Technische Universität Berlin **SoSe 2020**
Fakultät II, Institut für Mathematik
Sekretariat MA 6–2, Antje Schulz
Prof. Dr. Michael Joswig
Dr. Frank Lutz, Sarah Morell, Manuel Radons

Modulklausur „Computerorientierte Mathematik I+II “

30.10.2020, 13:15 Uhr bis 15:45 Uhr (Bearbeitungszeit: 150 Minuten)

Nachname, Vorname: _____

Matrikelnummer: _____

Studiengang: _____

Platznummer: _____

Füllen Sie bitte zuerst das Deckblatt vollständig und leserlich aus. Damit erklären Sie, dass

- **Sie keinerlei Symptome einer SARS-CoV-2 Erkrankung (wie Husten, Fieber, außergewöhnliche Müdigkeit, Kopf- und Gliederschmerzen oder eine Störung des Geruchs- oder Geschmackssinns) zeigen;**
- Ihnen die für diese Prüfung relevanten Zulassungsvoraussetzungen aus der StuPO bekannt sind. Ihnen ist außerdem bewusst, dass deren Nichterfüllung zur Ungültigkeit der Prüfung führen kann (§39 Abs. 2 Satz 4 AllgStuPO);
- Ihnen bekannt ist, dass die Teilnahme an der Prüfung eine ordnungsgemäße Anmeldung voraussetzt, andernfalls die Prüfung nicht gültig ist (§39 Abs. 2 AllgStuPO);
- Ihnen bekannt ist, dass eine Prüfung, die unter bekannten und bewusst in Kauf genommenen (sonstigen) gesundheitlichen Beeinträchtigungen abgelegt wird, grundsätzlich Gültigkeit hat.

Bei Zugang und Abgang sowie während der Prüfung sind Sie zum Tragen eines Mund-Nasen-Schutzes verpflichtet. Für diese Klausur sind keine Hilfsmittel zugelassen. Mit Bleistift oder Rotstift geschriebene Klausuren können nicht gewertet werden.

Zum Bestehen der Klausur sind mindestens **40** (ITM/NidI: **34**) **Punkte** zu erzielen.

1	2	3	4	5	6	7	8	$\sum$
10	10	10	10	10	10	10	10	80

1. Aufgabe (4 + 2 + 2 + 2 Punkte)

Sei $K_n = (V, E)$ ein vollständiger Graph mit $V := \{1, \ldots, n\}$, wobei $n \geq 3$, und $m := |E|$. Sei $T = (V, E_T)$ ein Teilgraph von K_n mit $E_T = \{\{i, i+1\} \mid 1 \leq i < n\}$.

Definition: *Ein vollständiger Graph ist ein einfacher, ungerichteter Graph, in dem jeder Knoten mit jedem anderen Knoten durch eine Kante verbunden ist.*

(a) Kreuzen Sie jeweils die richtige Lösung an. Pro Teilaufgabe ist genau eine Aussage richtig. Bei Ankreuzen mehrerer Antworten zu einer Teilaufgabe gibt es keine Punkte für diese Teilaufgabe.

(i) Es gilt ...

☐ $m = n!$ ☐ $m = n^2$ ☐ $m = \binom{n-1}{2}$ ☐ $m = \binom{n}{2}$

(ii) Die Inzidenzmatrix des Graphen T ist in ...

☐ $\{0,1\}^{n \times n}$ ☐ $\{0,1\}^{n \times m}$ ☐ $\{0,1\}^{n \times (n-1)}$ ☐ $\{0,1\}^{(n-1) \times (n-1)}$

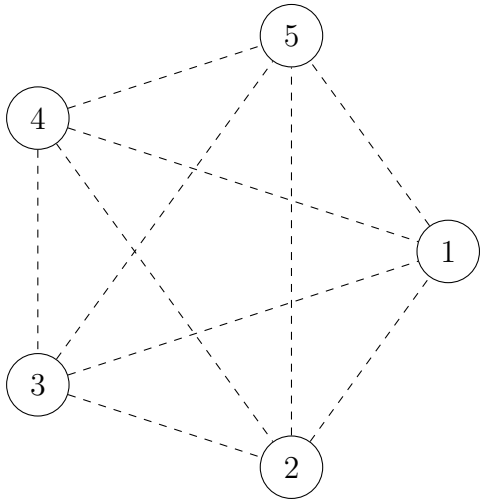
(iii) Für $u, v \in V$ mit $v \neq u$ gibt es ...

☐ genau einen u-v-Weg in K_n, der alle Knoten besucht.
☐ genau n u-v-Wege in K_n, die alle Knoten besuchen.
☐ genau $(n-2)!$ u-v-Wege in K_n, die alle Knoten besuchen.
☐ genau $n!$ u-v-Wege in K_n, die alle Knoten besuchen.

(iv) Der Graph T ist ...

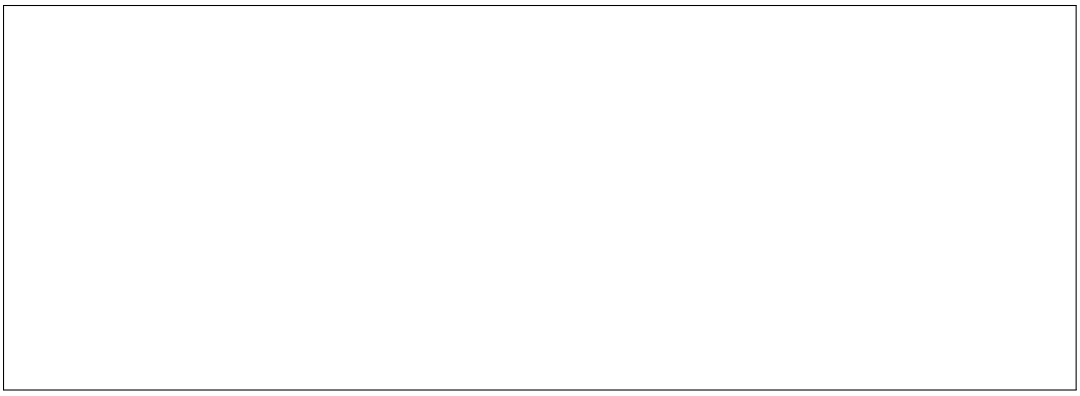
☐ zusammenhängend und kreisfrei.
☐ zusammenhängend, aber nicht kreisfrei.
☐ kreisfrei, aber nicht zusammenhängend.
☐ weder zusammenhängend noch kreisfrei.

(b) Wir betrachten nun K_5. Tragen Sie im unten angegebenen Graphen die Kantenmenge E_T des Teilgraphen T ein, indem Sie diese **fett** markieren. Tragen Sie außerdem Kantengewichte $c(e)$ für alle $e \in E$ ein, wobei $c \in \mathbb{N}^E$ folgende Eigenschaft haben soll: Bezüglich c kann T **nicht** mit Hilfe des Algorithmus von Prim berechnet werden.

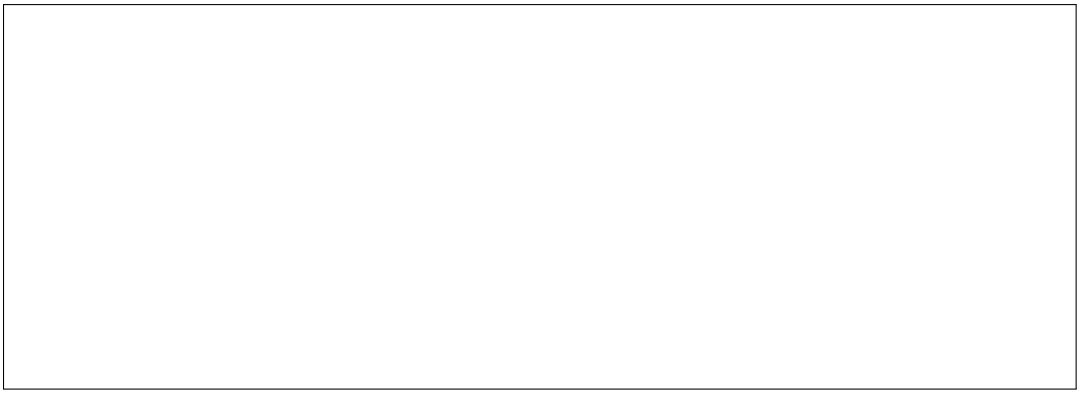


(c) Zeichnen Sie jeweils einen Graphen mit den gegebenen Eigenschaften.

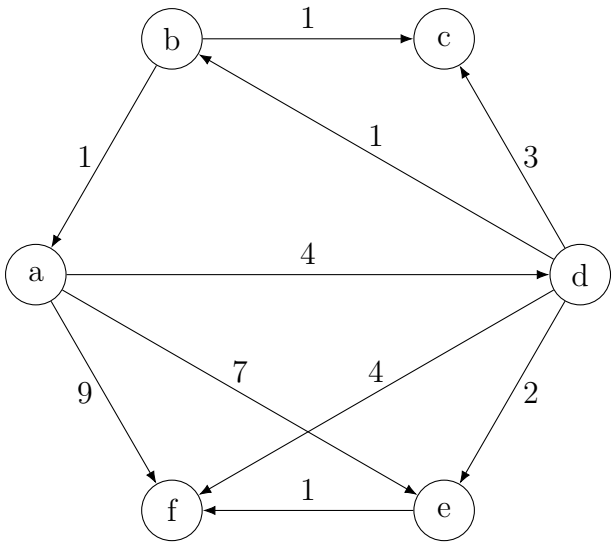
(i) Einfach, gerichtet, mit genau vier Knoten, sodass genau drei Knoten Ausgangsgrad 1 besitzen.



(ii) Einfach, ungerichtet, zusammenhängend, mit genau fünf Knoten und vier Kanten, sodass genau drei Knoten Grad 1 besitzen.



(d) Gegeben sei folgender gerichteter, gewichteter Graph G.



Führen Sie auf G den Algorithmus von Dijkstra aus, indem Sie in folgender Tabelle zu jedem Knoten und jeder Zwischenstufe des Algorithmus die korrespondierenden Knotenpotenziale und Vorgängerknoten eintragen. Der Startknoten ist a.

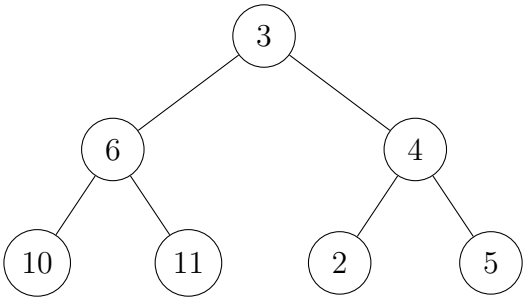
a	b	c	d	e	f
$0\ (-)$	$\infty\ (-)$	$\infty\ (-)$	$\infty\ (-)$	$\infty\ (-)$	$\infty\ (-)$

2. Aufgabe (3 + 4 + 3 Punkte)

(a) Geben Sie für die folgenden drei vergleichsbasierten Sortierverfahren die asymptotische Laufzeit in Abhängigkeit von der Anzahl n der zu ordnenden Elemente in Θ-Notation an.

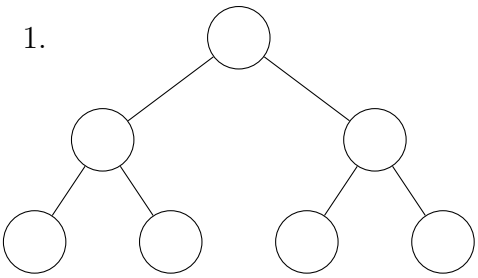
Verfahren	**Laufzeit im Worst-Case**	**Laufzeit im Average-Case**
Mergesort		
Insertionsort		
Heapsort		

(b) Gegeben sei folgender Binärbaum T.

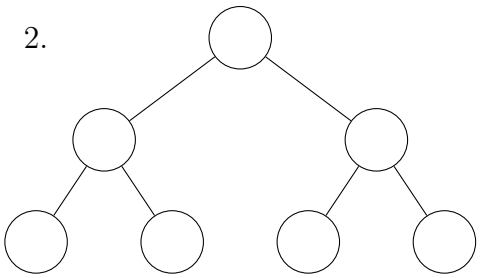


Wenden Sie die *Build-Max-Heap*-Methode auf T an, indem Sie jede durchgeführte Manipulation des Binärbaumes unten eintragen.

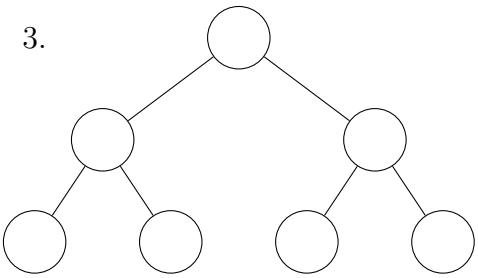
1.



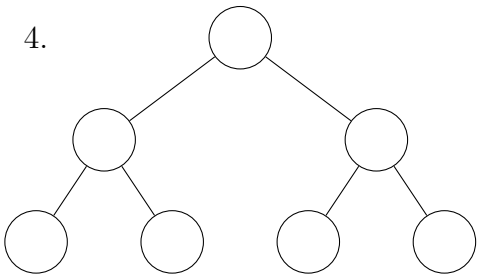
2.



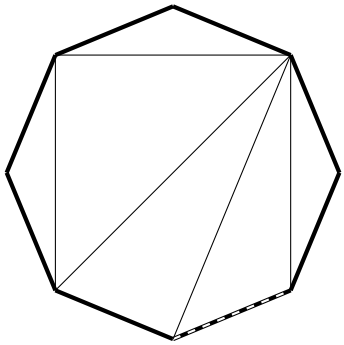
3.



4.



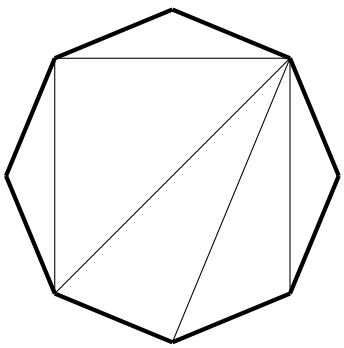
(c) Gegeben sei folgende Triangulierung eines 8-Ecks.



(i) Zeichnen Sie den dualen Graphen der Triangulierung, wobei der Wurzelknoten durch die gestrichelte Linie gekennzeichnet sei.

(ii) Erklären Sie den Begriff AVL-Baum.

(iii) Kennzeichnen Sie mittels eines **fetten Striches** eine Außenkante als Wurzelknoten, sodass der resultierende duale Graph ein AVL-Baum ist.



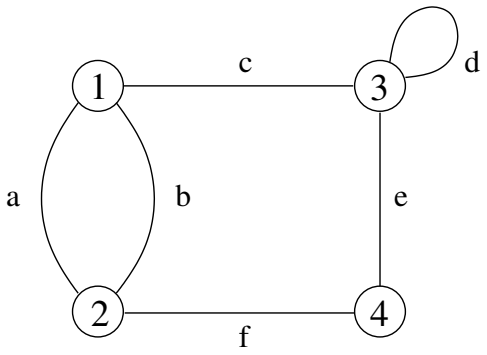
3. Aufgabe $(2 + 1 + 2 + 1 + 4$ Punkte$)$

(a) Sei E eine endliche Menge und $\mathcal{I}$ Teilmenge der Potenzmenge 2^E von E.

Geben Sie die Bedingungen an $\mathcal{I}$, so dass das Paar $(E, \mathcal{I})$ ein Unabhängigkeitssystem mit Grundmenge E ist.

(b) Was versteht man unter dem Rang eines Unabhängigkeitssystems $(E, \mathcal{I})$?

(c) Sei $G = (V, E)$ der folgende ungerichtete Graph mit Knotenmenge $V = \{1, 2, 3, 4\}$ und Kantenmenge $E = \{a, b, c, d, e, f\}$.



Geben Sie für das graphische Matroid $\mathcal{M} = (E, \mathcal{I})$ zum Graphen G die Menge $\mathcal{C}$ der Kreise von $\mathcal{M}$ an.

$\mathcal{C} =$

(d) Was ist die graphentheoretische Bedeutung einer Basis bei einem graphischen Matroid für einen zusammenhängenden Graphen?

(e) Sei

$$A_1 = \begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

die (3×3)-Einheitsmatrix und

$$A_n = \begin{array}{c} \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & \dots & a_{3n-2} & a_{3n-1} & a_{3n} \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & \dots & 0 & 0 & 1 \end{pmatrix} \end{array}$$

für $n \geq 1$ die $(3 \times 3n)$-Matrix mit n Kopien der Spalten von A_1.

Betrachten Sie das Matroid $M_n = (E_n, \mathcal{I}_n)$, wobei $E_n = \{a_1, \dots, a_{3n}\}$ die Menge der Spalten der Matrix A_n und $\mathcal{I}_n$ die Menge der Teilmengen von E_n von über $\mathbb{R}$ linear unabhängigen Spaltenvektoren seien.

Geben Sie die **Anzahl der Basen** des Matroids M_n in Abhängigkeit von n an:

Geben Sie die **Anzahl der Kreise** des Matroids M_n in Abhängigkeit von n an:

4. Aufgabe (3 + 3 + 4 Punkte)

(a) Gegeben sei ein Datensatz mit n Elementen. Geben Sie in Abhängigkeit von n die asymptotische Komplexität der Suche nach einem Schlüssel in einer Hashtabelle an, die mit den unten genannten Hashverfahren erstellt wurde. Nehmen Sie an, dass die in der Vorlesung bei der Analyse des jeweiligen Verfahrens vorausgesetzten Spezifikationen des Formats der zugehörigen Hashtabellen erfüllt sind.

	Best-Case	**Worst-Case**	**Average-Case**
Hashing mit Verkettung	$\Theta(\quad)$	$\Theta(\quad)$	$\Theta(\quad)$
Perfektes Hashing	$\Theta(\quad)$	$\Theta(\quad)$	$\Theta(\quad)$

(b) Für eine Primzahl k sei $\mathbb{Z}_k := \{0, 1, \ldots, k-1\}$ ein endlicher Körper. Für $a \in \mathbb{Z}_k \setminus \{0\}$ und $b \in \mathbb{Z}_k$ sei folgende Funktion gegeben:

$$\begin{aligned} f : \mathbb{Z}_k &\to \mathbb{Z}_k \\ x &\mapsto (ax + b) \mod k. \end{aligned}$$

(i) Erklären Sie den Begriff „kollisionsfrei“.

(ii) Beweisen Sie, dass f kollisionsfrei ist.

(c) Gegeben seien der Datensatz $[4, 6, 11, 5, 12, 3, 8]$ sowie die Hashfunktion

$$\begin{aligned} h : \mathbb{Z} &\to \mathbb{Z}_7 \\ x &\mapsto (3x + 2) \mod 7. \end{aligned}$$

(i) Hashen Sie den Datensatz mit h, wobei Kollisionen mittels **Verkettung** aufgelöst werden. Tragen Sie hierbei den Zustand der Hashtabelle nach jeder Einfügung in die Zeilen der folgenden Tabelle ein.

(ii) Hashen Sie den Datensatz mit h, wobei Kollisionen mittels **linearer Sondierung** aufgelöst werden. Tragen Sie hierbei den Zustand der Hashtabelle nach jeder Einfügung in die Zeilen der folgenden Tabelle ein.

5. Aufgabe (4 + 2 + 2 + 2 Punkte)

Gegeben sei eine deterministische Turingmaschine T durch

- die Zustandsmenge $Q = \{q_0, q_1, q_2, q_3, q_4\}$ mit Startzustand q_0,
- das Eingabealphabet $\Sigma = \{0, 1\}$,
- das Bandalphabet $\Gamma = \Sigma \cup \{B\}$,
- die Menge akzeptierender Zustände $F = \{q_4\}$

sowie folgendes Programm:

δ	0	1	B
q_0	$(q_0, 0, \mathtt{R})$	$(q_1, 1, \mathtt{R})$	—
q_1	$(q_2, 0, \mathtt{R})$	$(q_1, 1, \mathtt{R})$	—
q_2	$(q_0, 0, \mathtt{R})$	$(q_3, 1, \mathtt{R})$	—
q_3	$(q_3, 0, \mathtt{R})$	$(q_3, 1, \mathtt{R})$	$(q_4, B, \mathtt{N})$
q_4	—	—	—

Hier bedeutet ‚—‘ in Zeile q und Spalte σ den Eintrag $(q, \sigma, \mathtt{N})$.

(a) (i) Vervollständigen Sie die folgende Konfigurationsfolge:

q_0 100101 $\vdash$ _____ $\vdash$ _____ $\vdash$ _____

$\vdash$ _____ $\vdash$ _____ $\vdash$ _____

$\vdash$ _____

(ii) Wird der Input-String von der Turingmaschine T akzeptiert?

☐ Ja ☐ Nein

(b) Geben Sie die von T akzeptierte Sprache $L \subseteq \Sigma^*$ an. Ein Beweis ist hier nicht gefragt.

$L =$

(c) Wie ist T zu modifizieren, sodass die resultierende Turingmaschine T' die Sprache $\Sigma^* \setminus L$ akzeptiert? Ändern Sie **nur das Programm** der gegebenen Turingmaschine. Tragen Sie nur die **zu ändernden Einträge** in die folgende Tabelle ein.

δ	0	1	B
q_0			
q_1			
q_2			
q_3			
q_4			

(d) Kreuzen Sie jeweils die richtige Lösung an. Pro Teilaufgabe ist genau eine Aussage richtig. Bei Ankreuzen mehrerer Antworten zu einer Teilaufgabe gibt es keine Punkte für diese Teilaufgabe.

(i) Die von T akzeptierte Sprache L (siehe oben) ist ...

- ☐ eine endliche und rekursive Sprache.
- ☐ eine unendliche und rekursive Sprache.
- ☐ eine endliche und nicht rekursive Sprache.
- ☐ eine unendliche und nicht rekursive Sprache.

(ii) Die universelle Sprache ist ...

- ☐ eine endliche und rekursive Sprache.
- ☐ eine unendliche und rekursive Sprache.
- ☐ eine endliche und nicht rekursive Sprache.
- ☐ eine unendliche und nicht rekursive Sprache.

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

Gegeben seien zwei Funktionen $f, g : \mathbb{N} \to \mathbb{N}$.

(a) Geben Sie die formale Definition von $f \in \Theta(g)$ an.

(b) Beweisen oder widerlegen Sie mit einem Gegenbeispiel folgende Aussage:
Ist $f \in \Omega(g^2)$, so ist $f \notin O(g)$. Hier sei $g^2 : \mathbb{N} \to \mathbb{N} : n \mapsto (g(n))^2$.

(c) Beweisen oder widerlegen Sie mit einem Gegenbeispiel folgende Aussage:
Ist $f \in \Omega(g)$, so ist $O(g) \subseteq O(f)$.

(d) Gegeben sei eine Funktion $h : \mathbb{N} \setminus \{0, 1\} \to \mathbb{N}$ mit $h(n) = \binom{n}{2}$.

(i) Geben Sie $h(n)$ als Polynom an.

(ii) Zeigen Sie, dass $h(n) \in \Theta(n^2)$.

7. Aufgabe (4 + 6 Punkte)

(a) Kreuzen Sie jeweils die richtige Lösung an. Pro Teilaufgabe ist genau eine Aussage richtig. Bei Ankreuzen mehrerer Antworten zu einer Teilaufgabe gibt es keine Punkte für diese Teilaufgabe.

(i) Was ist die Binärdarstellung von $(19)_{13}$ bei gegebener Darstellung zur Basis 13?

☐ 00110 ☐ 10011 ☐ 10110 ☐ 01111

(ii) Welcher der folgenden Ausdrücke liefert in `python3` `False`?

☐ 2.25 == 0.25 + 0.5 + 1.5 ☐ 0.33 + 0.34 == 0.34 + 0.33
☐ 2 * 2 == 2 * 20 / 10 ☐ 7 * 9 / 90 == 1 / 10 * 7

(iii) Welche Ausgabe produziert das folgende `python3`-Codestück?

```
1 a = 2
2 b = 3
3 def square(a):
4     a = a * a
5 square(b)
6 print(b)
```

☐ 2 ☐ 3 ☐ 4 ☐ 9

(iv) Welche Ausgabe produziert das folgende `python3`-Codestück?

```
1 L = [2,4,2,3,2]
2 i = 0
3 while i < L[i]:
4     i = L[i] + i
5 print(i)
```

☐ 0 ☐ 2 ☐ 3 ☐ 4

(b) Erläutern Sie, was die nachfolgenden (s. nächste Seite) `python3`-Funktionen `func1` und `func2` berechnen sollen. Entscheiden Sie jeweils für den gegebenen Input, ob und ggf. mit welchem Rückgabewert das Programm terminiert oder ob ein Laufzeitfehler auftritt. Begründen Sie jeweils Ihre Antwort.

(i) Die Funktion `func1` ist gegeben durch:

```
def func1(x,y):
    if x < 0 or y <0:
        return "Ungueltige Eingabe"
    L=[]
    for i in range(int((x+1)//1)):
        for j in range(int((y+1)//1)):
            L.append((i,j))
    return L

print(func1(2.0143,1.7568))
```

(ii) Die Funktion `func2` ist gegeben durch:

```
def func2(k):
    if k < 2 or type(k) != type(0):
        return "Ungueltige Eingabe"
    if k == 2:
        return "Ja"
    for i in range(2,k):
        if k % i == 0:
            return "Nein"
    return "Ja"

print(func2(7.2))
```

8. Aufgabe (6 + 1 + 3 Punkte)

In dieser Aufgabe sollen Sie eine `python3`-Funktion selbst schreiben. Sie dürfen dabei alle `python3`-Befehle benutzen, die ohne das Importieren zusätzlicher Pakete in `python3` zur Verfügung stehen.

Das *Sieb des Eratosthenes* ist ein Algorithmus zur Ermittlung aller nichtnegativen Primzahlen kleiner oder gleich einer vorgegebenen Schranke $k \in \mathbb{Z}_{\geq 2}$. Das Sieb hat die folgenden Schritte:

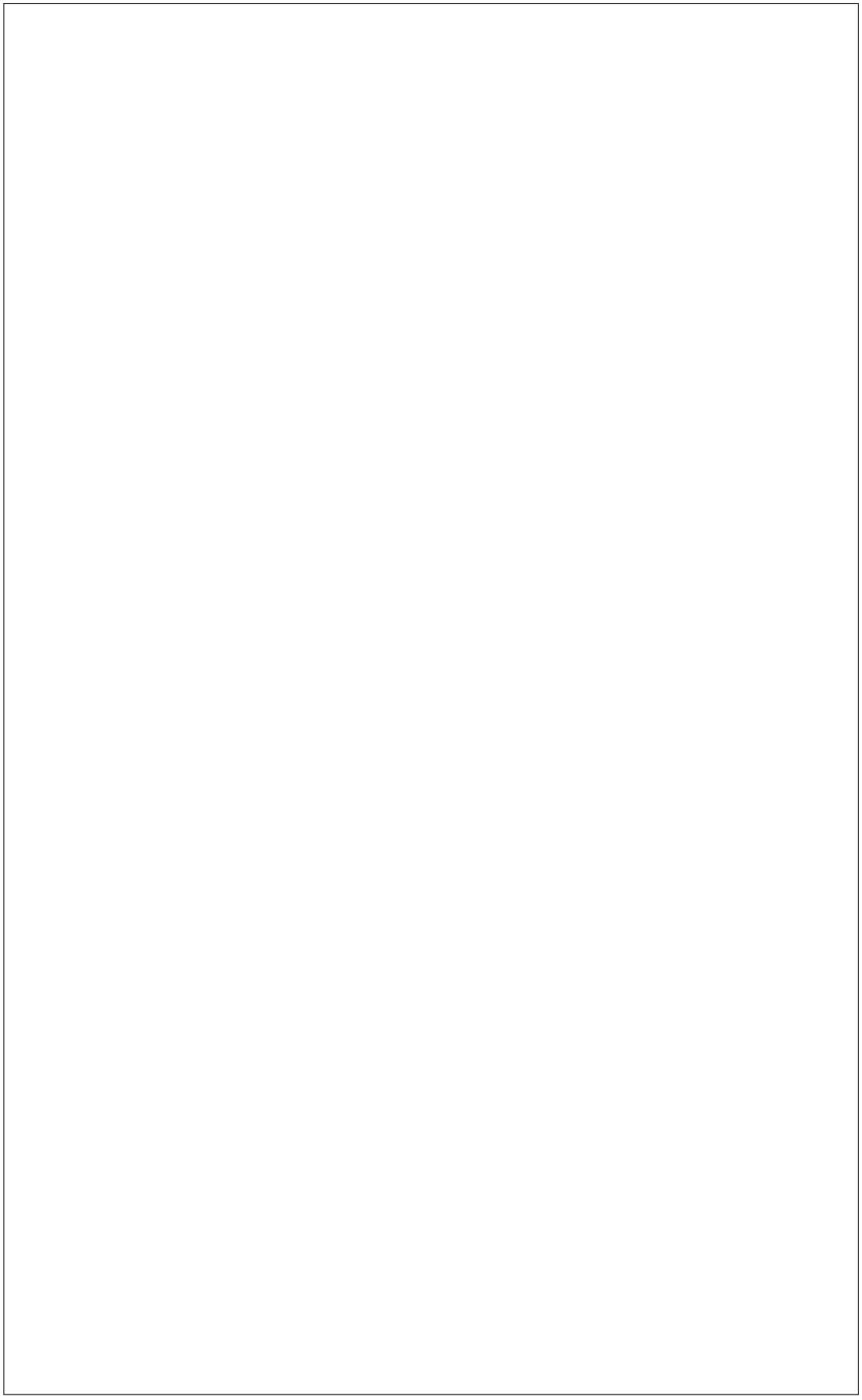
1. Schreibe alle ganzen Zahlen $2, 3, \ldots, k$ auf.
2. Setze $z := 2$.
3. Falls $z^2 > k$, brich ab. Ansonsten streiche all diejenigen der aufgeschriebenen Zahlen, welche gleich einem Produkt der Form $i \cdot z$ sind, wobei $i \in \{j \in \mathbb{Z}_{\geq 2} : j \cdot z \leq k\}$.
4. Redefiniere z als die kleinste nicht gestrichene Zahl, welche größer als das aktuelle z ist. Dann gehe zu Schritt 3.

Die nicht gestrichenen Zahlen sind die gesuchten Primzahlen kleiner oder gleich k. Die Tabelle unten stellt die Schritte des Siebes für $k = 17$ dar.

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2	3	–	5	–	7	–	9	–	11	–	13	–	15	–	17
2	3	–	5	–	7	–	–	–	11	–	13	–	–	–	17

(a) Schreiben Sie eine Funktion `sieve(k)`, welche für eine ganze Zahl $k \geq 2$ eine Liste der Primzahlen kleiner oder gleich k zurückgibt. Falls k kleiner 2 oder keine ganze Zahl ist, soll der String `Ungueltige Eingabe` zurückgegeben werden.

Hinweis: *Sie können Schritt 1 umsetzen, indem Sie eine Liste* L *mit* $[*, *, 2, 3, \ldots, k]$ *initialisieren, wobei die mit* $*$ *markierten Einträge frei wählbar sind. Dies vereinfacht die Umsetzung der Zugriffsoperationen in den Schritten 2–4.*



(b) Erklären Sie, weshalb es in Schritt 3 ausreicht, $z^2 > k$ statt $z > k$ als Abbruchbedingung zu fordern.

(c) Schreiben Sie eine Funktion `squarefree(k)`, die für eine ganze Zahl $k \geq 2$ `True` zurückgibt, falls k quadratfrei ist, und `False`, falls nicht. Dabei soll `squarefree(k)` in geeigneter Weise die Funktion `sieve(k)` aus dem ersten Aufgabenteil aufrufen.

Definition: *Eine ganze Zahl $k \geq 2$ heißt quadratfrei, falls für die (bis auf Anordnung der Faktoren) eindeutige Primfaktorzerlegung $k = \prod_{i=1}^{\ell} p_i^{e_i}$ gilt $e_1 = e_2 = \cdots = e_\ell = 1$.*