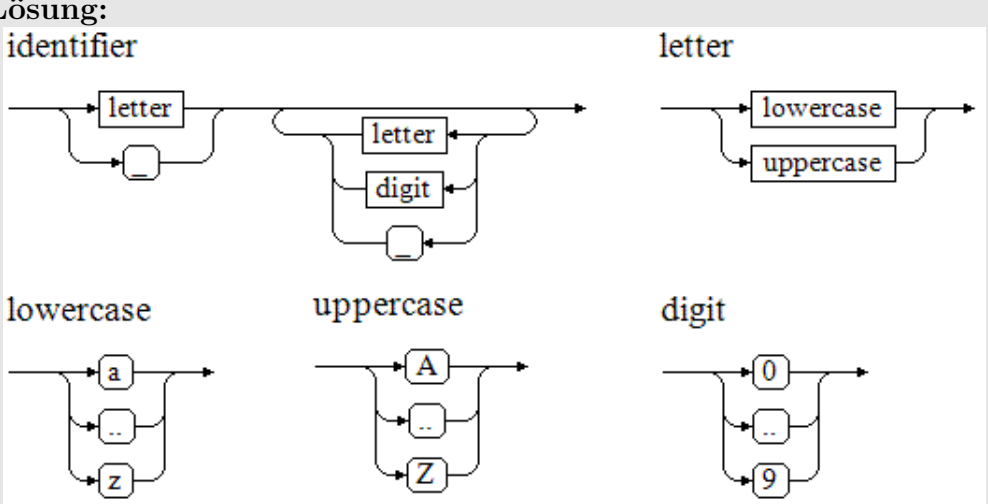
1. (2 Punkte) Wandeln Sie die oben gebene formale Notation für die Erzeugung eines Python Bezeichners in EBNF um.

Lösung:

```
identifier = (letter | '_') {letter | digit | '_'}.
letter = lowercase | uppercase.
lowercase = 'a'|' .. '|'z'.
uppercase = 'A'|' .. '|'Z'.
digit = '0'|' .. '|'9'.
```

2. (2 Punkte) Stellen Sie die EBNF zusätzlich als Syntaxdiagramm dar.

Lösung:



Folien V6

(b) (3 Punkte) Was beschreiben die folgenen drei Begriffe jeweils?

1. Syntax

Lösung:
Regelwerk zur Wort oder Satzbildung

2. Semantik

Lösung:
Bedeutung

3. Pragmatik

Lösung:
Anwendungskontext

Folien V6

(c) (3 Punkte) Gegeben sei die folgende 32 Zeichen lange Nachricht:

`HESSE_LACHT_ZUR_FASSENACHT_HELAU`

Gehen Sie im Folgenden davon aus, dass alle Zeichen des zugrunde liegenden Alphabets in der Nachricht vorkommen, und dass die Nachricht repräsentativ für die Auftrittswahrscheinlichkeit eines jeden Zeichens ist.
Wie groß ist der Informationsgehalt (Bit) der folgenden Zeichen?
Geben Sie auch den Rechenweg an!

1. `H`

Lösung:
$-ld(4/32)\text{Bit} = -ld(1/8)\text{Bit} = 3\text{Bit}$

2. `T`

Lösung:
$-ld(2/32)\text{Bit} = -ld(1/16)\text{Bit} = 4\text{Bit}$

3. `N`

Lösung:
$-ld(1/32)\text{Bit} = 5\text{Bit}$

Aufgabe 8: **Prozesse & Synchronisation** Punkte: ____ / 10

Folien V14 (a) (2 Punkte) Wie lauten die vier Strategien zur Lösung einer Verklemmung?

Lösung:
- das Problem **ignorieren**
- die Verklemmungen **erkennen und beseitigen**
- die Verklemmungen **vermeiden**
- die Verklemmungen **unmöglich machen**

Folien V14 (b) (1 Punkt) Was sind die Vorteile, was sind die Nachteile der *lightweight Threads*?

Lösung:
- **Vorteil**: sehr schneller thread-Wechsel
- **Nachteil**: Blockieren aller threads bei I/O-Warten von einem thread

Folien V14 (c) (1 Punkt) Was ist ein *kritischer Abschnitt*?

Lösung:
Ein Programmabschnitt, der Schreibzugriff auf eine globale Variable hat.

Folien V15 (d) (1 Punkt) Was ist eine *atomare Aktion*?

Lösung:
Eine Folge von Anweisungen, die vollständig oder garnicht ausgeführt werden

(e) (2 Punkte) Wie kann es zu einer Verletzung der *fairness-Bedingung* kommen? **Geben Sie ein Beispiel an.**

Folien V14

Lösung:
Spin locks können die *fairness-Bedingung* verletzen. Wenn beispielsweise ein Prozess mit hoher Priorität gerade dann die CPU erhalten hat, wenn ein Prozess mit niedriger Priorität in seinem kritischen Abschnitt ist. In diesem Fall wird der Prozess mit hoher Priorität so lange im spin lock verbleiben, bis der Prozess mit niederer Priorität den kritischen Abschnitt verlässt. Was aufgrund der geringenen Priorität nie der Fall sein wird.

Folien V14

(f) Das **Erzeuger-Verbraucher** Problem

1. (1 Punkt) Wie viele Semaphoren werden zu Puffersynchronisation des Erzeuger-Verbraucher-Problems benötigt?

   Lösung:
   Es werden 3 Semaphoren benötigt (`frei`, `belegt` und `mutex`)

2. (2 Punkte) Ergänzen Sie die Semaphoren anhand des Codebeispiels zum Lösen des Erzeuger-Verbraucher Problems.

   Lösung:

```
while True: # Erzeuger–Thread
    produce(item)
    frei.acquire()
    mutex.acquire()
    putInBuffer(item)
    mutex.release()
    belegt.release()
```

```
while True: # Verbraucher–Thread
    belegt.acquire()
    mutex.acquire()
    getFromBuffer(item)
    mutex.release()
    frei.release()
    consume(item)
```

Matrikelnummer: _____

Aufgabe 9: **Algorithmenentwurf** Punkte: ____ / 10

Folien V15
Skript V15

(a) 1. (1 Punkt) Was berechnet der folgende Algorithmus?

```
sets = [[v] for v in g.getvertices()]
X = Graph()
for e in g.getedges():
    v1, v2 = e[0][0], e[0][1]
    s1, s2 = findset(sets, v1), findset(sets, v2)
    if s1 != s2:
        X.addedge(v1,v2,e[1])
        joinset(sets,s1,s2)
    if len(sets) == 1:
        break
```

Lösung:
Den minimalen Spannbaum (MST) für g.

2. (1 Punkt) Welchem Entwurfsmuster ist er zuzuordnen?

Lösung:
Es handelt sich um einen Greedy-Algorithmus.

Folien V15
Skript V15

(b) 1. (3 Punkte) Formulieren Sie in Worten oder Pseudocode eine funktion die mit *logaritmischer* Komplexität nach der Divide & Conquer Methode feststellt, ob ein bestimmter Wert `x` in einer Liste `L` enthalten ist.

Lösung:

```
def contains(x, L):
    if len(L) == 1:
        return L[0] == x
    m = len(L) / 2
    if x >= L[m]:
        return contains(x, L[m:])
    return contains(x, L[:m])
```

2. (1 Punkt) Welche Vorraussetzung muss für die Liste `L` gegeben sein, damit Ihr Algorithmus korrekt arbeitet?

Lösung:
Die Liste muss sortiert sein.

Skript V15

(c) (1 Punkt) Wie ist die asymtotische Laufzeit für *Mergesort* bei eine Eingabe der Größe n abzuschätzen?

Lösung:
Die Laufzeit ist in $\mathcal{O}(n \cdot ld(n))$

Skript V15

(d) (1 Punkt) Was muss gegeben sein, um die *kombinatorische Explosion* bei Backtracking-Algorithmen zu verhindern?

Lösung:
Um die kombinatorische Explosion zu verhindern, ist es von Vorteil, den Suchraum so weit wie möglich auf eine Teilmenge einzuschränken.

Skript V15

(e) 1. (1 Punkt) Warum kann es vorkommen, dass ein Greedy-Algorithmus nicht das *globale Optimum* findet?

Lösung:
Greedy-Algorithmen „planen“ ihren nächsten Schritt auf der Grundlage einer lokalen Analyse ihrer Umgebung. Was dazu führen kann, dass sie in einem *lokalem Optimum* festsitzen, da jeder weitere Schritt im Vergleich zur aktuellen Position ein Verschlechterung darstellt.

2. (1 Punkt) Wie können wir die Güte eines Greedy-Algorithmus verbessen, so dass dieser bessere Chance hat, das *globale Optimum* zu finden?

Lösung:
In dem wir nicht nur in der lokalen Umgebung nach einer Optimierung suchen, sondern den Einzugsbereicher Nachbarschaftsanalyse vergrößern.

Aufgabe 10: **Bäume** Punkte: ____ / 10

Der Heap ist eine abstrakte Datenstruktur zur Verwaltung von Daten. Ein Array $a[0 \ldots n-1]$ heißt *Min-Heap* wenn auf den Elementen eine Ordnung $\leq$ definiert ist und für alle Indizes $0 \leq i < n$ gilt:

$$a[i] \leq a[2 \cdot i + 1] \text{ und } a[i] \leq a[2 \cdot i + 2]$$

(a) (1 Punkt) Überprüfen Sie das Array A und erklären Sie warum es sich nicht um einen *Min-Heap*, nach der oben genannten Definition, handeln kann.

```
A = [7, 10, 25, 23, 19, 99, 36, 17, 39]
```

Lösung:

Das Array A ist kein Heap, da $17 < 23$ ist.

(b) (1 Punkt) Welche Elemente müssen getauscht werden, damit das Array zu einem *Min-Heap* wird?

Lösung:

Es genügt, die beiden Wert 17 und 23 zu vertauschen.

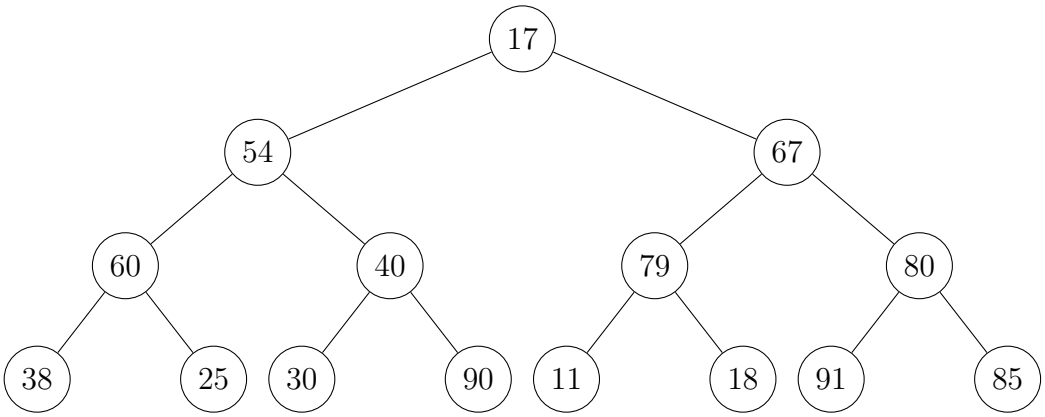
(c) (4 Punkte) Wenden Sie den folgenden Algorithmus auf das Array `A` an und notieren Sie die Ausgaben der `print`-Anweisung aus Zeile 20.

```
1  A = [11, 10, 23, 42, 11, 27, 15]
2  def pushdown(i):
3      n = len(A)
4      left = 2 * i + 1
5      right = 2 * i + 2
6      if left < n and A[left] > A[i]:
7          temp = left
8      else:
9          temp = i
10
11     if right < n and A[right] > A[temp]:
12         temp = right
13
14     if temp != i:
15         A[i], A[temp] = A[temp], A[i]
16         pushdown(temp)
17
18 for i in range(len(A) // 2, -1, -1):
19     pushdown(i)
20     print(A)
```

Lösung:

```
[11, 10, 23, 42, 11, 27, 15]
[11, 10, 27, 42, 11, 23, 15]
[11, 42, 27, 10, 11, 23, 15]
[42, 11, 27, 10, 11, 23, 15]
```

(d) Notieren Sie, in welcher Reihenfolge die Knoten des nachstehenden Baumes bei den jeweiligen Traversierungen besucht werden, indem Sie die Folge der Knotennummern aufschreiben.



1. (1 Punkt) Preorder:

   Lösung:
   `38 60 25 54 30 40 90 17 11 79 18 67 91 80 85`

2. (1 Punkt) Inorder:

   Lösung:
   `17 54 60 38 25 40 30 90 67 79 11 18 80 91 85`

3. (1 Punkt) Postorder:

   Lösung:
   `38 25 60 30 90 40 54 11 18 79 91 85 80 67 17`

4. (1 Punkt) Levelorder:

   Lösung:
   `17 54 67 60 40 79 80 38 25 30 90 11 18 91 85`