Abschlussklausur zu Programmierung 1 im Wintersemester 2014

Geschrieben am 07.02.2014

MUSTERLÖSUNG

Vorname _____

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, Smartwatch, 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

Hinweis: *Die folgenden Fragen sind Multiple-Choice-Fragen, d.h. mehrere Alternativen können richtig sein. Jede Frage erbringt einen Punkt. Es müssen alle richtigen Alternativen und keine falsche ausgewählt sein.*

(a) (1 Punkt) **Formale Sprachen**

Gegeben sei die folgende Grammatik $\mathcal{G}$ in *EBNF* (gemäß ISO 14977) mit dem Startsymbol `S`:

```
S = {A}.
A = A | B | 'a'.
B = AB | B | 'b' | 'ab'.
```

Für welche der folgenden Wörter w gilt $w \in \mathcal{G}$?

- ☒ ε **(das leere Wort)**
- ☒ **BaBBabAB**
- ☒ **bAbbABab**
- ☐ ABAadABA

(b) (1 Punkt) **Pythonanalyse**

Welches Ergebnis liefert die folgende Funktion?

```
1 def unknown(a):
2     for i in range(2, 1 // int(a**0.5)):
3         if a % i == 0:
4             return i
5     return a
```

- ☐ Die Wurzel von `a` für eine Zahl ≥ 2
- ☐ Den Kehrwert des Quadrats für eine positive Zahl `a`
- ☐ Einen ganzzahligen Teiler von `a`
- ☒ **Etwas ganz anderes**

(c) (1 Punkt) **Python Ausdrücke**

Welche der folgenden Ausdrücke werden von Python (Version 3.x) zu `False` ausgewertet?

- ☒ `bool(1 // 2 ** 3 % 4)`
- ☐ `True if not False else not False`
- ☒ `bool(3>>2>>1)`
- ☐ `bool(set([1, 2]) - set([2, 3]))`

(d) (1 Punkt) **Built-in Datentypen**

Über welchen primitiven *Built-in* Datentypen verfügt Python?

- ☐ Array
- ☒ **Liste**
- ☐ Record
- ☐ Struct

(e) (1 Punkt) **Wörterbuch**
Was gilt für die Schlüssel eines Wörterbuches?

- ☐ Die Schlüssel sind `mutable`
- ☒ **Die Schlüssel sind `unmutable`**
- ☒ **Sie dürfen Tuple enthalten**
- ☒ **Sie nicht aus Strings bestehen, da diese `mutable` sind**

(f) (1 Punkt) **Funktionen**
Was ist streng genommene nicht Bestandteil der Signatur einer Funktion

- ☐ Der Name
- ☐ Die Reihenfolge der Parameter
- ☐ Die Typen der Parameter
- ☒ **Der Rückgabewert**

(g) (1 Punkt) **Objektorientierung**
Gewinnt ein Programm Erkenntnis über seine eigene Struktur, so reden wir von

- ☐ Intension
- ☐ Initiation
- ☒ **Introspektion**
- ☐ Interaktion

(h) (1 Punkt) **Welche der folgenden Deklarationen macht in Python die Variable `x` pseudo-privat?**
Ein *Prozess* besteht unter anderem aus

- ☐ `private x = 1`
- ☐ `x__ = 1`
- ☐ `x 1`
- ☒ **`_x = 1`**

(i) (1 Punkt) **Abstrakte Datentypen**
Welche Eigenschaften muss ein abstrakter Datentyp aufweisen?

- ☒ **Präzise Beschreibung**
- ☐ Vererbbarkeit
- ☐ Entscheidbarkeit
- ☒ **Kapselung**

(j) (1 Punkt) **Synchronisation**
Wie lässt sich ein kritischer Abschnitt schützen?

- ☒ **Mit Semaphoren**
- ☒ **Durch abschalten der Interrupts**
- ☐ Auf ein Monoprozessorsystemen gibt es keine kritischen Abschnitte
- ☒ **Mit Hilfe von atomaren Aktionen**

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

(a) (2 Punkte) Vervollständigen Sie die nachstehende Tabelle gemäß der in der Tabelle gegebenen Zahlenbasen.

Binär	Hexadezimal
0100 1100 0000 1111	4C0F
0010 1110 1101	2ED

(b) (2 Punkte) Vervollständigen Sie die nachstehende Tabelle gemäß des Zweierkomplementes für eine Wortlänge von 8 Bit.

2er-Komplement	Dezimal
0110 1001	105
1010 1010	-86

(c) (2 Punkte) Was ist der Unterschied zwischen dem Einerkomplement und dem Zweierkomplement? Geben Sie ein Beispiel an.

Lösung:

Im Einerkomplement gibt es eine positive und eine negative 0 (± 0) im Zweierkomplement ist die Null eindeutig.
Beispiel: Dezimal 5 ist binär 0101. Das Einer-Komplement ist 1010 für -5. Addieren wir beides, so ergibt sich 1111. Damit sich 0 (mit Übertrag) ergibt, muss das Komplement eins größer sein: 1011.

(d) (2 Punkte) Welche Dezimalzahl z wird mit folgenden Bitmuster gemäß *IEEE-754* Codiert? Geben Sie den Rechenweg an.

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

Lösung:

$s = 1_2 \rightarrow \underline{-1_{10}}$
$e = 10000011_2 \rightarrow 131_{10} - 127_{10} = \underline{2_4}$
$m = 01011000000000000000000_2 \rightarrow 1_{10} + \frac{1}{4}_{10} + \frac{1}{16}_{10} + \frac{1}{32}_{10} = 1.0_{10} + 0.25_{10} + 0.0625_{10} + 0.03125_{10} = \underline{1.34375_{10}}$
$z = s \cdot 2^e \cdot m = -1_{10} \cdot 2^4_{10} \cdot 1.34375_{10} = -16_{10} \cdot 1.34375_{10} = \underline{-21.5_{10}}$

(e) (2 Punkte) Geben Sie das Bitmuster und den Rechenweg der Zahl $z = 6.875$ gemäß *IEEE-754* mit einfacher Genauigkeit an.

Lösung:

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

Für jeden Schritt 0,5 Punkte. Bei Rechenfehlern Dezimal → Binär einmalig 0,5 Punkte Abzug.

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

$e = \lfloor ln_2(|z|) \rfloor = 4_{10}$

$\Rightarrow 4_{10} + 127_{10} = 131_{10} \rightarrow 10000011_2$

$m = z/2^e = 21.5_{10}/16_{10} = 1.34375_{10}$

$\Rightarrow 1.34375 = 1.0_{10} + 0.25_{10} + 0.0625_{10} + 0.03125_{10} =$

$1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{32} \Rightarrow 01011000000000000000000_2$

Aufgabe 3: **Python-Datentypen** Punkte: ____ / 10

Gegeben seien die folgenden Python Codezeilen:

```
1 a = (2 << 1) / 3 + 2
2 b = 2 ** 3 - 4 * 2.0
3 c = 5 & 3 * (4 >> 2)
4 d = 3 / 2 + 2 ==  3 / 2.0 + 2
```

(a) (2 Punkte) Geben Sie die Auswertungsreihenfolge der Operatoren von links (zuerst) nach rechts (zuletzt) an.

Lösung:

Auswertungsreihenfolge der Operatoren aus Zeile 1	`<<`	`/`	`+`
Auswertungsreihenfolge der Operatoren aus Zeile 2	`**`	`*`	`-`
Auswertungsreihenfolge der Operatoren aus Zeile 3	`>>`	`*`	`&`
Auswertungsreihenfolge der Operatoren aus Zeile 4	`/`	`+`	`==`

(b) (2 Punkte) Welche Ausgabe erzeugt der folgende Befehl in Bezug auf die Variablen `a`, `b`, `c` und `d` aus der vorherigen Teilaufgabe?
`print(a, b, c, d, end="\t")`

Lösung:

```
1 3.333333333333333   0.0     1   True
```

(c) (2 Punkte) Vervollständigen Sie die folgenden Aussagen:

1. Wie viele Zustände lassen sich mit n-Bit kodieren?

   1. 2^n

2. Wie lautet die größte positive Ganzzahl die sich mit n-Bit darstellen lässt?

   2. $2^n - 1$

3. Wie werden *veränderliche* Datentypen in Python genannt?

   3. **mutable**

4. Zählt der Python-Datentyp `str` zu den veränderliche Datentypen?

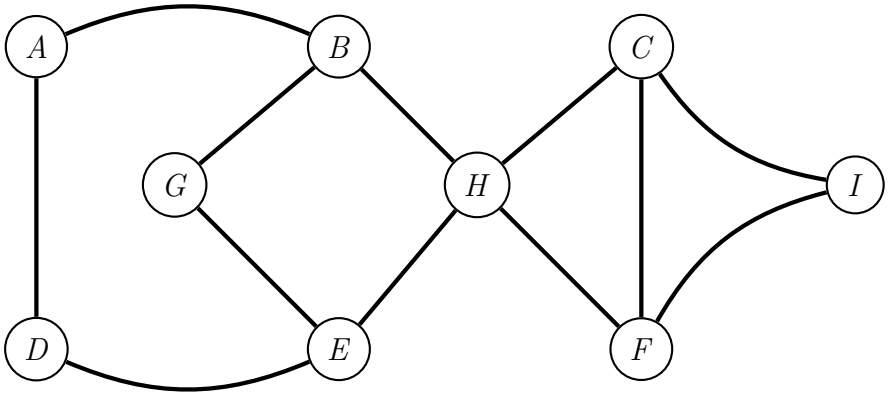
   4. **nein**

(d) (1 Punkt) Vervollständigen Sie die `print`-Anwseisung, mittels *slicing*, so dass der Text `'eipmacscL'` ausgegeben wird?

```
1 s = "Lachsschaumspeise"
2 print(s[17::-2])
```

(e) (3 Punkte) Rekonstruieren Sie den Graphen G anhand der Adjazenzliste `L` für die Knoten `A`, `B`, `C`, `D`, `E`, `F`.

```
A = [ [A, [B, D]],
      [B, [A, G, H]],
      [C, [F, H, I]],
      [D, [A, E]],
      [E, [D, G, H]],
      [F, [C, H, I]],
      [G, [B, E]],
      [H, [B, C, E, F]],
      [I, [C, F]] ]
```



Lösung:
$\frac{1}{4}$ **Punkt pro Kante**

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

(a) (1 Punkt) Überführen Sie die Ackermannfunktion

$$ack(n,m) = \begin{cases} m+1 & , n = 0 \\ ack(n-1,1) & , m = 0 \wedge n \neq 0 \\ ack(n-1, ack(n, m-1)) & \end{cases}$$

in eine rekursive Python Funktion.

```
def ack(n, m):
if n==0:
    return m + 1
elif m==0:
    return ack(n - 1, 1)
else:
    return ack(n - 1, ack(n, m - 1))
```

(b) (1 Punkt) Geben Sie eine iterative Implementierung der folgenden Funktion an.

$$f(n) = \begin{cases} n & , n \leq 1 \\ f(n-1) + f(n-2) & \end{cases}$$

```
def f(n):
    last, temp = 0, 1
    while n > 0:
        n -= 1
        last, temp = temp, last + temp
    return last
```

(c) (1 Punkt) Überführen Sie den Python-Code in mathematische, Notation.

```
def pell(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return 2 * pell(n - 1) + pell(n - 2)
```

Lösung:

$$pell(n) = \begin{cases} 0, & \text{wenn } n = 0, \\ 1, & \text{wenn } n = 1, \\ 2 \cdot pell(n-1) + pell(n-2), & \text{sonnst} \end{cases}$$

oder wie folgt, aber nur wenn $n \in \mathbb{N}_0^+$ mit angeben wird

$$pell(n) = \begin{cases} n, & \text{wenn } n < 2, \\ 2 \cdot pell(n-1) + pell(n-2), & \text{sonnst} \end{cases}$$

(d) (1 Punkt) Über welchen **Schleifentyp** verfügt Python nicht?

Lösung:
Nachprüfende oder Fußlastig Schleifen

(e) (2 Punkte) Welche Ausgabe erzeugt folgendes Programm?

```
1  a = 6
2  def foo(bar=1):
3      return bar ** 3
4
5  def bar(b):
6      global a
7      if a<b:
8          a -= 1
9          return foo(b)
10     else:
11         b = b + 1
12         return a + b
13
14 for i in range(5):
15     print(bar(i), end='+')
```

Lösung:

```
1  7+8+9+10+11+
```

(f) (1 Punkt) Beantworten Sie die folgenden Fragen.

1. Mit welchem Schlüsselwort kann die Ausführung einer Schleife - in Python - übersprungen werden?
   `continue`
2. Wie lautet das Python-Äquivalent für eine `switch-/case`-Anweisung?
   `if/elif`

(g) (3 Punkte) **Merge-Sort**:
Merge-Sort ist ein vergleichsbasiertes Sortierverfahren dessen Grundannahme darauf aufbaut, dass eine einelementige Menge sortiert ist. In der *Merge-Phase* (`merge(A, B)`) werden zwei Listen `A`, `B` zu einer sortierten Liste zusammengesetzt.

```
1  def merge(A, B):
2      merged_list = []
3      while len(A) > 0 and len(B) > 0:
4          if A[0] < B[0]:
5              merged_list.append(A.pop(0))
6          else:
7              merged_list.append(B.pop(0))
8      merged_list.extend(A)
9      merged_list.extend(B)
10     return merged_list
```

Implementieren Sie - unter Verwendung der Funktion `merge(A, B)` - die **rekursive** Funktion `merge_sort(A)`, so das diese die übergebene Liste `A` sortiert.

Lösung:

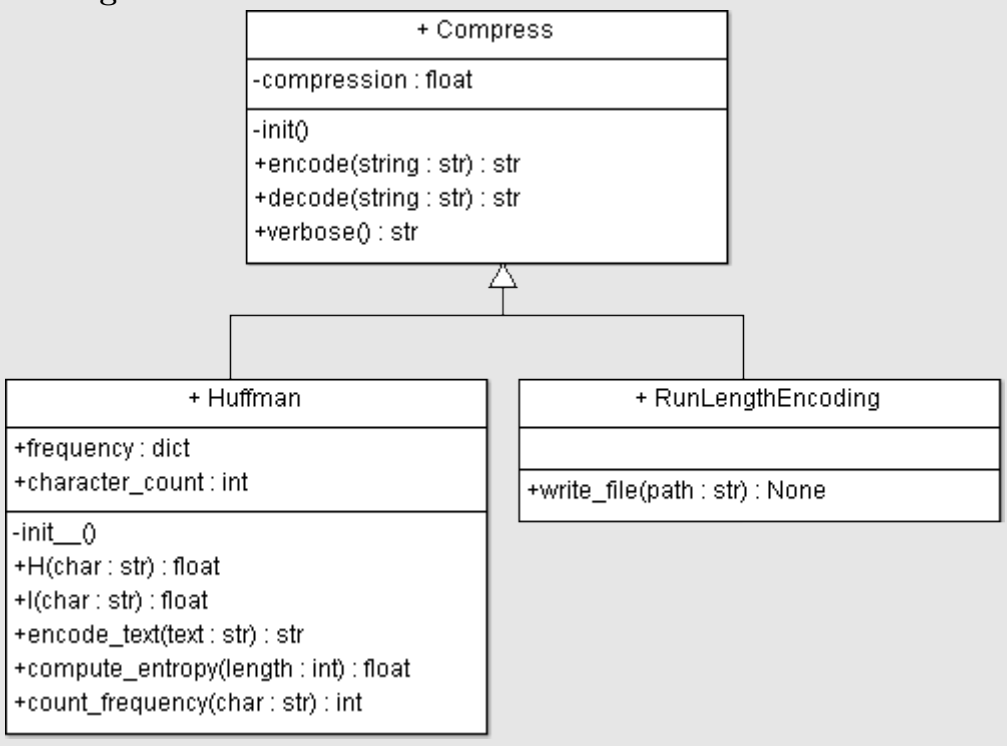
```
1  def merge_sort(A):
2      if len(A) <= 1:          ## oder len(A) < 2
3          return A
4      m = len(A) // 2          ## wichtig //
5      l = merge_sort(A[:m])    ## oder A[0:m]
6      r = merge_sort(A[m:])    ## oder A[m:len(A)]
7      return merge(l, r)
```

Aufgabe 5: **OOP & OOAD** Punkte: ____ / 10

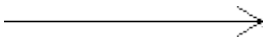
(a) (3 Punkte) Erstellen Sie aus dem gegebenen Python-Code das zugehörige UML-Klassendiagramm.

```
1  class Compress(object):
2      def __init__(self):
3          self._compression = 0.0
4      def encode(self, string:str) -> str:
5          pass
6      def decode(self, string:str) -> str:
7          pass
8      def verbose(self) -> str:
9          pass
10
11 class Huffman(Compress):
12     def __init__(self):
13         Compress.__init__(self)
14         self.frequency, self.character_count = {}, 0
15     def H(self, char:str) -> float:
16         pass
17     def I(self, char:str) -> float:
18         pass
19     def encode_text(self, text:str) -> str:
20         pass
21     def compute_entropy(self, length:int) -> float:
22         pass
23     def count_frequency(self, char:str) -> int:
24         pass
25
26 class RunLengthEncoding(Compress):
27     def __init__(self):
28         Compress.__init__(self)
29     def write_file(self, path=:str) -> None
30         pass
```

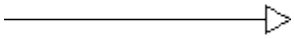
Lösung:



(b) (5 Punkte) Benennen Sie die folgenden Elemente der UML.

1. 

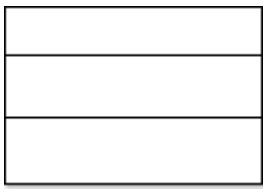
1. **Assoziation**

2. 

2. **Generalisierung**

3. 

3. **Akteur**

4. 

4. **Klasse**

5. 

5. **Usecase, Ellipse ist nicht richtig :)**

(c) (1 Punkt) Erläutern Sie den Begriff des *Mehrfachvererbung* im Kontext der Objektorientierung. Welche Problem können hierbei auftreten?

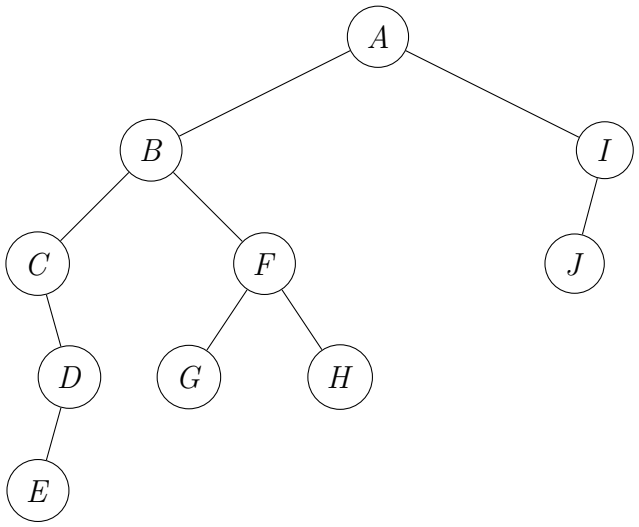
Lösung:
Von Mehrfachvererbung sprechen wir wenn eine Klasse von mehr als einer Basisklasse abgeleitet ist. Das kann zu Problemen führen, wenn mehrere der Basisklassen gleichnamige Attribute oder Methoden besitzen.

(d) (1 Punkt) Was ist das „alternative“ Konzept in Objektorientierten Sprachen die keine Mehrfachvererbung erlauben? Welcher Nachteile entsteht hierdurch?

Lösung:
Die Alternative besteht darin Schnittstellen Beschreibungen vorzugeben (Interfaces). Allerdings muss mehr Code geschrieben werden, da dieser nicht aus der Basisklasse stammen kann.

Aufgabe 6: **Datenstrukturen** Punkte: ____ / 10

Gegeben sei die folgende ungerichtete Graph $\mathcal{B}$:



(a) (2 Punkte) Stellen Sie den Binärbaum $\mathcal{B}$ in einer **Adjazenzmatrix** dar.

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

Lösung:
Der Rest wird mit 0er aufgefüllt, ließt sich aber ohne besser. Die Knoten müssen nicht unbedingt aufsteigend geordnet in die Zeilen / Spalten eingetragen werden. Wichtig ist: Der Baum ist als ungerichteter Graph zu behandeln, d.h. alles muss symmetrisch sein. Für jeden Fehler 0,5 Punkte Abzug, einmalig 1 Punkt Abzug, wenn der Baum als gerichteter Graph behandelt wird.

(b) (3 Punkte) Traversieren Sie den Baum und nennen Sie die Reihenfolge, in der die Knoten betrachtet werden für:

```
Preorder:   A B C D E F G H I J
Inorder:    C E D B G F H A J I
Postorder:  E D C G H F B J I A
```

(c) (2 Punkte) Mit ineinander geschachtelten Listen lassen sich zweidimensionale Datenstrukturen realisieren. Eine solche zweidimensionale Struktur kann immer auf eine eindimensionalen Struktur abgebildet werden. Geben Sie für eine Matrix mit `rows` Zeilen und `cols` Spalten eine Funktion `f(x, y)` an, welche den Index des Elementes in einer eindimensionalen Datenstruktur an Position x aus Zeile y ermittelt und zurück gibt.

```
1 def f(x:int, y:int) -> int:
2     global rows, cols # Anzahl an Zeilen und Spalten
3     return y * cols + x
4
5
6
```

(d) (1 Punkt) Nach welchem Prinzip arbeitet der *Stack*?

Lösung:

Der Stack arbeitet nach dem LIFO-Prinzip (Last In First Out)

(e) (1 Punkt) Nach welchem Prinzip arbeitet eine *Queue*?

Lösung:

Eine Queue arbeitet nach dem FIFO-Prinzip (First In First Out)

(f) (1 Punkt) Wie wird ein Baum bezeichnet, für den in jedem Knoten gilt, dass sich die Höhe des linken und des rechten Teilbaums jeweils um maximal eins unterscheiden?

Lösung:

Balancierter Baum

Aufgabe 7: **Daten – Information – Wissen** Punkte: ____ / 10

Gegeben sei die folgende 32 Zeichen lange Nachricht:

```
OHHHHH_NEIN_ER_HAT_EIN_HANDTUCH!
```

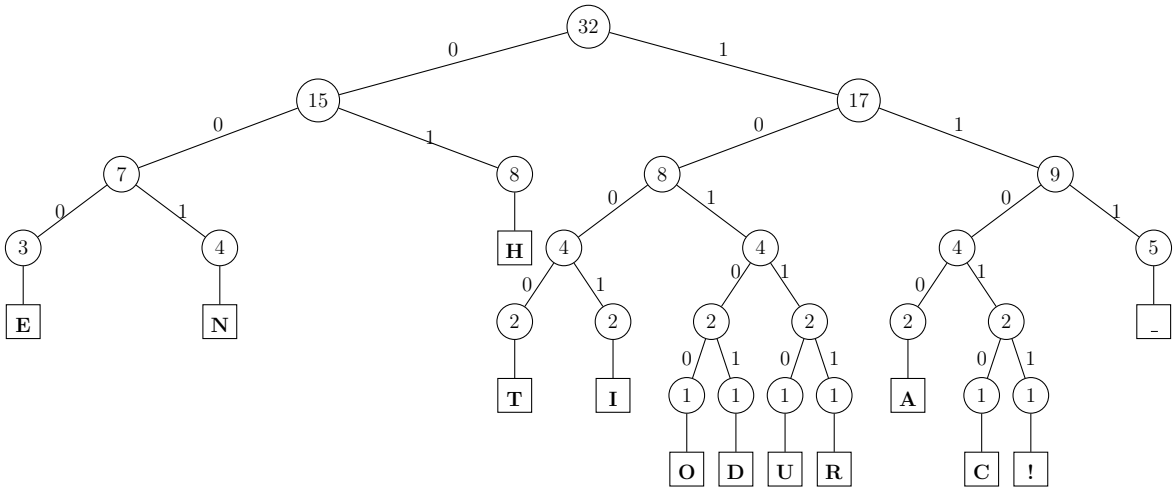
Gehen Sie im Folgenden davon aus, dass alle Zeichen des zugrundeliegenden Alphabets in der Nachricht vorkommen und das die Nachricht repräsentativ für die Auftrittswahrscheinlichkeit eines jeden Zeichens ist.

(a) (5 Punkte) Erzeugen Sie aus der Nachricht den Huffman-Code. Zeichnen Sie den zugehörigen Codebaum und geben Sie die Codetabelle an.

	A	!	C	E	D	I
#	2	1	1	3	1	2
Code	1100	11011	11010	000	10101	1001

	H	O	N	R	U	T	_
#	8	1	4	1	1	2	5
Code	01	10100	001	10111	10110	1000	111

Lösung:
3 Punkte für den Baum
2 Punkte für die Tabelle
Aber der Code ist nicht Eindeutig, also genau hinsehen.



(b) (2 Punkte) Wie groß ist der Informationsgehalt der Zeichen `O`, `A` `N` und `H`? Geben Sie auch den Rechenweg an.

$$O : -log_2(\frac{1}{32}) = 5Bit$$
$$A : -log_2(\frac{2}{32}) = 4Bit$$
$$N : -log_2(\frac{4}{32}) = 3Bit$$
$$H : -log_2(\frac{8}{32}) = 2Bit$$

(c) (1 Punkt) Mit was kann die Entropie eines Zeichens einer Informationsquelle gleichgesetzt werden?

Lösung:
Mit dem *mittlere Informationsgehalt*

(d) (1 Punkt) Wie häufig muss bei der Digitalisierung eines Signals abgetastet werden, um eine fehlerfreie Rekonstruktion zu gewährleisten?

Lösung:
Das Signal muss mit **mehr** als dem Doppelten der Maximalfrequenz abgetastet werden.

(e) (1 Punkt) Welche Randbedingungen muss das Signal erfüllen um aus dem zeitdiskreten Signal das Ursprungssignal ohne Informationsverlust rekonstruieren bzw. mit endlichem Aufwand, aber beliebig genau, approximieren zu können?

Lösung:
Das Signal muss kontinuierlich und bandbegrenztes sein.

Aufgabe 8: Debuggen & Testen Punkte: ____ / 10

(a) (6 Punkte) Finden Sie in folgendem Code alle Fehler.

```
1 def add_queen(new_row, col, old_solution)
2     new solution = ()
3     foreach solution in old_solutions:
4         for new_col in range(0:len(col):-1):
5             if self.attacks(new_row, new_col, solution);
6                 new_solution.append(Solution % [new_col])
7     raise new_solution
```

Fehler in Zeile 1:

Lösung:
`self` fehlt
: Fehlt am Ende

Fehler in Zeile 2:

Lösung:
`new_solution` statt `new solution`
`[]` statt `()` wegen append in Zeile 6

Fehler in Zeile 3:

Lösung:
`for` statt `foreach`
`old_solution` statt `old_solutions`

Fehler in Zeile 4:

Lösung:
, anstatt : bei `range`
1 statt -1 bei `range`

Fehler in Zeile 5:

Lösung:
: statt ;

Fehler in Zeile 6:

Lösung:
`solution` statt `Solution`
`+` statt `%`

Fehler in Zeile 7:

Lösung:
`return` statt `raise`

(b) (3 Punkte) Zwischen welchen drei Arten der *Bedingungsüberdeckung* wird unterschieden und was besagen diese jeweils?

Lösung:
Unterscheiden wird zwischen Einfacher-, Mehrfacher- und minimal Mehrfacherübderdeckung.

- Bei der einfachen Bedingungsüberdeckung muss jeder Teilausdruck einer Bedingung einmal zu True und einmal zu False ausgewertet werden.
- Bei der mehrfachen Bedingungsüberdeckung werden alle Kombinationen der Teilausdrücke einer Bedingung zu True, bzw. False ausgewertet.
- Die minimal Mehrfacherübderdeckung beachtet die Auswertungsreihenfolge, bzw. die Logik, der Teilausdrücke einer Bedienung. Somit lassen sich Belegungen, die zu äquivalenten Testfällen führen vermeiden, sie reduziert die Tesfälle der mehrfachen Bedingungsüberdeckung auf eine minimum, daher der Name.

(c) (1 Punkt) Worin besteht der Unterschied zwischen Black- und Glass-Box-Test?

Lösung:
Beim Black-Box-Test werden die Testfälle aus der Spezifikation eines Systems oder einer Komponente abgeleitet. Das System wird als nicht einsehbar aufgefasst, dessen Verhalten nur über das Beobachten seiner E in- und Ausgaben erfasst werden kann. Beim Glass-Box-Test haben wir alle Informationen über das Programm, seine Verzweigungslogiken und Programmpfade.

Aufgabe 9: **Prozesse & Synchronisation** Punkte: ____ / 10

(a) (4 Punkte) Gegeben sei der folgende unvollständige Programmcode zur Lösung des *Erzeuger-Verbraucher-Problems*. Ordnen Sie die 8 fehlenden Befehle den Zeilen (10, 11, 14, 15, 19, 20, 23, 24) im Code zu.

`s_mutex.acquire()` gehört in Zeile 11 oder 20
`s_mutex.acquire()` gehört in Zeile 20 oder 11
`s_mutex.release()` gehört in Zeile 14 oder 23
`s_mutex.release()` gehört in Zeile 23 oder 14
`s_empty.acquire()` gehört in Zeile 19
`s_empty.release()` gehört in Zeile 15
`s_full.acquire()` gehört in Zeile 10
`s_full.release()` gehört in Zeile 24

```
1  DEPOT, DEPOT_CAPACITY = [], 5
2
3  s_empty = threading.Semaphore(DEPOT_CAPACITY)
4  s_full  = threading.Semaphore(0)
5  s_mutex = threading.Semaphore(1)
6
7  def consumer():
8      while True:
9         # Zeile 10
10        # Zeile 11
11         shoe = DEPOT.pop()
12         print("Sell {0}, {1} shoes left.".format(shoe, len(DEPOT)))
13         # Zeile 14
14         # Zeile 15
15
16 def producer():
17     while True:
18         # Zeile 19
19         # Zeile 20
20         shoe = "pumps"
21         DEPOT.append(shoe)
22         # Zeile 23
23         # Zeile 24
```

(b) (1 Punkt) Erklären Sie in eigenen Worten - unter Verwendung der Begriffe *Betriebsmittel* und *Prozesse* was in der Informatik als „kritischer Abschnitt“ verstanden wird.

Lösung:
Wollen in einem mehr Prozessor Szenario mehrere **Prozesse** gleichzeitig auf das selbe **Betriebsmittel** (schreibend) zugreifen so wird dieser zeitliche *Abschnitt* als *kritisch* bezeichnet.

(c) (1 Punkt) Erklären Sie in eigenen Worten was der „wechselseitiger Ausschluss“ in der Informatik bewirkt?

Lösung:
Mit dem *wechselseitiger Ausschluss* (Mutex) wird garantiert das sich nie mehr als ein Prozess in einem kritischen Abschnitt befindet.

(d) (2 Punkte) Was wird mit dem Begriff *busy wait* umschrieben?

Lösung:

Aktives Warten - auch busy waiting genannt - bezeichnet eine Aktivität eines Computerprogramms, mit der die Zeit bis zur Erfüllung einer Bedingung aktiv durch Ausführung von Anweisungen, welche den Zustand des Programms nicht verändern, überbrückt wird.

(e) (2 Punkte) Der folgende Code zeigt wie man theoretisch ein Semaphor programmieren kann. Bei der Verwundung einer solchen Lösung kann es jedoch zu dem Problem kommen das keine echt Synchronisation stattfindet. Woran liegt das und wie lässt sich dieser Fehler vermeiden?

```
1  def P(d):
2      d.value -= 1
3      if d.value < 0 :
4          enqueue(my_id, d.list)
5          sleep()
6
7  def V(d):
8      if d.value < 0:
9          wakeup(dequeue(d.list))
10     d.value += 1
```

Lösung:

Der Fehler rührt daher, das die Addition und Zuweisung (+V+) bzw. Subtraktion (+P+) und Zuweisung zwei Befehle sind, die nicht atomar ausgeführt werden. Die Lösung des Problems ist es atomare Befehle wie `fetch_and_add` oder `test_and_set` zu verwenden.

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

Falls Sie auf der Titelseite das Kreuz für 9 CP's gesetzt haben entfällt diese Aufgabe für Sie.

(a) (6 Punkte) In der Vorlesung wurden drei Entwurfsmethoden für Algorithmen vorgestellt. Benennen Sie diese und erklären Sie kurz deren Arbeitsweise.

Lösung:

Je 0,5 Punkte für den Namen und 1,5 Punkte für die Erklärung. Divide and Conquer:
Teile ein komplexes Problem solang auf, bis es „beherrschbar" wird. Löse dieses Teilproblem und setze schließlich alle Teilprobleme zusammen (Teilen, Herrschen, Mergen).

Greedy:
Verwenden einer Bewertungsfunktion um die „Güte" einer Teil-/ Lösung abzuschätzen. Dabei tastet sich das Greedyverfahren Schritt für Schritt an eine Lösung heran.

Backtracking:
Systematisches durchlaufen des (gesamten) Lösungsraum. Backtracking kann als Erweiterung von Greedy angesehen werden, indem auch *Backpropagation* möglich sind, die das verharren in einem lokalen Optimum übergehen.

(b) (4 Punkte) Formulieren Sie einen **Greedy**-Algorithmus als Anweisungsvorschrift (oder Pseudocode), der die Elemente l_i einer n-elementigen Liste `L` in aufsteigender Reihenfolge sortiert.

Lösung:
Unterteile die Liste `L` in einen sortierten und einen unsortierten Bereich. Wähle aus dem unsortierte Bereich das kleinste Element aus und vertausche es mit der letzten Position des sortierten Bereichs. Zu Beginn ist das Intervall des sortierten Bereiches $[0, 0]$ das des unsortierten Bereiches ist $[0, n-1]$.

```
1 def selection_sort(v):
2     for p in range(len(v) - 1):
3         v_min = p
4         for q in range(p, len(v)):
5             if v[q] < v[v_min]:
6                 v_min = q
7         v[v_min], v[p] = v[p], v[v_min]
8     return v
```