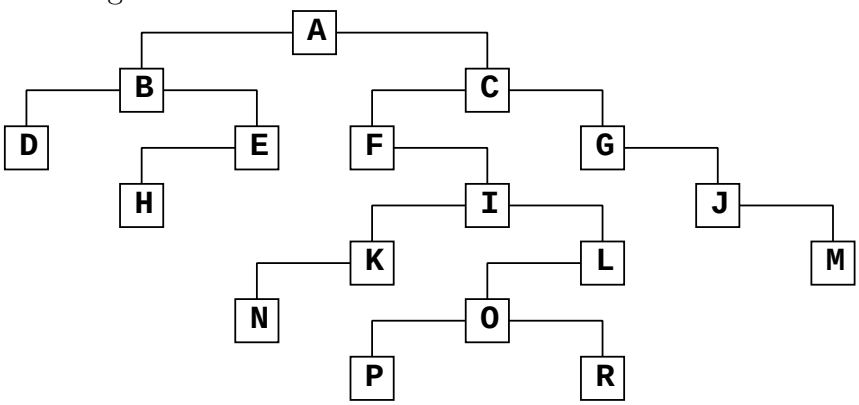
Name: Musterlösung *Matrikelnummer:*

Aufgabe 1: Suchbäume **(14 Punkte)**

Betrachten Sie den folgenden Suchbaum.



(a) (1 Punkt) Geben Sie die Höhe des Knotens F an.

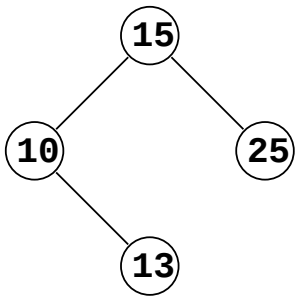
(b) (1 Punkt) Geben Sie die Tiefe des Knotens F an.

(c) (1 Punkt) Zeichnen Sie einen maximal unausgewogenen AVL-Baum der Höhe 4.

Punkte	

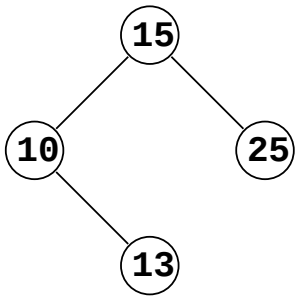
Name: Musterlösung *Matrikelnummer:*

(d) (1 Punkt) Zeichnen Sie den AVL-Baum nach Einfügen eines Knotens mit dem Schlüssel 12 in den gegebenen Baum. Geben Sie an, ob und welche Art der Rotation Sie durchgeführt haben.



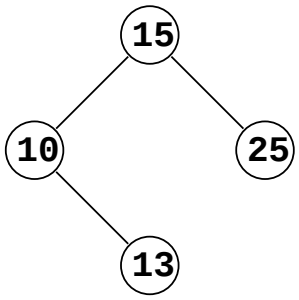
Rotation(en):

(e) (1 Punkt) Zeichnen Sie den AVL-Baum nach Einfügen eines Knotens mit dem Schlüssel 8 in den gegebenen Baum. Geben Sie an, ob und welche Art der Rotation Sie durchgeführt haben.



Rotation(en):

(f) (1 Punkt) Zeichnen Sie den AVL-Baum nach Löschen des Knotens mit dem Schlüssel 25 aus dem gegebenen Baum. Geben Sie an, ob und welche Art der Rotation Sie durchgeführt haben.

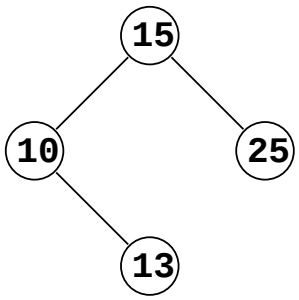


Rotation(en):

Punkte	

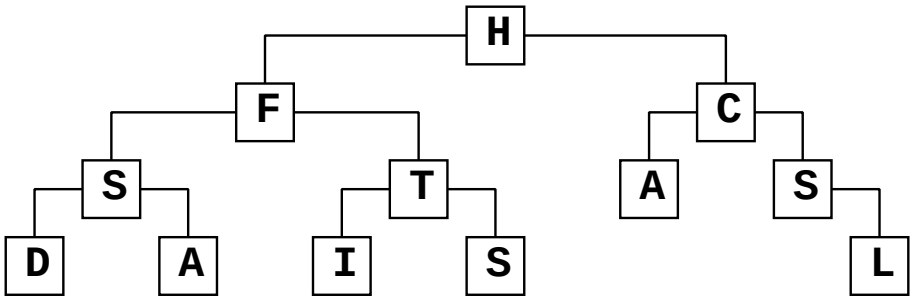
Name: Muster *Matrikelnummer:*

(g) **(1 Punkt)** Zeichnen Sie den AVL-Baum nach Löschen des Knotens mit dem Schlüssel 15 aus dem gegebenen Baum. Geben Sie an, ob und welche Art der Rotation Sie durchgeführt haben.



Rotation(en):

(h) **(1 Punkt)** Traversieren Sie den folgenden Baum in `preorder`-Reihenfolge und geben Sie den entstehenden String an.



Punkte	

Musterlösung

(i) (6 Punkte) Implementieren Sie eine Methode

```
public int insertNode(Node root, Node neuerKnoten);
```

zum Einfügen eines Knotens in einen knotenorientierten binären Suchbaum. Die Methode bekommt als Argumente den Wurzelknoten und den einzufügenden Knoten übergeben und soll den Wert -1 zurückliefern, wenn der Schlüssel des neuen Knotens schon im Baum enthalten ist oder die Tiefe des Knotens, wenn er erfolgreich eingefügt wurde. Falls der Baum leer ist, ist das `data`-Feld des Wurzelknotens auf -1 gesetzt.

Der Knoten ist durch folgende Klasse gegeben:

```
public class Node {
  public int  data;    // Der Schluessel, "-1" entspricht leer
  public Node left;    // linker Unterbaum
  public Node right;   // rechter Unterbaum
}
```

```
public int insertNode(Node root, Node neuerKnoten) {
```

Punkte	

Name: *Matrikelnummer:*

Muster

```
}
```

Punkte	

Name: Musterlösung *Matrikelnummer:*

Aufgabe 2: Hashing **(15 Punkte)**

(a) (1 Punkt) Worin besteht der Unterschied zwischen der Kollisionsauflösung bei Hashtabellen mittels *Open Adressing* und *Separate Chaining*?

(b) (1 Punkt) Nennen Sie zwei Eigenschaften einer guten Hashfunktion.

(c) (2 Punkte) Ist beim Einfügen mittels *double hashing* (schlüsselabhängiges Sondieren) in eine Hashtabelle die Anzahl der auftretenden Kollisionen abhängig von der Reihenfolge der eingefügten Datensätze? Begründen Sie ihre Antwort (gegebenenfalls mit einem kurzen Beispiel)!

Punkte	

Muster

(d) (6 Punkte) Gegeben sind drei verschiedene Hashtabellen, die jeweils durch Einfügen (in der angegebenen Reihenfolge) der Elemente mit ihrem zugehörigem Schlüssel

```
(Michael,  5), (Bettina , 11), (Uwe , 9), (Daniela, 10)
```

entstanden sind. Tragen Sie jeweils in das Kästchen unter der Hashtabelle die Nummer des verwendeten Algorithmus ein.

Als Algorithmen stehen zur Auswahl:

1. $\mathrm{h}(k) = k \bmod 4$ und mit *linearem Sondieren* (Inkrement $b = 1$)
2. *double hashing* mit $\mathrm{h}(k) = k \bmod 4$ und $\mathrm{a}(k) = k + 1$
3. *separate chaining* mit $\mathrm{h}(k) = k \bmod 4$
4. *double hashing* mit $\mathrm{h}(k) = k \bmod 4$ und $\mathrm{a}(k) = k + 2$
5. $\mathrm{h}(k) = (k * 3) \bmod 4$ und mit *linearem Sondieren* (Inkrement $b = 1$)
6. keiner der hier genannten Algorithmen (nur wenn kein hier angegebener die Hashtabelle erzeugen kann!)

Index	**Name der Person**
0	Uwe
1	Michael
2	Daniela
3	Bettina
Algorithmus:	

Index	**Name der Person**
0	Uwe
1	Bettina
2	Daniela
3	Michael
Algorithmus:	

Index	**Name der Person**
0	
1	Michael
2	Daniela
3	Bettina
Algorithmus:	

Hinweis: Der Eintrag `Uwe` konnte nicht eingefügt werden.

Punkte	

Musterlösung

(e) (5 Punkte) Implementieren Sie eine Java-Funktion

```
public int insertElement(Data D);
```

die einen Datensatz D mit *double hashing* in eine Hashtabelle einfügt. Die Funktion soll die Anzahl der Kollisionen zurückliefern oder den Wert -1, falls das Eintragen fehlschlug. Eine Hilfsfunktion `public int calculateKey(Data D);` sei gegeben. Sie berechnet aus einem Datensatz den dazugehörigen Schlüssel und liefert ihn im Rückgabewert zurück. Die Hashfunktion lautet $\mathrm{h}(k) = k \bmod M$ (mit M gleich Größe der Hashtabelle); die Inkrementfunktion soll $\mathrm{a}(k) = k * k + k + 1$ sein. Die Hashtabelle ist als `Data HashTable[256];` deklariert.

```
public int insertElement(Data D) {

}
```

Punkte	

Name: Musterlösung *Matrikelnummer:*

Aufgabe 3: Optimierung **(13 Punkte)**

(a) (2 Punkte) Kreuzen sie alle richtigen Aussagen an.

Für alle Probleme der Klasse NP gilt:

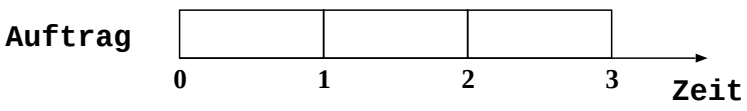
- ◯ Sie können nicht in der Klasse P enthalten sein.
- ◯ Eine Lösung kann deterministisch in polynomieller Zeit verifiziert werden.
- ◯ Sie sind nicht mit einem Computer lösbar.
- ◯ Eine Lösung kann nichtdeterministisch in polynomieller Zeit berechnet werden.

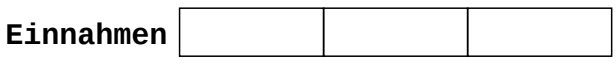
(b) (2 Punkte) Beschreiben sie kurz die Idee, die bei *Dynamischer Programmierung* benutzt wird.

(c) (5 Punkte) Ein Großrechner wird für Rechenaufgaben vermietet. Jedes Programm benötigt genau *eine* Zeiteinheit (der Zeitbedarf ist also für alle Aufträge identisch). Die Programmläufe müssen bis zu einem bestimmten Zeitpunkt beendet sein und bringen unterschiedliche Einnahmen. Konnten die Programme nicht termingerecht ausgeführt werden, so bringen sie keine Einnahmen und können verworfen werden. Da es mehr Aufträge als Rechenzeit gibt, sucht der Betreiber die Belegung, die am meisten Einnahmen bringt.

Finden Sie eine optimale Belegung, wenn aktuell folgende Aufträge vorliegen.

Auftrag	Einnahmen	Endzeitpunkt (Deadline) (in Zeiteinheiten ab jetzt)
1	80	3
2	60	1
3	50	3
4	70	1
5	40	2





Punkte	

Name: Musterlösung *Matrikelnummer:*

(Fortsetzung 3c)

Das Problem ist greedy-lösbar. Skizzieren Sie (z.B. verbal oder in Pseudocode) einen Greedy-Algorithmus, der die optimale Belegung findet.

Punkte	

Name: *Matrikelnummer:*

Musterlösung

(d) **(4 Punkte)** Skizzieren Sie (z.B. verbal oder in Pseudocode) einen möglichen Algorithmus, der folgendes Problem mit Backtracking löst:

Gesucht ist eine spezielle Permutation $(x_1, \ldots, x_9)$ der Ziffern $1, \ldots, 9$, die folgende Bedingung erfüllt:

$$\forall i \text{ mit } 0 < i < 10: \quad i \text{ teilt } \sum_{j=1}^{i} x_j * 10^{i-j}$$

Hinweis: Eine Lösung ist zum Beispiel die Zahl 381654729. Zum einen kommt jede Ziffer genau einmal vor und zum anderen gilt:

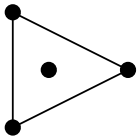
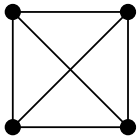
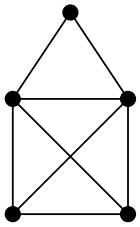
```
1 teilt 3
2 teilt 38
3 teilt 381
4 teilt 3816
5 teilt 38165
6 teilt 381654
7 teilt 3816547
8 teilt 38165472
9 teilt 381654729
```

Punkte	

Name: Muster *Matrikelnummer:*

Aufgabe 4: Graphen **(8 Punkte)**

(a) (3 Punkte) Kreuzen Sie alle zutreffenden Aussagen zu den Eigenschaften der folgenden Graphen an.

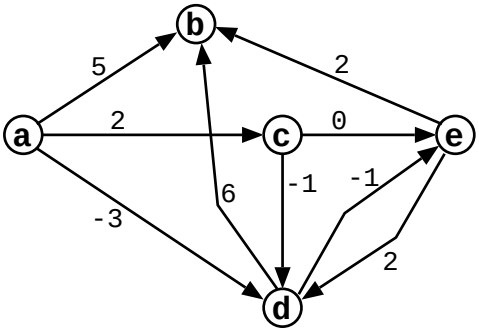
Graph	ist vollständig	enthält (mindestens) eine 3-Clique	ist bipartit
	◯ ja ◯ nein	◯ ja ◯ nein	◯ ja ◯ nein
	◯ ja ◯ nein	◯ ja ◯ nein	◯ ja ◯ nein
	◯ ja ◯ nein	◯ ja ◯ nein	◯ ja ◯ nein
$G = (V, E)$ ungerichtet mit $V = \{a, b, c, d, e\}$ und $E = \{(a, b), (a, d), (c, d), (b, c), (d, e)\}$	◯ ja ◯ nein	◯ ja ◯ nein	◯ ja ◯ nein
$G = (V, E)$ ungerichtet mit $V = \{a\}, E = \emptyset$	◯ ja ◯ nein	◯ ja ◯ nein	◯ ja ◯ nein

Hinweis: Eine n-Clique ist eine Clique mit n Knoten.

Punkte	

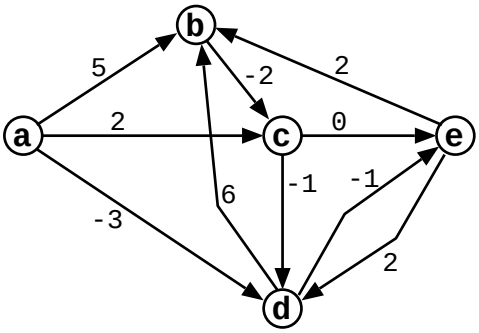
Name: Muster *Matrikelnummer:*

(b) **(1 Punkt)** Welcher der in der VL vorgestellten Algorithmen ist geeignet, um im folgenden Graphen die kürzesten Wege vom Startknoten **a** zu allen anderen Knoten zu finden? Begründen Sie Ihre Auswahl kurz.



(c) **(1 Punkt)** Geben Sie für den in Aufgabe 4b) von Ihnen ausgewählten Algorithmus den Aufwand im $\mathcal{O}$-Kalkül in Abhängigkeit von der Zahl der Knoten (V) und Kanten (E) des Eingabegraphen an.

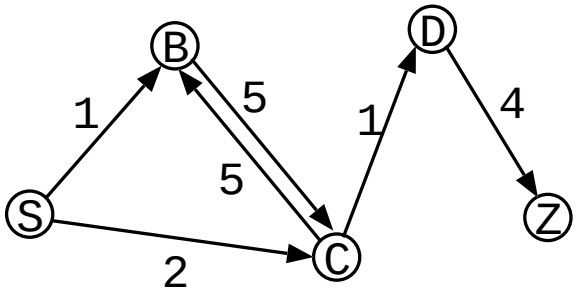
(d) **(1 Punkt)** Kann der von Ihnen in Aufgabe 4b) ausgewählte Algorithmus die kürzesten Wege im folgenden Graphen finden (Startknoten sei Knoten **a**)? Begründen Sie Ihre Antwort!



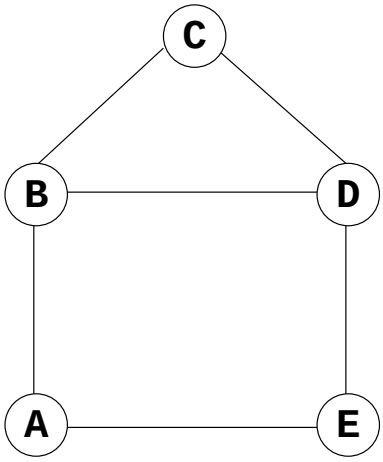
Punkte	

Name: Muster *Matrikelnummer:*

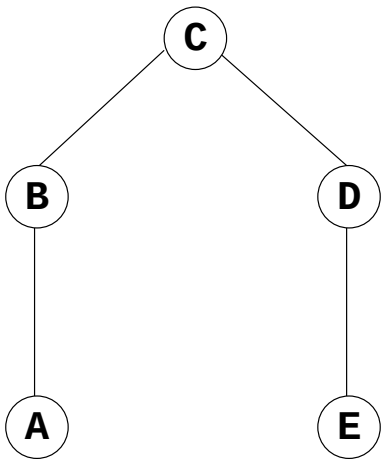
(e) (1 Punkt) Geben Sie den folgenden Graphen in Adjazenzlisten-Darstellung an.



(f) (1 Punkt) Gegeben sei folgender ungerichteter Graph G und ein spannender Baum T. Dieser wurde durch Ablaufen des Graphen G mittels *Tiefensuche* erzeugt.



Graph G



Baum T

Geben Sie den Startknoten und die Reihenfolge der Knoten an, in der die Tiefensuche diese besucht hat. Hinweis: Gleiche Knoten haben den gleichen Namen.

Startknoten:
Knotenreihenfolge:

Punkte	