Technische Universität Berlin **WS 2014/2015**
Fakultät II, Institut für Mathematik
Sekretariat MA 5–2, Dorothea Kiefer
Prof. Dr. Michael Joswig, Prof. Dr. Martin Skutella
Dr. Yann Disser, Georg Loho, Dr. Frank Lutz

Zwischentest zur Computerorientierten Mathematik I

10.12.2014, Beginn: 14:15 Uhr, Bearbeitungszeit: 90 Minuten

Es sind keine Hilfsmittel erlaubt. Bitte nicht mit Bleistift oder mit roter Farbe schreiben!

Nachname, Vorname: _____

Matrikelnummer: _____

Studiengang: _____

1	2	3	4	5	6	Summe
10	12	8	6	10	8	54

Sie haben den Test bestanden, wenn Sie mindestens 50 % der Punkte erreichen.

1. Aufgabe (2 + 2 + 2 + 2 + 2 Punkte)

Geben Sie die asymptotische Laufzeit folgender Codefragmente in Abhängigkeit von n an. Die Laufzeit soll dabei **möglichst knapp** in Θ-Notation ausgedrückt werden. Nehmen Sie dazu an, dass elementare Operationen wie `print,+,-,*,/,//,**` jeweils eine konstante Laufzeit haben.

(a)
```
i = 0
while i < n:
    i += 5
```

Antwort: $\Theta($_____$)$

(b)
```
i = 0
while i**2 < n:
    i += 1
```

Antwort: $\Theta($_____$)$

(c)
```
j = 1
while j < n:
    j *= 2
```

Antwort: $\Theta($_____$)$

(d)
```
for i in range(1, n):
    for j in range(i, n):
        print(i)
```

Antwort: $\Theta($_____$)$

(e)
```
for i in range(1, n//2):
    for j in range(1, n//4):
        print(i)
```

Antwort: $\Theta($_____$)$

2. Aufgabe (3 + 3 + 3 + 3 Punkte)

Nachfolgend sind einige Funktionen in Pythoncode gegeben. Geben Sie zu konkreten Eingaben an, was die jeweilige Funktion zurückgibt.

(a)
```
def func1(word):
    return ';'.join(word.split(','))
```

Input: word = "C,o,M,a" Output: _____

Input: word = "CoMa," Output: _____

Input: word = "" Output: _____

(b)
```
def func2(num, bas):
    k = 0
    while bas**(k+1) <= num:
        k += 1
    return k
```

Input: (num,bas) = (27,3) Output: _____

Input: (num,bas) = (17,2) Output: _____

Input: (num,bas) = (7,10) Output: _____

(c)
```
def func3(num):
    if num == 0:
        return False
    elif num % 10 == 6:
        return True
    else:
        return func3(num//10)
```

Input: num = 10 Output: _____

Input: num = 345 Output: _____

Input: num = 5678 Output: _____

(d)
```
def func4(word):
    if word:
        if word[0] == word[-1]:
            return func4(word[1:-1])
        return False
    else:
        return True
```

Input: word = "coco" Output: _____

Input: word = "" Output: _____

Input: word = "abba" Output: _____

3. Aufgabe $(2 + 2 + 2 + 2$ Punkte)

Geben Sie bei jedem der folgenden Funktionen an, ob die Funktion

(i) bei einer gültigen Eingabe in eine Endlosschleife laufen kann,

(ii) bei einer gültigen Eingabe einen Laufzeitfehler produzieren kann,

(iii) keinen der vorigen Fehler hat, aber nicht immer das gewünschte Ergebnis berechnet,

(iv) für jede gültige Eingabe terminiert und das korrekte Ergebnis berechnet.

Es ist jeweils genau eine Auswahlmöglichkeit anzukreuzen. Für eine falsche Antwort gibt es -1 Punkte, für eine fehlende Antwort gibt es 0 Punkte. Insgesamt wird die Aufgabe nicht mit weniger als 0 Punkten bewertet.

(a) Die folgende Funktion soll die Fakultät einer natürlichen Zahl n berechnen:

```
def factorial(n):
  result = 0
  for i in range(1,n):
    result *= i
  return result
```

☐ Endlosschleife ☐ Laufzeitfehler ☐ falsches Ergebnis ☐ korrekt

(b) Die folgende Funktion soll ein Element in der Mitte der nicht-leeren `int`-Liste L bestimmen:

```
def find_mid(L):
  while(len(L) > 1):
    L = L[1:-1]
  return L[0]
```

☐ Endlosschleife ☐ Laufzeitfehler ☐ falsches Ergebnis ☐ korrekt

(c) Die folgende Funktion soll aus einer nicht-leeren `int`-Liste L eine neue Liste erstellen, in der die Differenzen aufeinanderfolgender Elemente stehen:

```
def differences(L):
  result = []
  for i,l in enumerate(L):
    result.append(L[i] - L[i+1])
  return result
```

☐ Endlosschleife ☐ Laufzeitfehler ☐ falsches Ergebnis ☐ korrekt

(d) Die folgende Funktion soll die gegebene nicht-leere `int`-Liste L umdrehen:

```
def reverse(L):
  return reverse(L[1:]) + [L[0]]
```

☐ Endlosschleife ☐ Laufzeitfehler ☐ falsches Ergebnis ☐ korrekt

4. Aufgabe (6 Punkte)

Es sei $a_0, \ldots, a_{n-1}$ eine Liste von natürlichen Zahlen mit $n \geq 2$. Ein Listenelement a_i ist eine *Spitze* der Liste, falls:

(i) $i = 0$ und $a_0 \geq a_1$, oder

(ii) $i = n - 1$ und $a_{n-1} \geq a_{n-2}$, oder

(iii) $0 < i < n - 1$ und $a_i \geq \max\{a_{i-1}, a_{i+1}\}$.

Eine Spitze ist also ein Element, das größer oder gleich seinen Nachbarn in der Liste ist, wobei die Randelemente nur einen Nachbarn haben. Die Liste

$$L = [4, 5, 3, 2, 1, 3, 6, 4, 7]$$

hat beispielsweise die Spitzen $5, 6, 7$.

Geben Sie eine Pythonfunktion `minpeak` an, die zu einer gegebenen `int`-Liste der Länge $n \geq 2$ den Wert der kleinsten Spitze bestimmt. Im obigen Beispiel ergibt `minpeak(L)` die Zahl 5.

5. Aufgabe $(2 + 2 + 2 + 2 + 2$ Punkte)

Geben Sie ohne Begründung an:

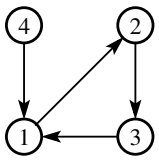
(a) Sei G ein Baum mit 23 Knoten. Wieviele Kanten hat G?

Antwort: _____

(b) Sei G ein (einfacher, ungerichteter) Graph mit 23 Knoten und 5 Kanten. Was ist die minimale Anzahl an Zusammenhangskomponenten, die für G möglich ist?

Antwort: _____

(c) Sei G der folgende gerichtete Graph:



(i) Geben Sie die Adjazenzmatrix für G samt Beschriftung der Zeilen und Spalten an:

(ii) Geben Sie die Kantenmenge einer aufspannenden Arboreszenz für G an.

Antwort: _____

(iii) Geben Sie die Knotenmengen der starken Zusammenhangskomponenten von G an.

Antwort: _____

6. Aufgabe $(2 + 2 + 2 + 2$ Punkte)

Es ist jeweils genau eine Auswahlmöglichkeit anzukreuzen. Für eine falsche Antwort gibt es -1 Punkte, für eine fehlende Antwort gibt es 0 Punkte. Insgesamt wird die Aufgabe nicht mit weniger als 0 Punkten bewertet.

(a) Die Binärdarstellung der Zahl 100 ist:

☐ 1110110 ☐ 1100010 ☐ 1010110 ☐ 1100100

(b) Die Ternärdarstellung (zur Basis 3) der Zahl 76 ist:

☐ 2121 ☐ 2211 ☐ 2112 ☐ 1222

(c) Die 6-stellige 2-Komplementdarstellung der Zahl -3 zur Basis 2 ist:

☐ 111101 ☐ 001100 ☐ 000011 ☐ 010111

(d) In Python hat nach der Zuweisung a = (2e+15 + 2e−15)*(2e+15 + 2e−15) die Variable a den Wert:

☐ 4e+30 + 8.0 + 4e−30 ☐ 4e−30 ☐ 4e+30 ☐ 4e+15

Platz für Nebenrechnungen: