Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2014

46116

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Softwaretechnologie/Datenbanksysteme**

Anzahl der gestellten Themen (Aufgaben): **2**

Anzahl der Druckseiten dieser Vorlage: **11**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

Aufgabe 1: „OOA/OOD"

Betrachten Sie folgende Spezifikation einer Stammbaum-App:

Ein Stammbaum besteht aus Personen, die Verbindungen miteinander haben. Man kann einen Stammbaum laden oder speichern, wofür man jeweils auch den Pfad als Zeichenkette angibt - die Methode „speichern" gibt zurück, ob der Stammbaum erfolgreich gespeichert werden konnte. Dem Stammbaum kann man beliebig viele Personen oder Verbindungen hinzufügen; ebenso kann man eingetragene Personen bzw. Verbindungen auch wieder löschen. Eine Person bzw. eine Verbindung kann nur in genau einem einzigen Stammbaum vorkommen.
Beim Anlegen einer Person können Vorname, Nachname, Geburtsdatum und optional (sofern überhaupt schon bekannt) auch das Sterbedatum angegeben werden. Das Sterbedatum kann nachträglich gesetzt werden, wobei ein evtl. schon eingetragenes Datum zurückgegeben wird. Einer Person wird zunächst genau ein vorgegebenes Bild zugeordnet, das aber jederzeit geändert werden kann. Die Angaben zur Person sind privat.
Zwischen genau zwei Personen kann eine der drei Beziehungen bestehen: Verliebt, Verlobt, Verheiratet. Eine abstrakte Beziehung kann es im Stammbaum zwar nicht geben, aber zu jeder der drei genannten Beziehungen kann man jeweils das Datum angeben, von wann und bis wann sie gehalten hat (muss man aber nicht eintragen!) und welches Personen-Paar darin verwickelt war. Selbstverständlich kann jede Person beliebig viele Beziehungen gleichzeitig haben. Bei Verliebten kann man noch angeben, ob beide glücklich sind (könnte ja sein, dass Romeo zwar Julia liebt, aber bei Julia bereits ein Sterbedatum eingetragen werden konnte...). Verliebte können sich trennen, eine Verlobung kann man auflösen und ist man erst verheiratet, dann lässt man sich ggf. auch mal scheiden - wir sagen dazu: eine Beziehung kann man kaputt machen, wozu dann aber das Datum angegeben werden muss, bis zu dem die Beziehung gehalten hat.
Eine ganz andere Verbindung stellt ein Kindschaftsverhältnis dar. Dazu gehören stets eine Mutter und ein Kind, allerdings kann der tatsächliche Vater nicht immer zugeordnet werden (der Kerl ist dann ja wohl „eine ziemliche Null"). Auch werden Zeugungsdatum und Zeugungsort möglicherweise erst später bekannt, so dass man sie nachträglich gemeinsam setzen kann. Eine Person kann offensichtlich beliebig viele öffentlich bekannte Kindschaftsverhältnisse haben und auch die Mutter-/Vater-/Kind-Rolle wird publik gemacht.

Erstellen Sie aus der obigen „Spezifikation" ein UML-Klassendiagramm mit allen Klassen, Vererbungsrelationen, Attributen, Methoden (auch Konstruktoren) und Assoziationen. Tragen Sie bei Assoziationen stets auch die Multiplizitäten und Navigationspfeile ein. Falls Rollennamen bekannt sind, so müssen diese ebenfalls angegeben werden. Achten Sie ferner auf die Sichtbarkeiten von Methoden und Attributen.
Als UML-Klassen explizit (graphisch) modellierte Typen sollen nicht als Attribute im Klassenkästchen erscheinen. Nehmen Sie hierbei aber an, dass `Datum` und `Pixel` sowie `String` vorgegebene („primitive") Datentypen sind, die nicht mehr als explizite UML-Klasse eingezeichnet werden müssen. Verwenden Sie *möglichst restriktive,* aber sinnvolle Sichtbarkeiten, sofern der obige Text nichts Genaueres spezifiziert.

Fortsetzung nächste Seite!

Aufgabe 2: „Abstrakte Datentypen (ADT)“

Gegeben sei folgender Ansatz eines abstrakten Datentyps (ADT) *SortedSet*, der eine Menge von Objekten eines generischen Typs `T` verwaltet. Auf `T` ist außerdem eine Ordnung „$\leq$ “ sowie die Relation „= “ mit der üblichen Semantik definiert. Das spezielle Element `nil` ist (ähnlich der **`null`**-Referenz in Java) ein Hilfskonstrukt und soll als Element vom Typ `T` verstanden werden – berücksichtigen Sie diesen Sonderfall in Ihren Axiomen, sodass `nil` nicht zur Menge hinzugefügt werden kann.

adt *SortedSet*
sorts *SortedSet, T, int, boolean*
ops

create:		$\rightarrow$ *SortedSet*	// Konstruktor
add:	*SortedSet* $\times$ *T*	$\rightarrow$ *SortedSet*	
...			

axs
...
end *SortedSet*

a) Ergänzen Sie den vorgegebenen Ausschnitt der algebraischen Spezifikation des ADTs **`SortedSet`** um die notwendigen Signaturen und Axiome folgender Operationen:

   i) **`remove`**: entfernt ein als Argument übergebenes Element aus der Menge, sofern darin vorhanden – andernfalls bleibt die Menge unverändert

   ii) **`contains`**: prüft, ob ein als Argument übergebenes Element in der Menge enthalten ist und gibt einen Wahrheitswert zurück

   iii) **`size`**: gibt die Anzahl der in der Menge enthaltenen Elemente zurück

Vereinfachen Sie folgenden Ausdruck schrittweise soweit wie möglich (also bringen Sie ihn in die kanonische Normalform) und geben Sie jeweils an, welches Ihrer Axiome Sie zur Transformation angewandt haben:

$$contains(add(add(remove(add(add(create, 42), 42), 42), 666), 42), 42)$$

b) Ergänzen Sie Signatur und Axiome, um folgende Operationen zu modellieren:

   i) **`front`**: gibt das kleinste Element der Menge zurück; ist die Menge leer, soll **`nil`** zurückgegeben werden

   ii) **`pop`**: entfernt das kleinste Element der Menge, sofern eines vorhanden ist

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1: Modellierung

Zur digitalen Verwaltung deutscher Handballvereine soll eine zweiteilige Datenbank eingerichtet werden. Der erste Teil soll die Mannschaften, Spieler und Trainer beinhalten und speichert dafür folgende Informationen:

- Für eine Mannschaft wird der Name und die Stadt, zu der diese gehört, gespeichert.
- Logos werden durch ihre Form beschrieben. Jede Mannschaft besitzt genau ein Logo.
- Ein Spieler besitzt Informationen über seine Spieler-ID, seinen Vor- und Nachnamen. Eine Mannschaft hat eine beliebige Anzahl an Spielern. Umgekehrt kann ein Spieler aber nur in einer Mannschaft spielen.
- Trainer besitzen eine Funktion (z.B. Trainer, Co-Trainer, Konditionstrainer, ...), sowie Vor- und Nachnamen. Ein Trainer trainiert genau eine Mannschaft und soll in seiner Rolle nur in Zusammenhang mit der zugehörigen Mannschaft bestehen und eindeutig identifiziert werden können. Eine Mannschaft kann mehrere Trainer besitzen.

Im zweiten Teil der Datenbank werden Trainingsdaten erfasst. Diese soll folgende Daten umfassen:

- Ein Trainingsblock wird durch seinen zugehörigen Namen bestimmt. Dieser Block hat mehrere einzelne Trainingseinheiten als Teile. Eine Trainingseinheit wird durch eine Einheits-ID identifiziert.
- Trainingseinheiten unterteilen sich in Konditions- und Spielzugtraining. Spielzugtraining kann auf einer oder mehreren anderen Spielzugtrainingseinheiten aufbauen.
- Zu einer Halle wird deren Name und die Stadt, in der sie sich befindet, gespeichert.
- Eine Trainingseinheit wird von einem Trainer in einer Halle gehalten.

a) Entwerfen Sie für das Szenario des **ersten** Teils der Datenbank ein ER-Diagramm in Chen-Notation. Bestimmen Sie hierzu:
   - die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
   - die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
   - die Funktionalitäten der Relationship-Typen.

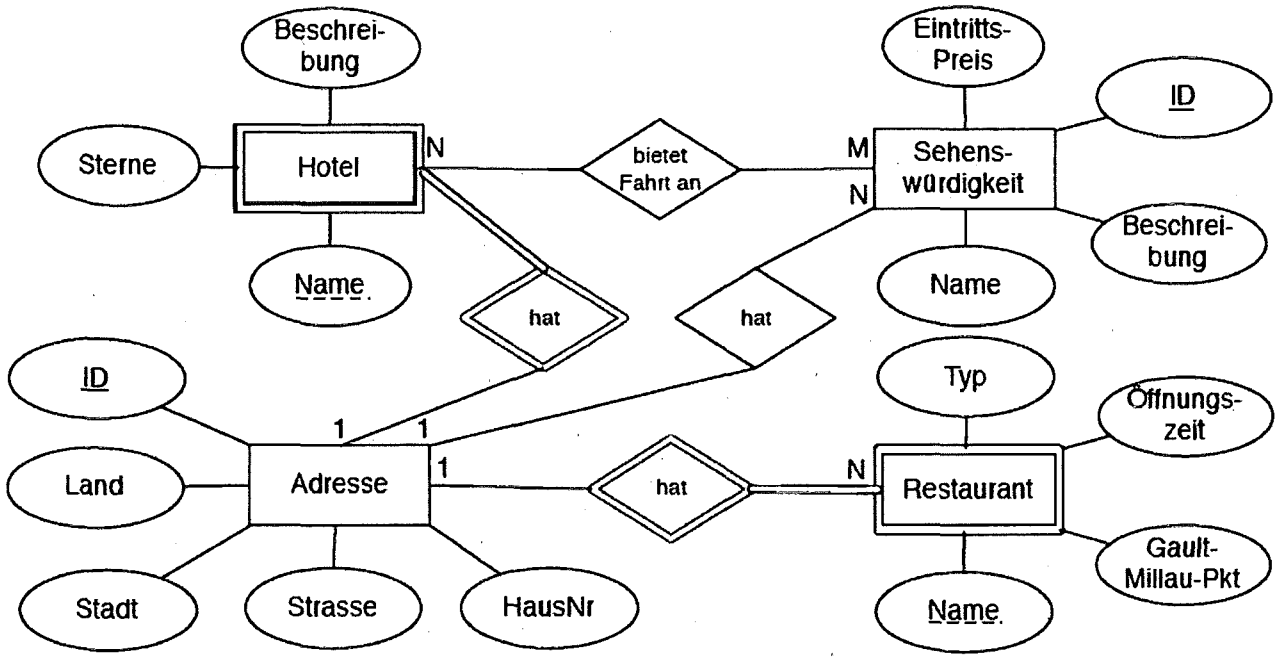
b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

c) Übernehmen Sie den Entity-Typen für einen Trainer aus Aufgabenteil a) (Speichern Sie hier als Primärschlüssel eine ID) und entwerfen Sie ausgehend davon ein weiteres ER-Modell für den **zweiten** Teil der Datenbank. Bestimmen Sie dazu ebenfalls dieselben Kriterien wie in Aufgabenteil a).

Fortsetzung nächste Seite!

Aufgabe 2: SQL-Anfragen

Gegeben sei folgendes ER-Modell, welches eine Datenbank für die App *TravelGuide 2.0* beschreibt. Darin sind Hotels, Sehenswürdigkeiten, Restaurants sowie deren Adressen enthalten. Der Einfachheit halber können Sie davon ausgehen, dass ein Hotel, eine Sehenswürdigkeit oder ein Restaurant exakt eine Adresse besitzt. Des Weiteren bieten Hotels Fahrten zu verschiedenen Sehenswürdigkeiten an.



Gehen Sie von folgendem relationalen Schema aus:

Hotel {[Name, Adresse_ID, Sterne, Beschreibung]}
Sehenswürdigkeit {[ID, Name, Beschreibung, Eintrittspreis]}
Adresse {[ID, Land, Stadt, Strasse, HausNr]}
Restaurant {[Name, Adresse_ID, Typ, Öffnungszeit, Gault-Millau-Pkt]}
Bietet Fahrt An {[Hotel Name, Adresse ID, Sehenswürdigkeit ID]}

Formulieren Sie folgende Anfragen in SQL. Achten Sie falls nötig auf Duplikat-freie Ausgaben:

a) Geben Sie die Beschreibung und die Sterne-Anzahl aller Hotels, die den Namen „Hilton" tragen, aus.

b) Geben Sie die Namen aller italienischen Restaurants der Stadt München aus.

c) Geben Sie die Namen aller 4-Sterne Hotels in Berlin aus, die eine Fahrt zur Sehenswürdigkeit „Schloss Sans Souci" anbieten.

d) Geben Sie den Namen der Stadt mit den besten Restaurants, gemessen am Durchschnitt ihrer Gault-Millau-Punkte, aus.

e) Geben Sie die Straßennamen und die dazugehörigen Städte aus, an denen mehr als ein Hotel angesiedelt sind.

Fortsetzung nächste Seite!

Aufgabe 3: Normalformen

Eine IT-Beratungsfirma stellt zwei Unternehmen vorübergehend Experten zur Verfügung, die an gemeinsamen IT-Projekten arbeiten. Die folgende Tabelle (Relation S) listet die Unternehmen sowie die Zeit, die jeder Experte bei ihnen verbringt, auf. Außerdem ist jeder Experte durch eine **eindeutige** Nummer (Attribut *Experte ID*) identifiziert:

Experte ID	**Vertrag Nr**	**Stunden**	**Expertenname**	**Unternehmen ID**	**Unternehmensort**
E12345	V100	36	M. Schneider	U810	München
E6789	V100	60	K. Schmitt	U810	München
E2468	V500	40	W. Mayer	U920	Berlin
E12345	V500	40	M. Schneider	U920	Berlin

a) Bestimmen Sie alle funktionalen Abhängigkeiten von S.

b) Bestimmen Sie alle Kandidatenschlüssel von S.

c) Welcher Normalform genügt die Relation S (1NF, 2NF, 3NF, BCNF)? Begründen Sie Ihre Aussagen!

d) Bestimmen Sie eine Zerlegung von S in 3NF. Verwenden Sie hierfür den Synthesealgorithmus.

e) Erstellen Sie aus dem 3NF der Relation S das entsprechende Relationenschema. Bestimmen Sie für jede Tabelle den **Primärschlüssel** und, wenn nötig, **Fremdschlüssel** mit den entsprechenden Verknüpfungen (markieren Sie Fremdschlüssel-Beziehungen mit einem *).

Aufgabe 4: Transaktionen

Gegeben seien die beiden folgenden, aus elementaren Operationen der Transaktionen T_2, T_2 und T_3 bestehenden Schedules:

$S_1 = r_1(a)\ r_2(b)\ r_3(a)\ w_3(a)\ r_1(b)\ w_1(b)\ r_2(c)\ r_1(c)\ r_2(d)\ r_1(d)\ w_1(d)\ c_1\ w_3(c)\ c_2\ c_3$

$S_2 = r_1(c)\ r_2(b)\ w_1(a)\ w_2(b)\ r_1(b)\ r_2(c)\ w_1(b)\ w_2(c)\ c_2\ c_1$

a) Geben Sie den Konfliktgraphen für S_1 und S_2 an. Wie kommen die Knoten und Kanten zustande?

b) Sind die beiden Schedules S_1 und S_2 serialisierbar? Begründen Sie Ihre Antwort kurz. Falls S_1 und/oder S_2 serialisierbar sind, geben Sie jeweils einen äquivalenten seriellen Plan an.

c) Benennen und erläutern Sie die drei klassischen Synchronisationsprobleme.

Thema Nr. 2

Teilaufgabe 1:

Aufgabe 1: *Allgemeine SWT, Vorgehensmodelle und Requirements Engineering*

Kreuzen Sie für die folgenden Multiple-Choice-Fragen genau die richtigen Antworten deutlich an. Es kann mehr als eine Antwort richtig sein.
Jedes korrekt gesetzte oder korrekt nicht gesetzte Kreuz wird mit 1 Punkt gewertet. Jedes falsch gesetzte oder falsch nicht gesetzte Kreuz wird mit -1 Punkt gewertet. Eine Frage kann entwertet werden, dann wird sie nicht in der Korrektur berücksichtigt. Einzelne Antworten können nicht entwertet werden. Entwerten Sie eine Frage wie folgt:

	Fragen
1.	Frage 1 ☐ Antwort 1 ☐ Antwort 2

Die gesamte Aufgabe wird nicht mit weniger als 0 Punkten gewertet.

	Fragen
1.	Welche Aussage ist wahr? ☐ Je früher ein Fehler entdeckt wird, umso teurer ist seine Korrektur. ☐ Je später ein Fehler entdeckt wird, umso teurer ist seine Korrektur. ☐ Der Zeitpunkt der Entdeckung hat keinen Einfluss auf die Kosten.
2.	Mit welcher Methodik können Funktionen spezifiziert werden? ☐ Als Funktionsvereinbarung in einer Programmiersprache ☐ Mit den Vor- und Nachbedingungen von Kontrakten ☐ Als Zustandsautomaten
3.	Welche Vorgehensmodelle sind für Projekte mit häufigen Änderungen geeignet? ☐ Extreme Programming (XP) ☐ Das V-Modell 97 ☐ Scrum
4.	Welche der folgenden Aussagen ist korrekt? ☐ Mittels Prototyping versucht man die Anzahl an nötigen Unit-Tests zu reduzieren. ☐ Ein Ziel von Prototyping ist die Erhöhung der Qualität während der Anforderungsanalyse. ☐ Mit Prototyping versucht man sehr früh Feedback von Stakeholdern zu erhalten.
5.	Welche der folgenden Aussagen ist korrekt? ☐ Bei der Architektur sollten funktionale und nicht-funktionale Anforderungen beachtet werden. ☐ Bei der Architektur sollten nur funktionale Anforderungen beachtet werden. ☐ Bei der Architektur sollten nur nicht-funktionale Anforderungen beachtet werden. ☐ Bei der Architektur sollte auf die mögliche Änderungen von Komponenten geachtet werden.

Fortsetzung nächste Seite!

Aufgabe 2: Vorgehensmodelle

Software wird oft in definierten Prozessen entwickelt. Diese nennt man Vorgehensmodelle.

Allgemein

a) Was sind die Aufgaben eines Vorgehensmodells im Allgemeinen?

b) Was sind die wesentlichen Bestandteile eines Vorgehensmodells und in welcher Beziehung stehen diese zueinander?

Ein frühes Vorgehensmodell ist das von Dr. Winston Royce 1970 formalisierte Wasserfallmodell.

c) Geben Sie eine schematische Darstellung des Wasserfallmodells an.

d) Nennen Sie zwei Probleme des Modells und erläutern Sie diese kurz.

e) In welchen Situationen lässt sich das Wasserfallmodell gut einsetzen?

Barry Boehm erweiterte das Wasserfallmodell 1979 zum so genannten V-Modell.

f) Geben Sie eine schematische Darstellung des V-Modells an.

g) Nennen Sie zwei Probleme des Modells und erläutern Sie diese kurz.

h) Welchen Vorteil hat das V-Modell gegenüber dem Wasserfallmodell?

In neuerer Zeit finden immer häufiger iterative und inkrementelle Vorgehensweisen Anwendung.

i) Erklären Sie den Begriff iterative Softwareentwicklung.

j) Erklären Sie den Begriff inkrementelle Softwareentwicklung und grenzen Sie ihn von iterativer Softwareentwicklung ab.

k) Nennen Sie jeweils zwei Vor- und Nachteile eines iterativen und inkrementellen Vorgehens im Vergleich zum Wasserfallmodell.

Fortsetzung nächste Seite!

Aufgabe 3: UML Diagramme in der Anwendung

Gegeben sei folgender Sachverhalt:

> Für eine Verwaltungssoftware einer Behörde soll ein Bestellsystem entwickelt werden. Dabei sollen die Nutzer ihre Raummaße eingeben können. Anschließend können die Nutzer über ein Web-Interface das Büro gestalten und Möbel (wie zum Beispiel Wandschränke) und andere Einrichtungsgegenstände in einem virtuellen Büro platzieren. Aus dem Web-Interface kann die Einrichtung dann direkt bestellt werden. Dazu müssen die Nutzer ihre Büro-Nummer und den Namen und die Adresse der Behörde eingeben und die Bestellung bestätigen.
>
> Weiterhin können Nutzer auch Büromaterialien über das Web-Interface bestellen. Dazu ist anstatt der Eingabe der Raummaße nur das Eingeben von Büro-Nummer und des Namens und der Adresse der Behörde erforderlich.
>
> Zusätzlich zum Standard-Nutzer können sich auch Administratoren im System anmelden und Möbel zur Kollektion hinzufügen und aus der Kollektion entfernen. Die Möbel können eindeutig durch ihre Inventurnummer identifiziert werden.
>
> Um jegliche Veränderungen im System protokollieren zu können müssen Nutzer und Administratoren zur Bestätigung eingeloggt sein.

a) Erfassen Sie die drei Systemfunktionen *Möbel bestellen*, *Login* und *Kollektion verwalten* in einem UML-konformen Use Case Diagramm.

b) Erstellen Sie ein UML-Klassendiagramm, welches die Beziehungen und sinnvolle Attribute der Klassen „Nutzer, Büro, Möbelstück und Wandschrank" darstellt.

c) Instanziieren Sie das Klassendiagramm in einem Objektdiagramm mit den zwei Nutzern mit Namen *Ernie* und *Bernd*, einem Büro mit der Nummer *CAPITOL2* und zwei Schränken mit den Inventurnummern *S1.88* und *S1.77*.

d) Geben Sie für ein Büromöbelstück ein Zustandsdiagramm an. Überlegen Sie dazu, welche möglichen Zustände ein Möbelstück während des Bestellvorgangs haben kann und finden Sie geeignete Zustandsübergänge.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1:

a) Erstellen Sie das konzeptuelle Modell eines (einfachen) Bibliotheksverwaltungssystems (z. B. für Ihre Schule) als Entity-Relationship-Schema. Es sollen insbesondere folgende Aspekte modelliert werden:

- Bücher können in mehreren Exemplaren vorhanden sein, die über die Signaturen eindeutig identifiziert werden.
- Schüler/innen können Bücher reservieren und ausleihen.
- Lehrer unterrichten Klassen in bestimmten Fächern und empfehlen dafür Bücher.
- Lehrer haben die Möglichkeit Bücher solange auszuleihen, bis eine Reservierung erfolgt. Daraufhin wird für sie und das entliehene Buch ein Mahneintrag erstellt.
- Die Bibliothek wird von einem Lehrer geleitet.

Erstellen Sie Ihr ER-Schema so, dass jedem Gegenstandstyp mindestens 3 Attribute zugeordnet sind. Spezifizieren Sie die Funktionalität / Kardinalität der Beziehungstypen.

b) Setzen Sie Ihr ER-Schema systematisch in ein relationales Schema ein. Wenn möglich sollten Relationen zusammengefasst werden, wenn dadurch keine Redundanzen / Anomalien auftreten. Markieren Sie die Schlüssel.

c) Für Ihr Bibliotheksverwaltungssystem formulieren Sie folgende SQL-Anfragen:

- ❖ Welche Lehrer haben wie viele Bücher ausgeliehen?
- ❖ Für welches Buch liegen die meisten Reservierungen vor?
- ❖ Welche/r Lehrer/in empfehlt für welche Klasse und welches Fach ein Buch, dessen Autor denselben Nachnamen hat wie der Lehrer.

Aufgabe 2:

Normalisieren Sie folgendes relationale Schema:

Allerlei: {[A, F, V, T, P, M]}

mit den funktionalen Abhängigkeiten

$A \rightarrow F$
$V \rightarrow T$
$P \rightarrow M$
$V \rightarrow P$
$A \rightarrow P$

Fortsetzung nächste Seite!

Aufgabe 3:

Zeigen Sie schrittweise (aber nicht jeden Einfügevorgang – nur vor und nach dem Überlauf) den Aufbau eines B*-Baums mit der maximalen Knotenkapazität von 4 Einträgen (die sowohl für innere Knoten als auch für die Blätter gilt). Die Einträge sind in folgender Reihenfolge einzufügen.

130, 120, 110, 100, 90, 80, 70, 60, 50, 40, 30, 20, 10

Wie sieht eine (statische) Hashtabelle für die obigen Einträge aus?