Informatik 1
Einführung in die Programmierung
WS 2018

Prof. Dr. Peter Thiemann
Institut für Informatik
Albert-Ludwigs-Universität Freiburg

Name: _____

RZ-LOGIN: _____

Übungsgruppe/Tutor:

☐ Hans Albert (ha60)	☐ Tobias Buerger (tb305)	☐ Johannes Gramsch (jg405)
☐ Julia Abels (ja131)	☐ Corbinian Gruber (cg278)	☐ Zacharias Haeringer (zh11)
☐ David Hoefflin (dh234)	☐ Hannes Jakob (hj51)	☐ Felix Karg (fk265)
☐ Maximilian Nazarati (mn181)	☐ Florian Pollitt (fp118)	☐ Frank Schuessele (fs321)
☐ Lars Sipos (ls305)	☐ Janek Spaderna (js1344)	☐ Stefan Strang (ss984)
☐ Joseline Veit (jv27)	☐ The Son Vu (tv54)	☐ Francine Wagner (fw122)

- Bitte **schreiben Sie Ihren Namen auf jedes Blatt**.
- Es sind **keine Hilfsmittel** wie Skripte, Bücher, Notizen oder Taschenrechner erlaubt. Desweiteren sind alle elektronischen Geräte (wie z.B. Handys) auszuschalten.
- Benutzen Sie zur Bearbeitung der Aufgaben jeweils den Platz unterhalb der Aufgaben sowie ggf. den Platz auf den beigefügten leeren Seiten.
- Falls Sie mehrere Lösungsansätze einer Aufgabe erarbeiten, markieren Sie deutlich, welcher gewertet werden soll.

	Erreichbare Punkte	**Erzielte Punkte**	**Nicht bearbeitet**
Aufgabe 1	5		
Aufgabe 2	6		
Aufgabe 3	5		
Aufgabe 4	6		
Aufgabe 5	6		
Aufgabe 6	12		
Gesamt	40		

Aufgabe 1 (Python-Shell; Punkte: 1+1+1+1+1)**.**

Welche Ausgaben erhalten Sie in der Python-Shell bei Eingabe der folgenden Befehle?

(a)
```
>>> x = 2000
>>> x % 4 == 0 and (x % 100 != 0 or x % 400 == 0)
True
```

(b)
```
>>> list(zip([1, 3, 7], 'XKCD'))
[(1, 'X'), (3, 'K'), (7, 'C')]
```

(c)
```
>>> xs = [2, 0, 7, 3]
>>> for i in range(3):
...     if xs[i] > xs[i+1]:
...         xs[i], xs[i+1] = xs[i+1], xs[i]
...
>>> xs
[0, 2, 3, 7]
```

(d)
```
>>> n, b = 22, ""
>>> while n > 0:
...     rem = n % 2
...     b = str(rem) + b
...     n = n // 2
...
>>> n, b
(0, '10110')
```

(e)
```
>>> x = 'MANBAT'
>>> (x[2] + x[1] + ' ') * 8 + x[3:] + x[:3]
'NA NA NA NA NA NA NA NA BATMAN'
```

Aufgabe 2 (Wissen; Punkte: 1+1+1+1+1+1)**.**

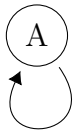
Kreuzen Sie im folgenden jeweils ausschließlich die richtige(n) Antwort(en) an. Falsche Kreuze führen zu einer 0-Punkte-Bewertung der jeweiligen Teilaufgabe.

(a) Welche der folgenden Begriffe sind keine notwendigen Eigenschaften eines Algorithmus, entsprechend der Definition aus der Vorlesung?

(1) Präzision (2) statische Finitheit (3) dynamische Finitheit (4) Generalität (5) Effizienz (6) Vollständigkeit (7) Effektivität (8) Terminierung

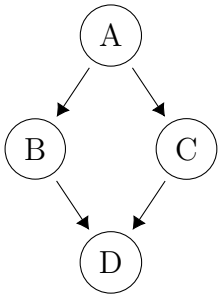
- ☐ (3), (5) und (6)
- ☐ (4), (6) und (7)
- ☐ (4), (5) und (6) ✓
- ☐ (4) und (7)

(b) Welche der folgenden Strukturen sind Bäume?

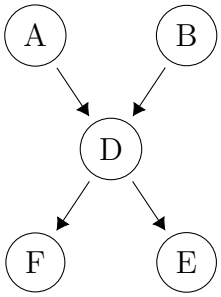


(1)
☐

(2)
☐ ✓



(3)
☐



(4)
☐

(c) Welche der folgenden Definitionen der Suchbaumeigenschaft ist korrekt?

- ☐ Alle Markierungen im linken Teilbaum sind kleiner als oder gleich groß wie die aktuelle Knotenmarkierung, alle Markierungen im rechten Teilbaum sind größer.
- ☐ Alle Markierungen im linken Teilbaum sind kleiner als die aktuelle Knotenmarkierung, alle Markierungen im rechten Teilbaum sind größer. ✓
- ☐ Alle Markierungen im linken Teilbaum sind größer als die aktuelle Knotenmarkierung, alle Markierungen im rechten Teilbaum sind kleiner.

(d) Die In-Order-Traversierung eines Suchbaums gibt dessen Knotenmarkierungen

- ☐ aufsteigend sortiert aus. ✓
- ☐ absteigend sortiert aus.
- ☐ weder absteigend noch aufsteigend sortiert aus.

(e) Eine Dateninvariante ist eine logische Aussage über die Attribute eines Objekts, die

- ☐ während der gesamten Lebensdauer des Objekts unerfüllt bleiben muss.
- ☐ ausschließlich bei der Erzeugung des Objekts erfüllt sein muss.
- ☐ während der gesamten Programmausführung unerfüllt sein muss.
- ☐ während der gesamten Lebensdauer des Objekts erfüllt sein muss. ✓

(f) Aggregierung liegt vor, falls

- ☐ Argumente von Funktionen Objekte sind.
- ☐ Der Rückgabewert einer Funktion selbst eine Funktion ist.
- ☐ Argumente von Funktionen selbst wieder Funktionen sind.
- ☐ Attribute von Objekten selbst wieder Objekte sind. ✓

Aufgabe 3 (Fehler; Punkte: 3+2)**.**

Die Funktion `sorted(ls: list) -> list` soll, von einer Eingabeliste `ls` eine aufsteigend sortierte Kopie erstellen und diese zurückgeben. Der folgende Code enthält drei Fehler:

```
1 def sorted(ls: list) -> list:
2   """Return a sorted copy of a list."""
3     xs = ls[:]
4     for i in range(len(ls) - 1):
5         for j in range(0, len(ls) - i):
6             if xs[j] > xs[j+1]:
7                 xs[j], xs[j+1] = xs[j+1], xs[j]
8     print(xs)
```

(a) Finden Sie die Fehler und tragen Sie, für jeden Fehler, die Zeile in welche der Fehler im Programmcode auftritt, den Fehlertyp (Semantik-, Laufzeit- oder Syntaxfehler), sowie eine kurze Erläuterung, in die untenstehende Tabelle ein.

Zeile	Fehlertyp	Beschreibung
2	Syntax	Docstring falsch eingerückt
5	Laufzeit	IndexError, für j+1 = len(xs)
8	Semantik	Returned None statt der Liste

(b) Geben Sie, für jeden Semantik- oder Laufzeitfehler, einen geeigneten Testfall, bestehend aus Eingabeliste und erwartetem Rückgabewert, an. Die Testfälle sollten im nicht-korrigierten Code zu einem Fehler führen.

Eingabeliste (`ls`)	Erwarteter Rückgabewert
[ ]	[ ]
[2, 3, 1]	[1, 2, 3]

Aufgabe 4 (Testen; Punkte: 2+4)**.**

Die Funktion `fizzbuzz(n: int) -> str` soll für eine positive ganze Zahl n `'Fizz'` zurückgeben, falls n ganzzahlig durch 3 aber nicht ganzzahlig durch 5 teilbar ist, `'Buzz'`, falls n ganzzahlig durch 5 aber nicht ganzzahlig durch 3 teilbar ist, und `'FizzBuzz'` falls n ganzzahlig durch 3 und durch 5 teilbar ist. Ist die Zahl weder ganzzahlig durch 3 noch durch 5 teilbar, so soll die Zahl selbst als String zurückgegeben werden. Das Gerüst der Funktion sieht wie folgt aus:

```
def fizzbuzz(n: int) -> str:
    if n % 3 == 0 and n % 5 == 0:
        # ...
    elif n % 3 == 0:
        # ...
    elif n % 5 == 0:
        # ...
    else:
        # ...
```

(a) Wieviele Testfälle sind mindestens notwendig um `fizzbuzz` zu testen? Begründen Sie Ihre Antwort.

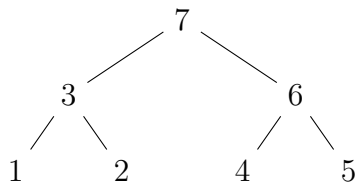
Antwort: 4, damit jeder if/elif/else Zweig besucht wird

(b) Tragen Sie, entsprechend Ihrer Antwort aus (a) viele Testfälle in die untenstehende Tabelle ein. (Es müssen und sollen nicht alle Zeilen ausgefüllt werden.)

Eingabe (`n`)	Erwarteter Rückgabewert
1	'1'
3	'Fizz'
5	'Buzz'
15	'FizzBuzz'

Aufgabe 5 (Bäume; Punkte: 2+4)**.**

(a) Geben Sie die Knotenmarkierungen des folgenden Binärbaums in der Reihenfolge einer *post-order* Traversierung an:



Antwort: 1, 2, 3, 4, 5, 6, 7

(b) Betrachten Sie die folgende Klasse zur Repräsentation eines Baumes:

```
class Tree:
    def __init__(self, mark, children: list):
        self.mark = mark
        self.children = children
```

Hierbei entspricht `mark` der Markierung und `children` der Liste an Kindern. Der in (a) abgebildete Baum kann wie folgt dargestellt werden:

```
t = Tree(7, [
        Tree(3, [ Tree(1, []), Tree(2, []) ]),
        Tree(6, [ Tree(4, []), Tree(5, []) ]) ]
)
```

Implementieren Sie eine Methode `leaves`, welche die Markierungen aller Blattknoten als Liste zurückgibt. Verwenden Sie hierzu das vorgegebene Methodengerüst. Hinweis: Ein Knoten ist ein Blattknoten, wenn die Kinderliste leer ist. Beispiel:

```
# t as defined above
>>> t.leaves()
[1, 2, 4, 5]
```

Lösung:

```
class Tree:
    # __init__ as defined above

    def leaves(self):
        if self.children:
            v = []
            for child in self.children:
                v += child.leaves()
            return v
        else:
            return [self.mark]
```

Aufgabe 6 (Rekursion, Endrekursion; Punkte: 6+6)**.**

(a) Implementieren Sie eine endrekursive Version der folgenden rekursiven Funktion:

```
def sum(n: int) -> int:
    if n == 0:
        return 0
    else:
        return n + sum(n - 1)
```

Lösung:

```
def sum(n):
    return add_sum(0, n)

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

(b) Die Funktionen `p(n: int) -> bool` und `q(n: int) -> bool`, sowie `bar(n: int) -> int`, `baz(n: int) -> int` und `quux(n: int) -> int` sind entsprechend ihrer Signatur definiert. Implementieren Sie eine iterative Variante der folgenden endrekursiven Funktion:

```
def foo(n: int) -> int:
    if p(n):
        return bar(n)
    elif not q(n):
        return foo(baz(n))
    else:
        return foo(quux(n))
```

Lösung 1:

```
def foo(n: int) -> int:
    while not p(n):
        if not q(n):
            n = baz(n)
        else:
            n = quux(n)
    return bar(n)
```

Lösung 2:

```
def foo(n: int) -> int:
    while True:
        if p(n):
            n = bar(n)
            break
        elif not q(n):
            n = baz(n)
            continue
        else:
            n = quux(n)
            continue
    return n
```