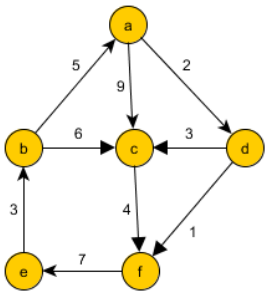
Name:_____ Matr.-Nr.:_____

Aufgabe 1 *(10 Punkte)*

Gegeben sei folgender Graph G:



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

b) G enthält Zyklen. Zählen Sie alle auftretenden **Zyklenlängen** auf und geben Sie jeweils ein Beispiel an. *(3 Punkte)*

c) Sei A die **Adjazenzmatrix** eines **ungerichteten**, **ungewichteten** Graphen. Erläutern Sie kurz, wie man anhand der Matrix folgende Grapheneigenschaften überprüfen kann. *(3 Punkte)*

- Der Graph ist vollständig.

- Der Graph enthält Schleifen (Loops).

- Der Graph besitzt einen isolierten Knoten (d. h. einen Knoten ohne Kanten).

Aufgabe 2 *(10 Punkte)*

Gegeben sei eine Zeichenkette der Länge n mit n > 0. Die Zeichenkette besteht nur aus Großbuchstaben und wird durch ein Array A[0, .., n-1] repräsentiert, wobei jedes Array-Element einen einzelnen Buchstaben darstellt.

Geben Sie einen **vollständigen Algorithmus in Pseudocode-Notation** an, der als Eingabe A erhält und bestimmt, ob die durch A repräsentierte Zeichenkette ein Palindrom ist. Ein Palindrom ist eine Zeichenkette, die vorwärts und rückwärtsgelesen identisch ist, z. B. „UHU“, „EBBE“, „KAJAK“, „RELIEFPFEILER“.

Hinweis: 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) Eine Liste von Datensätzen soll **nach Nachnamen** sortiert werden. Personen mit identischen Nachnamen sollen **zusätzlich nach ihrem Vornamen** sortiert sein. Beispiel:

Nachname	Vorname
Flanders	Ned
Simpson	Bart
Simpson	Homer
Simpson	Marge
Wiggum	Clancy

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

Mit Insertionsort zuerst nach Nachnamen und dann nach Vornamen sortieren.	□ ja	□ nein
Zuerst die Vornamen mit Quicksort und dann die Nachnamen mit Bubblesort sortieren.	□ ja	□ nein
Zuerst die Nachnamen mit Bubblesort und dann die Vornamen mit Countingsort sortieren.	□ ja	□ nein
Mit Mergesort zuerst nach Vornamen und dann nach Nachnamen sortieren.	□ ja	□ nein
Zuerst die Vornamen mit Bubblesort und dann die Nachnamen mit Heapsort sortieren.	□ ja	□ nein

b) Gegeben seien das Alphabet {A, G, I, K, N, P, U, Leerzeichen} und das Suchmuster „PINGUIN“. Füllen Sie die entsprechende **Shift-Tabelle** des Horspool-Algorithmus aus. *(2 Punkte)*

Buchstabe:	**A**	**G**	**I**	**K**	**N**	**P**	**U**	**_**
Verschiebung:								

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	4	11	5	1
Person 2	8	7	2	6
Person 3	6	4	7	13
Person 4	9	3	5	1

Aufgabe 4 *(10 Punkte)*

a) Gegeben sei untenstehender Algorithmus f(n). Bestimmen Sie für den Algorithmus die genaue **Laufzeitfunktion** und die **Komplexitätsklasse**. Verwenden Sie dabei als Basisoperation die **Multiplikation**. Unterscheiden Sie zwischen **Best- und Worst-Case**. *(6 Punkte)*

Hinweise:
- Die Funktion `sqrt(x)` liefert die Quadratwurzel von x
- Die Funktion `floor(x)` liefert die größte Ganzzahl, die kleiner oder gleich x ist.

```
ALGORITHM f(n)
// Input: Eine nicht-negative ganze Zahl n
i ← floor(sqrt(n))
while i > 0 do
    j ← 1
    while j * j ≤ n do
        print(i * j)
        j ← j + 1
    i ← i - 1
```

b) Zur Lösung eines Problems stehen zwei Algorithmen $F_1(n)$ und $F_2(n)$ 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 $T_1(n) = 3n^2 \log_2 n$ und $T_2(n) = 75n^2$. Welcher Algorithmus sollte aus Effizienzgründen **bei welcher Kundenzahl** verwendet werden? **Begründen** Sie Ihre Antwort kurz und geben Sie die **relevanten Rechenschritte** an. *(4 Punkte)*

Aufgabe 5 *(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 **Multiplikation**. Bestimmen Sie die genaue Laufzeitfunktion in geschlossener (d. h. nichtrekursiver) Schreibweise durch **rückwärtiges Einsetzen**. (Ein Beweis mittels vollständiger Induktion ist **nicht** notwendig.) Geben Sie außerdem die **Komplexitätsklasse** des Algorithmus an. *(8 Punkte)*

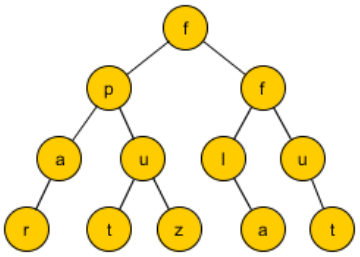
Hinweis: Die Funktion `floor(x)` liefert die größte Ganzzahl, die kleiner oder gleich x ist.

```
ALGORITHM f(n)
    // Input: Eine positive ganze Zahl n
    if n > 1 then
        result ← f(floor(n/2)) * f(floor(n/2))
        result ← result + f(floor(n/2)) * f(floor(n/2))
        return result
    else
        return 2 * n
```

b) Überprüfen Sie Ihre Berechnung der Komplexitätsklasse mit Hilfe des **Master-Theorems**. *(2 Punkte)*

Aufgabe 6 *(10 Punkte)*

a) Führen Sie an nebenstehendem Binärbaum eine **Postorder-Traversierung** durch und geben Sie die dabei entstehende Knotenreihenfolge an. *(3 Punkte)*



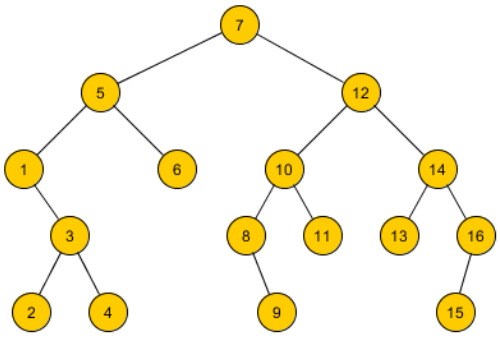
b) Zu einem Binärbaum B gehöre folgende **Inorder-Traversierungsreihenfolge**:

8, 2, 5, 9, 1, 7

Sie wissen zusätzlich, dass 9 das Wurzelelement von B ist. Welche der folgenden **Traversierungsreihenfolgen** könnten zu B gehören? Kennzeichnen Sie bei diesen Reihenfolgen den **Typ der Traversierung**. *(4 Punkte)*

Reihenfolge	Preorder	Postorder	weder noch
9, 7, 1, 2, 5, 8	☐	☐	☐
5, 8, 2, 7, 1, 9	☐	☐	☐
9, 5, 2, 8, 1, 7	☐	☐	☐
2, 5, 7, 1, 8, 9	☐	☐	☐

c) Löschen Sie in dem nebenstehenden sortierten Binärbaum die Schlüssel 1 und 12 gemäß dem **in der Vorlesung besprochenen Vorgehen**. Zeichnen Sie den resultierenden Baum **nachdem** beide Schlüssel gelöscht wurden. *(3 Punkte)*



Aufgabe 7 *(10 Punkte)*

a) Gegeben sei folgendes Array von Werten [4, 8, 3, 9, 5, 2, 7, 1]. Sortieren Sie die Werte in **absteigender** Reihenfolge mit Hilfe von ***S*elektionsort**: Geben Sie den Zustand des Arrays **nach jedem Schleifendurchlauf** an. *(4 Punkte)*

b) Gegeben sei folgendes Array von Werten [3, 6, 1, 8, 4, 5]. Sortieren Sie die Werte in **absteigender** Reihenfolge mit Hilfe von **Heapsort**. Stellen Sie dazu in beiden Phasen den Zustand des Arrays **nach jedem Arbeitsschritt** dar. *(6 Punkte)*

Aufgabe 8 *(10 Punkte)*

Gegeben sei folgendes Rucksackproblem:

Rucksackkapazität W = 6 kg

Gegenstand	Gewicht	Wert
A	4 kg	6.000 Euro
B	3 kg	4.200 Euro
C	1 kg	1.900 Euro
D	2 kg	2.000 Euro

Lösen Sie das Problem mit Hilfe der Dynamischen Programmierung:

- Erstellen und beschriften Sie die **Tabelle V(i, j)**.
- Füllen Sie anschließend die Tabelle aus.
- Geben Sie die **Menge der eingepackten Gegenstände** an und **kennzeichnen** Sie, wie diese Gegenstände mithilfe der Tabelle bestimmt werden können.

Aufgabe 9 *(10 Punkte)*

a) Gegeben sei ein leerer 2-3-Baum. Fügen Sie nacheinander die Schlüssel 20, 32, 8, 42, 59, 15, 3, 23 ein und zeichnen Sie den entstehenden Baum **nach jeder Einfügeoperation**. *(4 Punkte)*

b) Könnten die Schlüsselwerte aus Aufgabenteil a) in einer anderen Reihenfolge nacheinander eingefügt werden, sodass ein 2-3-Baum **der Höhe 1** entsteht? Falls ja, geben Sie ein Beispiel für eine solche Reihenfolge an. *(2 Punkte)*

c) Welche **Höhe** muss ein B-Baum der Ordnung m = 10 **mindestens** besitzen, um 10.000 Schlüsselwerte aufnehmen zu können? Skizzieren Sie Ihren Rechenweg. *(2 Punkte)*

d) Welche **Ordnung** m muss ein B-Baum **mindestens** besitzen, damit bei 10.000 Schlüsselwerten **höchstens ein Baum der Höhe 2** entsteht? Skizzieren Sie Ihren Rechenweg. *(2 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 Algorithmus muss weder deterministisch noch determiniert sein.		
2. Die Breitensuche ist geeignet, um die kürzesten Pfade zwischen dem Startknoten und allen anderen Knoten zu finden.		
3. Jeder Algorithmus, der alle Permutationen einer n-elementigen Menge ausgibt, verursacht mindestens faktoriellen Aufwand.		
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. Greedy-Algorithmen liefern immer nur Näherungslösungen.		
6. Wenn vier von den insgesamt sechs Knoten eines DAG Einstiegsknoten sind, dann besitzt der DAG höchstens neun Kanten.		
7. Die Höhe von sortierten Binärbäumen kann höchstens logarithmisch zur Anzahl der Schlüsselwerte wachsen.		
8. Bei einem AVL-Baum kann sich die Länge der Pfade von der Wurzel zu den Blättern insgesamt höchstens um den Wert 1 unterscheiden.		
9. Eine Hashing-Funktion sollte surjektiv (rechtstotal) 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