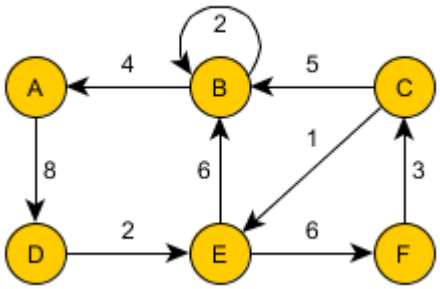
Name:_____ Matr.-Nr.:_____

Aufgabe 1 *(10 Punkte)*

Gegeben sei nebenstehender Graph G mit positiven Kantengewichten.



a) Stellen Sie G als Adjazenz-Matrix dar. *(4 Punkte)*

b) G enthält Zyklen unterschiedlicher Länge (= Anzahl der Kanten im Zyklus). Geben Sie für alle vorkommenden Zyklenlängen je einen Beispielzyklus an. *(4 Punkte)*

c) G ist stark zusammenhängend. Welche Kanten (jeweils einzeln für sich betrachtet) könnten in G entfernt werden, ohne dass diese Eigenschaft verloren geht? *(2 Punkte)*

Aufgabe 2 *(10 Punkte)*

Gegeben sei ein Array $A[0..n-1]$ mit den Zahlenwerten $x_0, x_1, \dots, x_{n-1}$. Es gilt $n \geq 1$. Die Werte x_i repräsentieren die Körpergröße einer Gruppe von n Studierenden. Geben Sie einen **vollständigen** Algorithmus **in Pseudocode-Notation** an, der als Eingabe A erhält und die **empirische Standardabweichung s** der Werte aus A zurückgibt. Es gelten folgende Beziehungen:

Die empirische Standardabweichung s ist die Quadratwurzel aus der durch n geteilten Summe der quadrierten Differenzen der Werte x_i von ihrem Mittelwert $\bar{x}$.

$$s = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^2}$$

Der Mittelwert $\bar{x}$ ist die durch n geteilte Summe der Werte x_i.

$$\bar{x} = \frac{1}{n}\sum_{i=0}^{n-1} x_i$$

Hinweise:

- Zur Berechnung der Quadratwurzel können Sie die Funktion sqrt(x) verwenden.
- Bei dieser Aufgabe kommt es nur auf die Korrektheit des Algorithmus und der Nutzung der Pseudocode-Notation an, nicht auf die Effizienz des Algorithmus.

Aufgabe 3 *(10 Punkte)*

a) Gegeben sei untenstehender Algorithmus f(m, n). Bestimmen Sie für den Algorithmus die genaue **Laufzeitfunktion** und die **Komplexitätsklasse**. Verwenden Sie dabei als Basisoperation die **Addition**. Unterscheiden Sie zwischen **Best-** und **Worst-Case**. *(7 Punkte)*

```
ALGORITHM f(m, n)
// Input: zwei positive ganze Zahlen m und n
result ← 0
for i ← 0 to m - 1 do
    for j ← n downto 1 do
        k ← 1
        while k * k ≤ n do
            result ← result + k
            k ← k + 1
return result
```

b) Zur Lösung eines Problems stehen zwei Algorithmen $A_1(n)$ und $A_2(n)$ zur Auswahl. Dabei repräsentiert n die Anzahl der Kunden eines Unternehmens. Für die Algorithmen gilt jeweils: *best case = worst case*.
Die Laufzeitfunktionen der Algorithmen sind $T_1(n) = 42n$ und $T_2(n) = 2n \log_2 n + 4n$.
Welcher Algorithmus sollte aus Effizienzgründen verwendet werden? Begründen Sie Ihre Antwort kurz und geben Sie die relevanten Rechenschritte an. *(3 Punkte)*

Aufgabe 4 *(10 Punkte)*

a) Gegeben sei untenstehender Algorithmus f(n). Stellen Sie für f(n) die ***Rekursionsgleichung der Laufzeitfunktion T(n)*** auf. Verwenden Sie als Basisoperation die ***print-Anweisung***. Bestimmen Sie die **genaue Laufzeitfunktion** in geschlossener (d. h. nicht-rekursiver) Schreibweise ***durch rückwärtiges Einsetzen***. (Ein Beweis mittels vollständiger Induktion ist *nicht* notwendig.) Geben Sie außerdem die ***Komplexitätsklasse*** des Algorithmus an. *(8 Punkte)*

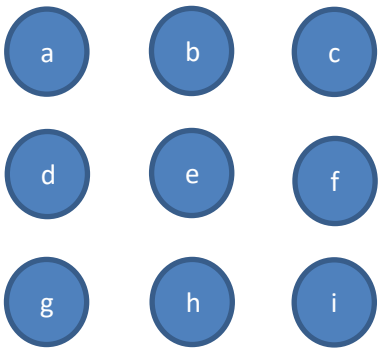
```
ALGORITHM f(n)
// Input: Eine positive ganze Zahl n
if n > 1 then
    f(⌊n/2⌋)
    print("Ra-")
    f(⌊n/2⌋)
    print("zu-")
    f(⌊n/2⌋)
    print("pal-")
    f(⌊n/2⌋)
    print("tuff!")
```

b) Erläutern Sie, welche Aufgabe die frei wählbare Konstante n_0 in der Definition der *O-Notation* besitzt. *(2 Punkte)*

Aufgabe 5 *(10 Punkte)*

Gegeben sei der ***gerichtete*** Graph G = (V, E) mit den Knoten V = {a, b, c, d, e, f, g, h, i} und Kanten E = {(b, a), (c, e), (c, f), (d, e), (d, g), (e, f), (f, i), (g, h), (h, e), (h, f)}

a) Zeichnen Sie die Kanten in den Graphen ein. *(2 Punkte)*



b) Sortieren Sie die Knoten des Graphen topologisch mit Hilfe der **Tiefensuche**. Berücksichtigen Sie dabei die **gegebene Knoten- und Kantenreihenfolge**. Geben Sie dabei insbesondere alle Zustände des **DFS-Traversierungsstacks** und die **endgültige Sortierreihenfolge** der Knoten an. *(7 Punkte)*

c) Wie viele **Einstiegspunkte** besitzt der Graph? *(1 Punkt)*

Aufgabe 6 *(10 Punkte)*

Gegeben sei folgendes Array von Werten [4, 2, 8, 9, 3, 7, 5, 6]. Sortieren Sie die Werte in **absteigender** alphabetischer Reihenfolge mit Hilfe der unten genannten Sortierverfahren.

- ***Bubblesort***: Geben Sie den Zustand des Arrays **nach jedem Durchlauf der äußeren Schleife** an. *(4 Punkte)*

- ***Quicksort***: Verwenden Sie die **Partitionierung nach Hoare** und geben Sie den Zustand des Arrays **nach jedem rekursiven Aufruf** an. Kennzeichnen Sie jeweils das verwendete Pivot-Element. *(6 Punkte)*

Aufgabe 7 *(10 Punkte)*

a) In der vierten Klasse einer Grundschule wurde ein Test geschrieben. Der Lehrer, Mr. Garrison, möchte die Testergebnisse **nach Noten** sortieren. Schüler mit identischen Noten sollen **zusätzlich nach ihrem Vornamen** sortiert sein. (Siehe nebenstehendes Beispiel.)

Note	Vorname
1	Wendy
2	Kyle
2	Stan
3	Butters
3	Eric
3	Timmy
5	Kenny

Durch welche **Kombination von Sortierverfahren** kann dieses Ergebnis erreicht werden? *(5 Punkte)*

Zuerst die Vornamen mit Quicksort und dann die Noten mit Bubblesort sortieren.	□ ja	□ nein
Zuerst die Noten mit Mergesort und dann die Vornamen ebenfalls mit Mergesort sortieren.	□ ja	□ nein
Zuerst die Vornamen mit Insertionsort und dann die Noten mit Heapsort sortieren.	□ ja	□ nein
Zuerst die Noten mit Selectionsort und dann die Vornamen mit Heapsort sortieren.	□ ja	□ nein
Zuerst die Vornamen mit Selectionsort und dann die Noten mit Mergesort sortieren.	□ ja	□ nein

b) Für welche Anwendungsfälle ist Quicksort gut geeignet? *(2 Punkte)*

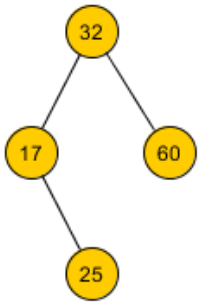
□ Das Sortieren verketteter Listen.
□ Das Sortieren von Datensätzen auf externen Speichermedien.
□ Das Sortieren sehr kleiner Datenmengen.
□ Das Sortieren von Datensätzen mit wahlfreiem Zugriff.

c) Gegeben sei folgende Kostenmatrix eines Zuordnungsproblems. Jede Tätigkeit muss ausgeführt werden und jede Person kann nur eine Tätigkeit übernehmen. **Welche Person** sollte **welche Tätigkeit** übernehmen, um möglichst geringe Kosten zu verursachen? Wie hoch sind die **minimalen Kosten**? *(3 Punkte)*

C(i, j)	Tätigkeit A	Tätigkeit B	Tätigkeit C	Tätigkeit D
Person 1	1	12	4	5
Person 2	6	7	8	2
Person 3	14	4	6	8
Person 4	1	3	10	5

Aufgabe 8 *(10 Punkte)*

a) Gegeben sei nebenstehender AVL-Baum:



Fügen Sie nacheinander die Schlüssel **28**, **31** und **40** ein. Zeichnen Sie den Baum nach **jeder Einfügeoperation**. Falls Sie beim Einfügen rotieren müssen, zeichnen Sie den Baum auch **vor jeder einzelnen Rotation**. Geben Sie ferner jeweils an, **welche Art von Rotation** Sie **um welchen Knoten** ausführen. *(4 Punkte)*

b) In einen **leeren AVL-Baum** sollen folgende Schlüssel eingefügt werden: **1, 2, 3, 4, 5, 6, 7**. Geben Sie eine **Einfügereihenfolge** der Schlüssel an, die dazu führt, dass **keinerlei Rotationen** ausgeführt werden müssen und nach jeder Einfügeoperation ein **vollständiger Baum** vorliegt. *(2 Punkte)*

c) Zeichnen Sie alle gültigen 2-3-Bäume, welche die Schlüssel 4, 12, 15, 27, 33, 41, 48 enthalten. *(4 Punkte)*

Aufgabe 9 *(10 Punkte)*

Gegeben seien das Alphabet {C, D, E, F, H, I, L, N, R, S, Leerzeichen}, der Text „FRIERENDE FRIESEN FEIERN FRIEDLICH“ und das Suchmuster „FEIER“.

a) Erstellen Sie die entsprechende Shift-Tabelle des Horspool-Algorithmus. *(2 Punkte)*

b) Wie sucht der Horspool-Algorithmus das gegebene Suchmuster im Text? Zeichnen Sie dazu die Zeichenvergleiche und Verschiebungen auf. *(4 Punkte)*

c) Wie viele Verschiebungen und Zeichenvergleiche werden bei der Suche im gegebenen Beispiel durch den Horspool-Algorithmus ausgeführt? Wie sehen die Werte bei der Brute Force-Methode aus? Erläutern Sie jeweils Ihren Rechenweg. *(4 Punkte)*

Aufgabe 10 *(10 Punkte)*

Kreuzen Sie an, ob folgende Aussagen wahr oder falsch sind. (*Hinweis*: Inkorrekte Antworten führen *nicht* zu Abzügen.):

Aussage	*wahr*	*falsch*
1. Ein nicht-deterministischer Algorithmus kann determiniert sein.		
2. Die Suche in einer Hashing-Tabelle mit kleinem Load-Faktor verursacht im Mittel logarithmischen Aufwand.		
3. B-Bäume werden insbesondere dann eingesetzt, wenn die komplette Schlüsselmenge nicht mehr vollständig in den Hauptspeicher passt.		
4. Wenn sich der beste und schlechteste Fall eines Algorithmus nur um einen konstanten Faktor unterscheiden, kann man die Komplexitätsklasse des Algorithmus mittels der θ-Notation angeben.		
5. Jeder Algorithmus, der alle Teilmengen einer n-elementigen Menge ausgibt, verursacht mindestens faktoriellen Aufwand.		
6. Zwei unterschiedliche Binärbäume können dieselbe Inorder-Traversierungsreihenfolge besitzen.		
7. Mit Hilfe des Master-Theorems lässt sich für alle rekursiven Algorithmen die Komplexitätsklasse ermitteln.		
8. Implementiert man eine Prioritätswarteschlange auf Basis eines Heaps statt auf Basis einer Liste, können Elemente sowohl schneller eingefügt als auch schneller entnommen werden.		
9. Eine Hashing-Funktion muss nicht injektiv (linkseindeutig) sein.		
10. Verwendet man beim Hashing die offene Adressierung zur Kollisionsbehandlung, so sollte der Load-Faktor α möglichst nahe bei 1 liegen.		

Zusätzlicher Platz zur Aufgabenbearbeitung

Zusätzlicher Platz zur Aufgabenbearbeitung