Name, Vorname, Unterschrift:	Matrikelnummer:

Klausur
Kompaktkurs Einführung in die Programmierung
(Dr. S. Zimmer)
4. Juli 2008

Wichtig:

- Die Angabe besteht aus 6 Seiten — prüfen Sie bitte Ihr Exemplar auf Vollständigkeit.
- Kennzeichen Sie jedes Blatt lesbar mit Ihrem Namen und dieses Deckblatt zusätzlich mit Ihrer **Matrikelnummer und Unterschrift**; trennen Sie die Heftung nicht auf.
- Bearbeiten Sie die Aufgaben auf den Angabenblättern. Sollte der Platz für einen Aufgabenteil nicht reichen, benutzen Sie bitte die Rückseite der Angabenblätter und machen Sie einen entsprechenden Vermerk bei der Aufgabe.
- Insgesamt können Sie 40 Punkte erreichen, bei jeder Aufgabe sind die für die jeweiligen Teilaufgaben vorgesehenen Punkte angegeben. Zum Bestehen sind 17 Punkte erforderlich.
- Als Hilfsmittel ist nur ein Ausdruck der Vorlesungsfolien erlaubt (handschriftliche Anmerkungen sind OK).
- Die Bearbeitungszeit beträgt 75 Minuten.

Aufgabe	1a)	1b)	1c)	1d)	2a)	2b)	3a)	3b)	3c)	3d)	Summe	Note
Punkte	/3	/4	/3	/4	/5	/7	/2	/3	/3	/6	/40	

1 Drei ganze Zahlen (3+4+3+4=14 Punkte)

a) Beschreiben Sie ein Verfahren, das drei Zahlen a, b und c der Größe nach (aufsteigend, also die kleinste zuerst) sortiert. Die Grundoperationen sollen dabei "zwei Zahlen vergleichen" und "zwei Zahlen vertauschen" sein. Welche Vergleiche führen Sie durch? Was für Vertauschungen finden bei welchem Ergebnis jeweils statt? (C-Code ist hier noch nicht verlangt.)

b) Implementieren Sie das Verfahren aus a): Schreiben Sie eine C-Funktion ohne Funktionsergebnis und mit drei ganzen Zahlen a, b und c als Parameter. Sie soll die drei Zahlen sortieren (also ggf. vertauschen, so dass $a \leq b \leq c$ gilt) und dann a, b und c in dieser Reihenfolge ausdrucken (`std::cout` oder `printf`).

b) Beschreiben Sie das Ergebnis der folgenden Funktion (z.B. "Berechnet das kleinste $x \in \mathbb{Z}$ mit $x > (a+b+c)/3$")! (Vielleicht ist $a = 1$, $b = 3$, $c = 2$ ein geeignetes Beispiel zum Probieren; denken Sie aber auch an den Fall, dass zwei oder auch alle drei Zahlen gleich sein können)

```
int blub(int a, int b, int c) {
  if ((a-b)*(a-c)<=0)
    return a;
  else
    return blub(b, c, a);
}
```

c) Schreiben Sie eine Funktion, die die Inhalte von drei `int`-Variablen a, b und c zyklisch vertauscht: a bekommt den Inhalt von b, b den von c und c den von a. Die Funktion soll also drei Variablenparameter haben und kein Funktionsergebnis (nur die Vertauschung als Seiteneffekt).

2 Felder (5+7=12 Punkte)

Zum Speichern von n nichtnegativen ganzen Zahlen wird in dieser Aufgabe ein Feld von $n + 1$ `int` verwendet, wobei die letzte Komponente auf -1 gesetzt wird, um das Ende des Feldes zu markieren. Wenn man die Werte 10, 20, 30, 40 und 50 speichern möchte, hätte man also das Feld `{10,20,30,40,50,-1}`.

a) Schreiben Sie eine Funktion mit einem solchen Feld als Parameter, die jede zweite Komponente ausdruckt (`std::cout` oder `printf` – im Beispiel von oben also die 20 und die 40).

b) Schreiben Sie eine Funktion mit einem solchen Feld als Parameter, die genau die Komponenten ausdruckt, die größer sind als alle vorhergehenden – wenn im Feld die Komponenten 1,2,3,2,3 und 4 (in dieser Reihenfolge) gespeichert wären, also die 1, die erste 2, die erste 3 und die 4.

3 Dezimalzahlen als Liste (2+3+3+6=14 Punkte)

In dieser Aufgabe sollen die Ziffern einer Dezimalzahl als einfach verkettete Liste gespeichert werden; jedes Listenelement enthält dabei

- eine Ziffer, dargestellt durch einen `int`-Wert (Sie dürfen dabei davon ausgehen, dass bei allen Listen, die Sie als Parameter Ihrer Funktionen bekommen, alle Ziffern zwischen 0 und 9 sind)
- und wie üblich einen Zeiger auf das nächste Element (bzw. den Nullzeiger), das die Ziffer mit der nächsthöheren Wertigkeit enthalten soll: der Listenanfang enthält also die Einerstelle, das zweite Element die 10er-Stelle etc.

Eine Liste wird wie üblich repräsentiert durch einen Zeiger auf das erste Element (Sie dürfen annehmen, dass alle vorkommenden Listen nicht leer sind). Führende Nullen sollten Sie bei den Aufgaben hier nicht stören (falls Sie sich doch gestört fühlen, dürfen Sie auch annehmen, dass es keine gibt).

a) Geben Sie einen passenden Strukturtyp für die Listenelemente an!

b) Schreiben Sie eine Funktion mit einer solchen Liste als Parameter, die die zugehörige Dezimalzahl ausdruckt. (Dabei dürfen Sie nicht voraussetzen, dass die Zahl als `int` darstellbar ist, Verwendung der Funktion aus c) scheidet also aus.)

c) Schreiben Sie eine Funktion mit einer Liste als Parameter und Ergebnistyp `int`, die den Wert der zugehörigen Dezimalzahl als Ergebnis hat. (Hier dürfen Sie voraussetzen, dass die Zahl als `int` darstellbar ist.)

d) Schreiben Sie eine Funktion mit zwei Listen als Parameter, die eine neue Liste anlegt und zurückliefert, die die Summe der beiden Parameter enthält. Wie in Teil b) dürfen Sie wieder nicht annehmen, dass die Zahlen als `int` darstellbar sind, Sie müssen also Ziffer für Ziffer addieren und sich um Überträge kümmern — z.B. mittels einer Hilfsfunktion, die als zusätzlichen Parameter einen in der vorigen Stelle angefallenen Überlauf enthält.