

Probeklausur ***Einführung in die Programmierung mit*** ***Skriptsprachen***

Hinweise:

Zur Bearbeitung der Aufgaben ist als Hilfsmittel ein beidseitig von Hand beschriebenes DIN A4-Blatt erlaubt. Schreiben Sie bitte Ihren Namen auf dieses Blatt und geben es mit ab. Handys müssen ausgeschaltet in den Taschen bleiben, und die Taschen stellen Sie bitte vorne in den Raum. Alle Zuwiderhandlungen werden als Betrugsversuch angesehen, und die Klausur wird als nicht bestanden gewertet. Die Arbeitszeit beträgt 90 Minuten.

Ich wünsche Ihnen viel Erfolg!

— vom Prüfling auszufüllen: —

Name:

Matr.-Nr.:

☐ Ich möchte meine Note unverbindlich an meine ILIAS-Mailadresse geschickt bekommen.

Mit meiner Unterschrift bestätige ich, dass ich die Klausur eigenständig bearbeitet habe und ausschließlich die zugelassenen Hilfsmittel benutzt habe. Ich bin (sofern oben angekreuzt) mit dem Versand der vorläufigen Note per E-Mail einverstanden.

Unterschrift:

— vom Prüfer auszufüllen: —

Aufgabe	1	2	3	4	5	6	7	8	9	Σ
max. Pkte	10	8	11	23	9	10	3	6	6	86
erreichte Pkte										

Note:

Datum, Unterschrift des Prüfers:

1. Pseudocode lesen / Python schreiben

10 Pkt.

Betrachten Sie folgenden Pseudocode:

```
input(s) //String
n = length(s)
c = 1
ch = s[1]
i = 2
while i <= n
    if s[i] == ch
        c++
    i++
print(c)
```

Teilaufgabe 1.1:

(2 Pkt)

Was berechnet dieser Code?

Teilaufgabe 1.2:

(2 Pkt)

Welche Werte haben die Variablen i und c am Ende des Programms, wenn für s der String **Pseudocodeprogramm** eingegeben wurde?

Teilaufgabe 1.3:

(6 Pkt)

Übertragen Sie den Pseudocode in Python. Benutzen Sie jedoch keine *while*-, sondern eine *for*-Schleife. Bitte führen Sie ausnahmsweise keine Eingabevalidierung durch.

2. Schleifen

8 Pkt.

Schreiben Sie eine Funktion *fak* in Pseudocode, die die Fakultät $n!$ einer positiven Integerzahl n berechnet. Benutzen Sie dabei Schleifen. Schreiben Sie auch das Hauptprogramm, in dem n eingegeben wird. Sichern Sie die Eingabe ab.

$$n! = n * (n - 1) * (n - 2) * \dots * 2$$

3. Rekursion

11 Pkt.

Teilaufgabe 3.1:

(4 Pkt)

Bringen Sie die Formel $n! = n * (n - 1) * (n - 2) * \dots * 2$ in eine rekursive Form.

Teilaufgabe 3.2:

(7 Pkt)

Schreiben Sie eine neue Funktion *fak*, die rekursiv ist und die in Teilaufgabe 1 entwickelte Formel umsetzt.

4. Programmierung

23 Pkt.

Ein Königspaar hat ein Gefängnis, das 10 Zellen fasst. In jeder Zelle sitzt eine Person. In jeder Zellentür steckt von außen ein Schlüssel. Die Schlösser sind so konstruiert, dass man die Schlüssel nur einmal umdrehen kann (und muss), wenn man die Tür verschließt (oder aufschließt).

Weil sich das Königspaar gnädig erweisen möchte, schickt es des nachts, während die Gefangenen schlafen, eine Wärter durch das Gefängnis, der die Schlösser an **allen** Zellentüren aufschließt. Dann überlegt es sich das Königspaar noch einmal. Es befiehlt dem Wärter, noch einmal durch das Gefängnis zu gehen und ab der zweiten Zelle jeden **zweiten** Schlüssel wieder umzudrehen. Kurze Zeit später soll der Wärter ab der dritten Zelle jeden **dritten** Schlüssel noch einmal umdrehen. Das setzt sich so fort, bis er schließlich den Schlüssel nur im Schloss der **zehnten** Zelle noch einmal umgedreht hat.

Teilaufgabe 4.1:

(3 Pkt)

Um diesen Ablauf in Python (ohne OOP) abbilden zu können, benötigen Sie eine Datenstruktur, die die augenblicklichen Zustände der Schlösser beschreibt.

Welche Datenstruktur wählen Sie und warum?

Teilaufgabe 4.2:

(15 Pkt)

Schreiben Sie ein Programm in Pseudocode, das berechnet, aus welchen Gefängniszellen die Insassen nach dem letzten Durchgang des Wärters entfliehen können. Benutzen Sie kein OOP.

Teilaufgabe 4.3:

(5 Pkt)

Entwerfen Sie für die Realisierung der Lösung in OOP (Python) eine Klasse (Tür-) *schloss* mit geeigneten Attributen und Methoden.

5. XML

9 Pkt.

Bitte finden Sie 6 Fehler in der folgenden Datei. Schreiben Sie, wenn möglich, die korrigierten Zeilen mit Zeilennummer auf. Wenn es mehrere Möglichkeiten der Korrektur gibt, entschieden Sie sich für die wahrscheinlichste. Betrachten Sie nicht die Einrückung.

```
01: <xml version=1.0 encoding="UTF-8">
02: <bestelllliste xmlns:r1="http://www.rechnung.de"
03:                 xmlns:kunde="http://www.kunde.de"
04:                 xmlns:kunde="http://www.rechnung.de">
05:   <kunde:rechnung r1:id="001">
06:     <kunde:name>Joerg Schmitt</kunde:name>
07:     <dollar:preis:
08:       xmlns:dollar="http://www.dollar.de">110
09:   </kunde:rechnung>
11:   <r1 rechnung r1:id="002">
12:     <name>Kurt Meier</name>
13:     <dollar:preis>213</dollar:preis>
14:   </kunde:rechnung>
15: <bestelllliste>
```

6. XML-Schema

10 Pkt.

Entwerfen Sie eine Schema-Datei, die folgendes abbildet: die Datei enthält alle Bestellungen von Kunden aus Bielefeld und Hannover eines bestimmten Monats in einem bestimmten Jahr. Beides wird in positiven Integerzahlen angegeben, Monat von 1 bis 12, Jahr von 2000 an. Die Datei listet für jeden Besteller seine bestellten Artikel auf mit Artikelnummer (positive Integerzahl), Menge (positive Integerzahl) und Preis (positive Integerzahl). Für jeden Kunden werden Anrede, Name, Vorname, Straße mit Hausnummer, Postleitzahl und Ort angegeben. Die Anrede ist ein Attribut von Name, der Ort muss Bielefeld oder Hannover sein, alle Felder sind einfache Strings. Für jeden Kunden muss mindestens eine Bestellung vorliegen.

7. XML

3 Pkt.

Geben Sie eine XML-Beispieldatei an zu den Vorgaben aus der letzten Aufgabe die mindestens zwei Kunden und mindestens drei Bestellungen insgesamt enthält.

8. Diverses

6 Pkt.

Welche der folgenden Aussagen sind richtig? Kreuzen Sie an.

Jede richtige Antwort gibt 1 Punkt, für jede falsche wird 1 Punkt abgezogen:

Falsche Antworten führen zu einem Punktabzug!

ja **nein**

- | | | |
|--------------------------|--------------------------|--|
| <input type="checkbox"/> | <input type="checkbox"/> | Es gibt in Python auch fußgesteuerte Schleifen. |
| <input type="checkbox"/> | <input type="checkbox"/> | Ob in Python <i>call-by-value</i> oder <i>call-by-reference</i> realisiert wird, hängt vom Datentyp ab. |
| <input type="checkbox"/> | <input type="checkbox"/> | Benannte Parameter müssen beim Funktionsaufruf hinter den unbenannten stehen. |
| <input type="checkbox"/> | <input type="checkbox"/> | Eine Funktion f , die in einer Funktion g definiert wird, kann nicht außerhalb von g benutzt werden. |
| <input type="checkbox"/> | <input type="checkbox"/> | Bei der Übergabe von Parametern an Funktionen mit $*$ werden Sequenzen entpackt. |
| <input type="checkbox"/> | <input type="checkbox"/> | <i>SAX</i> ist nicht standardisiert. |

9. Python-Syntax/Semantik

6 Pkt.

Spielen Sie Interpreter!

Nachfolgend finden Sie Python-Programmfragmente. Geben Sie zu jedem Programmfragment an, ob es syntaktisch richtig ist oder nicht. Ist es richtig, geben Sie zusätzlich an, welche Ausgabe es produziert. Ist es syntaktisch falsch, erklären Sie kurz den/die Fehler.

Teilaufgabe 9.1:

(2 Pkt)

```
t = ("Nr", 23, [99])
t[1] = 0
print(t)
t[3] = 0
print(t)
```

Teilaufgabe 9.2:

(2 Pkt)

```
def f(x):  
    if f(x) == 1:  
        return 0  
    else:  
        return 1  
print (f(13))
```

Teilaufgabe 9.3:

(2 Pkt)

```
def f(x, x):  
    return x*x + x  
print (f(2, 2))  
print (f(2, 3))
```