Fakultät IV
Fachgebiet NI

Lernerfolgskontrolle (C)
PPR
Nur zur Übung. Es gibt keinerlei Garantien, dass die tatsächliche Klausur den gleichen Umfang und Schwierigkeitsgrad hat.

Probeklausur

Musterlösung
Stand: 1. Februar 2014

Aufgabe	Punkte	Erreichte Punkte
1	8	
2	4	
3	5	
4	8	
5	4	
6	4	
7	9	
8	5	
9	5	
10	8	
Summe	60	

Aufgabe 1 (8 Punkte) Zahlensystem.

1. (3 Punkte) Berechen Sie 15-47 unter Verwendung der 8 Bit Zweierkomplementdarstellung. Die Rechenschritte müssen erkennbar sein.

   Lösung:

   $15_{10} = 8_{10} + 4_{10} + 2_{10} + 1_{10} = 00001111_2$

   $47_{10} = 32_{10} + 8_{10} + 4_{10} + 2_{10} + 1_{10} = 00101111_2$

   -47_{10} bilden: $47_{10} = 00101111_2$ invertieren, $\rightarrow 11010000_2$, um eins inkrementieren, $\rightarrow -47_{10} = 11010001_2$.

```
  0|00001111
+ 1|11010001
=============
  1|11100000
```

   1|1 $\rightarrow$ kein Über-/Unterlauf.

   $11100000_2 = -128_{10} + 64_{10} + 32_{10} = -32_{10}$

2. (2 Punkte) Die genetischen Informationen sind bei allen Lebewesen in den DNA-Molekülen gespeichert. Ein DNA-Molekül ist eine lineare Sequenz der vier Nukleobasen A,C,G und T. Das menschliche Genom enthält rund 2×10^9 Nukleobasen. Wie groß ist das gesamte Datenvolumen des menschlichen Genoms in Bytes?

   Lösung:

   $(\log_2 4 * 2 * 10^9)/8 = (2 * 2 * 10^9)/8 = 500 * 10^6$

3. (3 Punkte) Stellen Sie 30,25 in der 2 Byte binären Gleitkommadarstellung dar.
(1 Vorzeichenbit, 4 Bit Exponent: 7-Exzess-Darstellung, 11 Bit Mantisse).

Lösung:

Fixkomma: $30,25_{10} = 16_{10} + 8_{10} + 4_{10} + 2_{10} + 0,25_{10} = 11110,01_2$
Normieren: $11110,01_2 = 1,111001_2 * 2_{10}^{4_{10}} \rightarrow e = 4_{10}$, wegen 7-Exzess folgt $\rightarrow E = 11_{10} = 8_{10} + 2_{10} + 1_{10} = 1011_2$.
VZ-Bit: positiv $\rightarrow 0$, Mantisse: normierter Nachkommaanteil, aufgefüllt mit Nullen. $\rightarrow 11100100000_2$.

$\rightarrow 0\,1011\,11100100000$

.

Aufgabe 2 (4 Punkte) Logische Schaltungen.

1. (2 Punkte)

   Stellen Sie die Wahrheitstabelle für eine Funktion auf, die sowohl ausgeben kann, ob eine Zahl gerade ist, als auch, ob eine Zahl ungerade ist. Sie erhält als Eingabe eine Zahl von 0 bis 3 sowie eine Eingabe, die angibt, ob die Ausgabe anzeigt, ob die Zahl gerade oder ungerade ist. Diese Eingabe erfolgt natürlich binär.
   Beispiele (in Textform – eure Umsetzung dann in binär, und nur mit Zahlen von 0 bis 3!):
   f(123, gerade) = falsch
   f(514, gerade) = wahr
   f(865, ungerade) = wahr

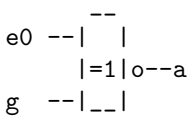
   Wahrheitstabelle:

   Lösung:

g	e_1	e_0	a
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**0**

2. (2 Punkte) Zeichnen Sie eine Gatterschaltung, die obige Wahrheitstabelle umsetzt. Sämtliche bekannten Gatter dürfen verwendet werden.

   Lösung:



Aufgabe 3 (5 Punkte) Rechnerarchitektur.

1. (3 Punkte)
   Was versteht man unter einem *Betriebsystem*? Nennen Sie *drei Aufgaben* eines Betriebsystems.

   Lösung:

   (a) Der Start und die Verwaltung von Prozessen (laufenden Programmen)
   (b) Speichermanagement
   (c) Erzeugung und Verwaltung eines logischen Dateisystems

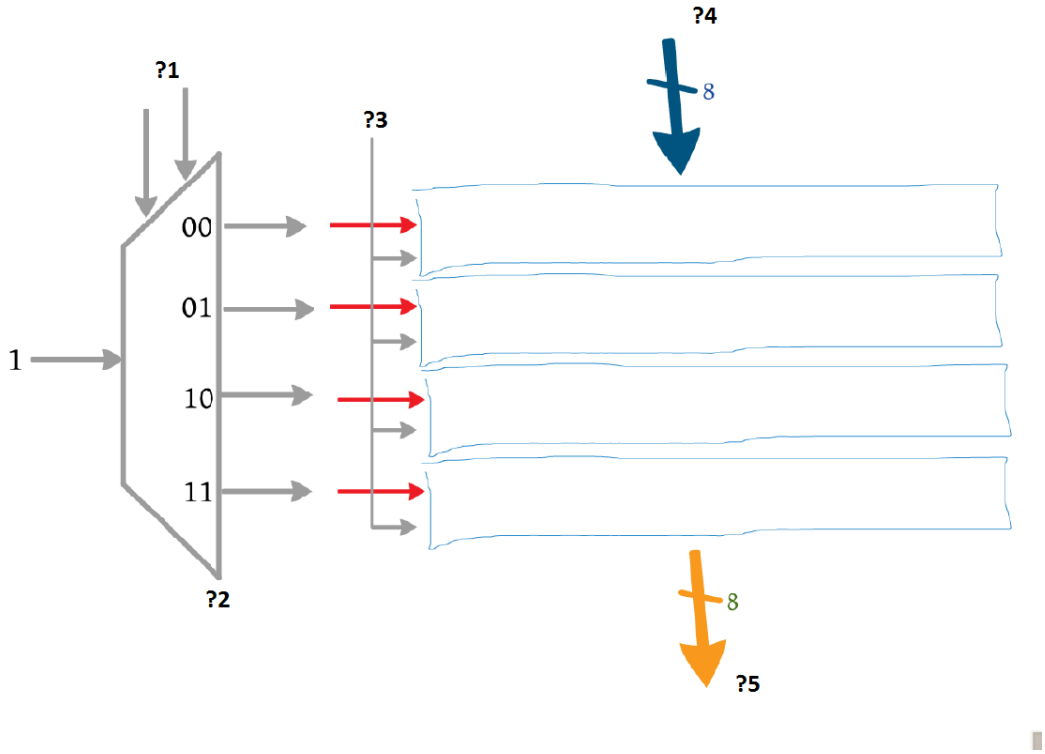
2. (2 Punkte)
   Erläutern Sie kurz die Begriffe *RAM* und *ROM*. Nennen Sie 2 Typen für RAM.

   Lösung:

   RAM ist die Abkürzung für random access memory. Der Inhalt kann jederzeit verändert werden. Es gibt statischen RAM, bestehend aus Flip-Flops, typischerweise auf Grund seiner hohen Geschwindigkeit für Cache verwendet. Dynamischer RAM, aufgebaut aus Kondensatoren, wird auf Grund seiner preiswerteren Herstellung typischerweise für den Hauptspeicher eingesetzt.
   ROM bezeichnet read-only memory, dessen Inhalt fest "eingebrannt" ist und deswegen nicht verändert werden kann.

Aufgabe 4 (8 Punkte) Rechneraufbau.

1. (4 Punkte)
   Gegeben ist folgendes (aus der Vorlesung bekannte) Bild:



   (a) Was stellt dies als Gesamtheit dar? (Was für eine Funktionseinheit)

   (b) Geben Sie `?4` und `?5` vernünftige Namen und erläutern Sie die Funktion dieser Leitungen. Was bedeutet der Querstrich mit der „8“?

   (c) Was ist die Funktion von Leitung `?3`?

   (d) Wie heißt das mit `?2` gekennzeichnete Bauteil? Wozu wird es *allgemein* (nicht nur in Bezug auf diese Schaltung) verwendet?

   (e) Erläutern Sie die Funktion der Leitungen `?1`.

   Lösung:

   (a) Dies stellt einen Tabellenspeicher dar.

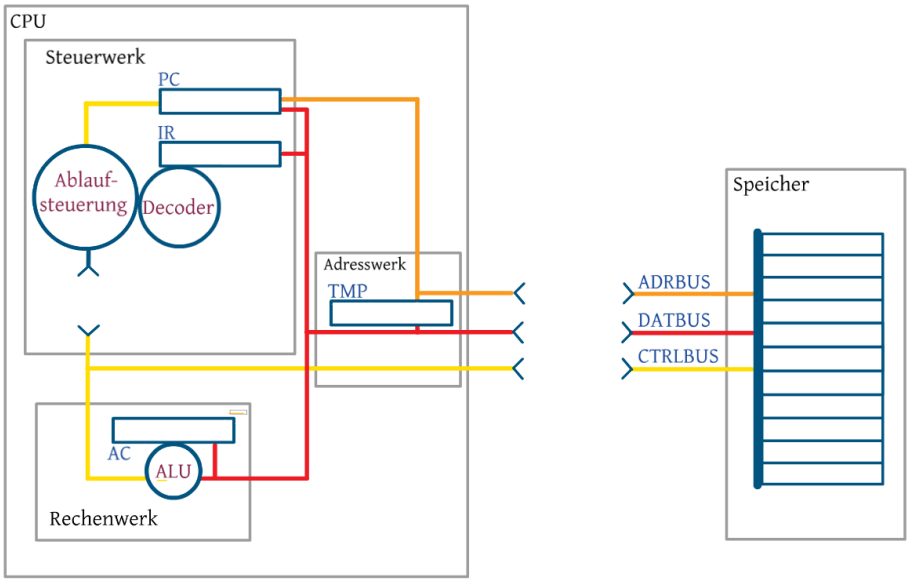
   (b) ?4: input, ?5: output. Darüber werden Daten in den Speicher geschrieben und ausgelesen. Die 8 markiert die Datenbreite dieser Leitungen (8 Parallelleitungen).

   (c) ?3 ist die Kontrollleitung, mit der gesteuert wird, ob gelesen oder geschrieben werden soll.

   (d) ?2 ist ein Demultiplexer. Es wird verwendet, um ein Eingangssignal auf verschiedene Ausgangsleitungen zu leiten, gesteuert über eine Adresse für jede Ausgangsleitung.

   (e) Dies ist der Adresseingang für den Speicher, mit dem die jeweils zu benutzende Speicherzelle angesprochen wird.

2. (4 Punkte)
Gegeben ist wieder das folgende aus dem Tutorium bekannte vereinfachte Rechnerblockschaltbild:



a) Erklären Sie kurz und knapp die Funktion der Bauteile PC, IR, ALU, AC, ADRBUS, CTRLBUS, DATBUS.

Lösung:

PC: Program Counter. Enthält die Speicheradresse, an der sich die Programmbearbeitung gerade befindet.
IR: Instruction Register. Enthält den aktuellen Befehl.
ALU: Arithmetical Logical Unit. Führt die arithmetischen und logischen Befehle aus.
AC: Accumulator Register. Enthält (Zwischen-)Ergebnisse für Berechnungen und Operationen.
ADRBUS: Adressbus. Über ihn läuft die Adresse der aktuell benutzten Speicherzelle des Hauptspeichers.
CTRLBUS: Kontrollbus. Über ihn laufen Kontrollanweisungen zwischen den Modulen, z.B., ob im Speicher geschrieben oder gelesen werden soll.
DATBUS: Datenbus. Darüber laufen die Daten zwischen Speicher und CPU.

b) Die folgende Tabelle stellt einen Speicherausschnitt vor Beginn der darunter angegeben Befehlsausführung dar. Der Index $_2$ ist benutzt, um anzuzeigen, dass diese Zahl im Binärsystem angegeben ist.

	RAM
5003	XOR
5004	5114
...	
5114	00001111_2

PC	IR	AC	TMP	CTRL	CBUS	ABUS	DBUS
5003	XOR	01100110_2	5113	PC++, PC→ABUS,RAM→DBUS,DBUS→IR	read	5003	XOR
5004	XOR	01100110_2	5114	PC++, PC→ABUS,RAM→DBUS,DBUS→TMP	read	5004	5114
5004	XOR	01101001_2	5114	TMP→ABUS,RAM→DBUS,DBUS→ALU	read	5114	00001111_2

Erläutern Sie für jede Ablaufzeile die Programmausführung.

Lösung:

(1) Der PC wird um eins erhöht (um den nächsten Befehl zu lesen); diese Adresse (5003) wird an den Speicher gelegt und der Inhalt der entsprechenden Zelle (XOR) über den Datenbus ins IR gelesen.
(2) Da XOR ein Befehl mit Operand ist, wird der PC auf die nächste Adresse gesetzt; von dieser wird dann aus dem Speicher die Operandenadresse ins TMP gelesen.
(3) Die Operandenadresse wird aus dem TMP an den Speicher gelegt, so dass der Operand über den Datenbus in die ALU gebracht werden kann, wo die Operation ausgeführt wird. Das Ergebnis landet im AC.

Aufgabe 5 (4 Punkte) Dateisystem.

1. (4 Punkte)

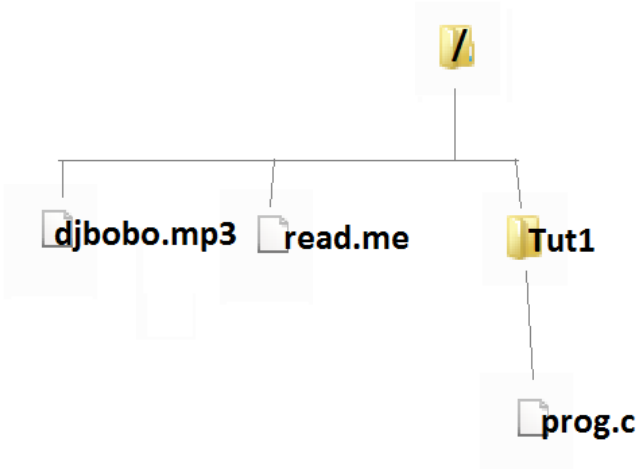
   Das Betriebssystem speichert alle Dateien in seinem internen Dateisystem, insbesondere in der Inode-Tabelle und im Datenblock.

   (a) Was wird im Datenblock abgelegt, was in der Inode-Tabelle?

   (b) Verzeichnisse sind auch Dateien – Verzeichnisdateien. Was liegt bei einer Verzeichnisdatei im Datenblock?

   (c) Wo findet das System den logischen Dateibaum, der dem Nutzer in Dateiexplorer-Tools angezeigt wird?

   (d) Wieviele Inode-Ids sind in folgendem Beispiel vergeben? (Nicht sichtbare Elemente werden nicht berücksichtigt)



Lösung:

(a) In der Inode-Tabelle werden zu jeder Datei die zugehörige id, der Dateityp, die physikalische Adresse der Daten im Datenblock, und Attribute wie Erstellungsdatum, Rechte etc gespeichert. Im Datenblock liegen die eigentlichen Dateiinhalte.

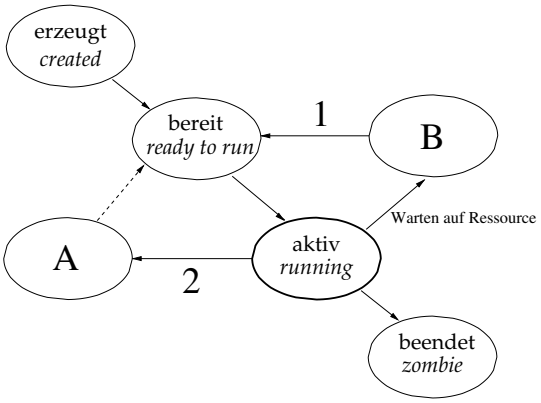
(b) Es liegt jeweils ein Eintrag bestehend aus inode-id und Dateiname für alle beinhaltenden Dateien und Ordner sowie für den Selbstverweis und den Verweis auf das übergeordnete Verzeichnis vor.

(c) Dieser ist nirgendwo explizit gespeichert. Das System kann ihn aus der Inode-Tabelle und den Verzeichnisdateien inferieren.

(d) 5.

Aufgabe 6 (4 Punkte) UNIX.

1. (2 Punkte)



Nennen Sie die in dem Diagramm fehlenden Prozesszustände A und B (englische oder deutsche Bezeichnung), sowie die Ursachen für die Übergänge 1 und 2:

Lösung:
A: preempted
B: asleep
1: Ressource verfügbar
2: Zeitscheibe abgelaufen

2. (2 Punkte) Beantworten Sie die folgenden Fragen durch Ankreuzen von „wahr“ oder „falsch“. Jede richtige Antwort gibt 0,5 Punkte, für jede falsche Antwort werden 0,5 Punkte abgezogen. Insgesamt können aber nicht weniger als 0 Punkte erreicht werden.

Lösung:

	wahr	*falsch*
Das Anhängen von & an einen Befehl lässt diesen im Hintergrund laufen.	(x)	( )
Systemprioritäten haben immer Vorrang vor Benutzerprioritäten.	(x)	( )
Mit dem Kommando `nice` kann jeder Benutzer die Prioritäten seiner eigenen Prozesse verbessern.	( )	(x)
Mehrere laufende Prozesse können sich eine Prozessnummer teilen	( )	(x)

Aufgabe 7 (9 Punkte) C.

1. (3 Punkte)

   Schreiben Sie eine Funktion `copyData`, die zwei void-Pointer übergeben bekommt (einer für die Quelle, einer für das Ziel) und die Anzahl der zu kopierenden Bytes. Es kann davon ausgegangen werden, dass der benötigte Speicherbereich am Ziel bereits reserviert ist.

   Lösung:

```
void copyData ( void* source, void* dest, int nrOfBytesToCopy )
{
  if ( !source || !dest ) return;
  char* p_source_byte = (char*)source;
  char* p_dest_byte = (char*)dest;
  for ( unsigned int i = 0; i < nrOfBytesToCopy; i++ )
  {
    p_dest_byte[i] = p_source_byte[i];
  }
}
```

2. (3 Punkte)

1.) Ergänzen Sie den Code unten (wo nötig mit beliebigen (aber halbwegs passenden) Werten). Hinweis: `fabs` ist eine Funktion der math-library, die den Betrag einer double-Zahl zurückgibt.

Lösung:

```
// Hier die Definitionen ergaenzen:
#define TRUE 1
#define FALSE 0

// Hier die fehlende define-Praeprozessor-Direktive ergaenzen:
#define TOLERANCE 0.000001

// Hier die fehlende Typdefinition ergaenzen:
typedef unsigned char bool;

bool areEqual ( double d1, double d2 )
{
  if ( fabs( d1 - d2 ) > TOLERANCE ) return FALSE;
  if ( fabs( d1 - d2 ) <= TOLERANCE ) return TRUE;
}
```

2.) Erklären Sie die Funktion und Motivation für die Funktion `areEqual`

Lösung:

Gleitkommazahlen können nich ohne weiteres mit dem Gleichheitsoperator == verglichen werden. Durch die begrenzte Genauigkeit und folgende Rundungsfehler in Berechnungen kann es zu Situationen kommen, in denen zwei Zahlen nicht genau gleich sind, obwohl sie nach mathematischer reeller Rechnung gleich sein sollten. Dies umgeht man, in dem man prüft, ob die beiden Zahlen hinreichend nah beieinander liegen – genau dies macht die Funktion `areEqual`.

3.) Vereinfachen Sie die Funktion so weit wie möglich (eine Zeile reicht...).

Lösung:

```
// Hier die Funktion vereinfachen (die Praeprozessor- und Typdefinitionen von oben sind noch gueltig !):

bool areEqual ( double d1, double d2 )
{
  return ( fabs( d1 - d2 ) <= TOLERANCE );
}
```

3. (3 Punkte) Schreiben Sie einen Prototypen (nur der Funktionskopf!) für eine Funktion, die...
a) einen String „Hallo Welt“ erzeugt und diesen zurückgibt
b) von einem float-Array der Länge 100 eine Kopie anlegt und die Adresse dieser Kopie zurückgibt
c) einen String „Hallo Welt“ erzeugt und diesen ausgibt

Lösung:

```
char* getHalloWelt();

float* copyArray( float array[100] );

void printHalloWelt ();
```

Aufgabe 8 (5 Punkte) C.

1. (1 Punkt)

   Programmieren Sie eine Ausgabe an den Nutzer, dass er angeben möge, wieviele float-Zahlen in einem später anzulegenden Array gespeichert werden sollen, und lesen Sie daraufhin die folgende Nutzereingabe in die Variable memoryRequest ein.

   Lösung:

```
int memoryRequest;

printf( "Bitte geben Sie an, wieviele float-Zahlen Sie im Array speichern wollen: " );
scanf( "%i", &memoryRequest );
```

2. (2 Punkte)

   Reservieren Sie Speicher für den float-Array floatArray mit der Größe memoryRequest (siehe oben). Dabei sollen schon bei der Speicherreservierung alle Zellen auf 0 initialisiert werden. Überprüfen Sie anschließend, ob die Reservierung erfolgreich war, wenn nicht, geben Sie eine Fehlermeldung aus und beenden das Programm mit dem Fehlercode -1.

   Lösung:

```
float* floatArray;

floatArray = (float*)calloc( memoryRequest, sizeof(float) );
if ( !floatArray )
{
  printf( "Speicher konnte nicht alloziiert werden.\n" );
  exit(-1);
}
```

3. (2 Punkte)

   Analysieren Sie folgenden Code:

```
float* fa = floatArray;
void* fav = (void*)floatArray;
double* fad = (double*)floatArray;
printf("%p", fa); // (Ausgabe)
fa = fa + 4; // (1)
fav = fav + 4; // (2)
fad = fad + 4; // (3)
```

   Angenommen, an der Stelle (Ausgabe) würde „160“ ausgegeben werden – welchen Wert hat fa nach Ausführung der mit (1) markierten Zeile, welchen Wert hat fav nach Ausführung der mit (2) markierten Zeile, und welchen Wert hat fad nach Ausführung der mit (3) markierten Zeile?

   Lösung:
   (1) fa == 176 (2) fav == 164 (3) fad == 192

Aufgabe 9 (5 Punkte) C.

1. (5 Punkte)

   Betrachten Sie folgendes Programm, und ergänzen Sie die darunter stehende Ablauftabelle.

   - Tragen Sie die Variablenbelegungen nach Ablauf der jeweiligen Zeile ein.
   - Schreiben Sie bei Funktionsaufrufen zusätzlich die aufrufende Zeile in Klammern hinter die momentane Zeilennummer. Bsp: Wir befinden uns in Zeile 14 in einer Funktion, die in Zeile 34 aufgerufen wurde. `Zeile: 14 (34)`
   - Die erste aufgeführte Zeile einer Funktion ist die öffnende Blockklammer ({).
   - Nach Aufruf eines `return`-Statements wäre die nächste ausgeführte Zeile der Funktionsabschluss (}).
   - Kennzeichnen Sie aktuell nicht im Speicher angelegte Variablen mit `-`, existierende Variablen mit undefinierten Werten mit `undef`.
   - Die Zahl der Zeilen in der Tabelle ist auch abgezählt, d.h., so viele Zeilen werden im Code auch durchlaufen und von euch bitte beschrieben.

```
1  double lcl = −3.0;
2
3  void sub(double z)
4  {
5     lcl  −= z;
6  }
7
8  int addCond(char x, double y)
9  {
10    if ( !x ) return (y + 0.49);
11    else if ( x >= 2 ) return (y + 4.9);
12    else return (y − 0.49);
13 }
14
15 int main()
16 {
17    int c;
18    c = addCond(2, lcl);
19    sub( lcl );
20    lcl = lcl − (c + 1);
21 }
```

Lösung:

Zeile	lcl	z	x	y	c
16	-3.0	-	-	-	-
17					undef
9(18)			2	-3.0	
10(18)					
11(18)					
13(18)			-	-	
18					1
4(19)		-3.0			
5(19)	0				
6(19)		-			
19					
20	-2.0				
21	-	-	-	-	-

Aufgabe 10 (8 Punkte) C.

1. (2 Punkte)

   Definieren Sie eine Struktur mit dem Namen **pprStudent**, welche folgende Informationen erfasst: Den Namen (String, max. Länge 100), die Hausaufgabennote (als Gleitkommazahl), und die Klausurnote (auch Gleitkommazahl). Definieren Sie dabei auch einen neuen Typ **PprStudent** für diese Struktur.

   Lösung:

```
typedef struct pprStudent {
  char name[101];
  float uenote;
  float klnote;
} PprStudent;
```

2. (1 Punkt)

   Definieren Sie eine neue Struktur, die eine einfach verkettete Liste von **PprStudent**-Strukturen darstellen kann. Definieren Sie wieder auch einen neuen Typ **PprlistElem** für die Struktur.

   Lösung:

```
typedef struct pprlistElem {
  PprStudent student;
  struct pprlistElem * nextListElem;
} PprlistElem;
```

3. (2 Punkte)

   Was ist der Vorteil einer verketteten Liste gegenüber der Speicherung in einem Array? Gibt es auch Nachteile? Fällt Ihnen eine weitere Datenstruktur ein, die besonders für die Suche in den Daten geeignet ist?

   Lösung:

   Ein einer verketteten Liste ist das Einfügen und Löschen von Elementen (außer ganz am Ende) deutlich zeiteffizienter als in einem Array, deswegen ist sie grundsätzlich im Vorteil, wenn es um sortierte Listen geht. Ein Nachteil ist der höhere Speicheraufwand im Vergleich zum Array. Für die Suche in sortierten Listen ist insbesondere die Baumstruktur sehr gut geeignet.

4. (3 Punkte)

   Schreiben Sie folgende Funktion, die die Durchschnittsnote aller Studenten berechnen soll (Note pro Student: 70% Klausur, 30% Hausaufgaben). Das übergebene Argument pprlist soll dabei das erste Element der Liste sein. Überprüfen Sie, ob überhaupt ein Element in der Liste vorhanden ist!

   Lösung:

```
float averageMark( PprlistElem* pprlist )
{
  if ( ! pprlist ) return 0.0;
  float markSum = 0.0;
  unsigned int numMarks = 0;
  for ( PprlistElem* p = pprlist ; p != 0; p = p->nextListElem )
  {
    float thisMark = 0.3 * p->student.uenote + 0.7 * p->student.klnote;
    markSum += thisMark;
    numMarks++;
  }
  return (markSum / (float)numMarks);
}
```