Dr. G. Zachmann, Dr. M. Frank
M. Guthe, P. Degener, C. Scholz

Universität Bonn
Institut für Informatik II/IV

Wierderholung der Modulprüfung zu "Grundlagen der Programmierung"

Wintersemester 2003/2004 und Sommersemester 2004

24. 9. 2004 , 10:00 – 12:00 Uhr

Name: ..... **Vorname**: .....

Matrikelnummer: ..... E-Mail: .....@cs.uni-bonn.de

Hauptfach: ..... Semesterzahl: .....

Bitte in Druckschrift ausfüllen.

Hinweise:

- Neben Papier und Schreibutensilien sind keine weiteren Hilfsmittel erlaubt. Verwenden Sie nur dokumentenechte Stifte. Verwenden Sie keine roten Stifte.
- Vergessen Sie nicht, Ihren Namen und die Matrikelnummer auf *jedes* Blatt zu schreiben. Blätter ohne diese Angaben werden nicht gewertet.
- Schreiben Sie Ihre Lösungen auf die Aufgabenblätter möglichst in die dafür vorgesehenen Felder. Sie können auch die Rückseiten verwenden. Weiteres Schreibpapier kann von den Betreuern angefordert werden. Benutzen Sie kein mitgebrachtes Papier.
- Bei Multiple-Choice-Fragen wird für jedes richtig gesetzte Kreuz die im Kopf der Aufgabe angegebene Teil-Punktzahl vergeben. Für jedes falsch gesetzte Kreuz wird eine entsprechende Anzahl von Punkten abgezogen. Nicht beantwortete Fragen werden nicht gewertet. Die Gesamtpunktzahl beträgt mindestens 0.
- Bitte schreiben Sie in Ihrem eigenen Interesse deutlich. Für nicht lesbare Lösungen können wir keine Punkte vergeben.
- Klausurblätter dürfen nicht voneinander getrennt werden.
- Werden mehrere unterschiedliche Lösungen für eine Aufgabe abgegeben, so wird die Aufgabe nicht gewertet.
- Im Fall von Täuschungsversuchen wird die Klausur sofort mit 0 Punkten bewertet. Eine Vorwarnung erfolgt nicht.
- Notenschlüssel (in Punkten, jew. einschl.): $45.0 - 49.0 \equiv 4.0$, $49.5 - 53.5 \equiv 3.7$, $54.0 - 58.0 \equiv 3.3$, $58.5 - 62.5 \equiv 3.0$, $63.0 - 67.0 \equiv 2.7$, $67.5 - 71.5 \equiv 2.3$, $72.0 - 76.0 \equiv 2.0$, $76.5 - 80.5 \equiv 1.7$, $81.0 - 85.0 \equiv 1.3$, $85.5 - 104 \equiv 1.0$.

Aufgabe	**1**	**2**	**3**	**4**	**5**			
max. Punkte	19	8	8	8	9			
erreicht								

Aufgabe	**6**	**7**	**8**	**9**	**10**	**11**	**12**	$\sum$
max. Punkte	11	4	9	8	7	7	6	**104**
erreicht								

Note: **Unterschrift:**

Variante A

Aufgabe		**19** Punkte
	Name:	(1 + 18 × 1)
1	Matrikel-Nr.:	Σ: .....

Multiple-Choice zu Programmierung II

Vorsicht: Falsche Antworten führen zu Punktabzug. Die Mindestpunktzahl der gesamten Aufgabe beträgt jedoch 0 (siehe Bemerkung auf Seite 1).

a) Welche der abgebildeten Personen sind die Dozenten der Vorlesung "Grundlagen der Programmierung" im Wintersemester 2003/2004 bzw. Sommersemester 2004? Machen Sie genau zwei Kreuze.


☐

☐

☐

☐

☐

☐

☐

☐

b) Entscheiden Sie bei jeder der folgenden Aussagen, ob sie richtig oder falsch ist.

Aussage	Aussage richtig	
	Ja	Nein
Das Transportprotokoll UDP (User Datagram Protocol) bietet unsichtbar für den Programmierer eine zuverlässige Verbindung mit Fehlerkorrektur.	☐	☐
Eine Variable vom Typ Double oder Long muss beim Versand über die Systemfunktionen bei Programmierung in C/C++ von TCP oder UDP entsprechend der Prozessorarchitektur (Little-Endian- bzw. Big-Endian-Byte-Order) zeichenweise in die Network-Byte-Order verwandelt werden.	☐	☐
Ein TCP-Client bereitet sich durch die Abfolge von socket() und connect() für die Kommunikation mit einem TCP-Server vor.	☐	☐
Ein TCP-Server bereitet sich durch die Abfolge von socket(), bind(), connect() und accept() auf eingehende Verbindungsaufbauwünsche vor.	☐	☐
Die Systemfunktion fork() liefert eine identische Kopie des aufrufenden Prozesses. Der Programmierer muss im Programmfluss nach fork() das ggf. unterschiedliche Verhalten der identischen Kopien entsprechend im Programmtext realisieren.	☐	☐
Bei der Programmierung von Java-Threads muss die Methode run() implementiert werden und im Konstruktor der Thread-Klasse wird der Thread durch Aufruf von this.start() gestartet.	☐	☐
Java RMI (Remote Method Invocation) nutzt zwischen Client und Server das Communication Module, in dem die Kommunikation völlig unsichtbar für den Programmierer abgewickelt wird.	☐	☐
Bei der Programmierung für Java RMI muss zur Programmlaufzeit am Client die Klassen-Datei des Server-Objektes vorliegen.	☐	☐
Bei Vererbung von Java-Klassen und Überladen von Methoden/Operatoren erzeugt der Java-Compiler jeweils eigene Klassendateien.	☐	☐
Bei der Programmierung von Python geschieht die Blockbildung durch Einrückung der Zeilen des Programmtextes.	☐	☐
Eine Zuweisung in Python erzeugt standardmäßig eine lokale Variable und es ist ein Laufzeitfehler, wenn eine Variable gelesen wird, der zuvor noch kein Wert zugewiesen wurde.	☐	☐
Das Prozessor-Status-Register kann mit Hilfe von bestimmten Sprungbefehlen zur Verzweigung im Assemblerprogramm ausgewertet werden.	☐	☐

Aufgabe		
1	Name: ..... Matrikel-Nr.:	

Aussage	Aussage richtig	
	Ja	Nein
Gemäß der cdecl-Calling-Convention für den Aufruf von Subroutinen werden lokale Variablen der Subroutine in reservierten Speicherstellen vor dem Assemblercode der Subroutine abgelegt.	☐	☐
Die Systemfunktion select() kann gleichzeitig TCP Listening-Sockets und TCP Connection-Sockets für Input (Lesen) und Output (Schreiben) überwachen.	☐	☐
Ein iterativer TCP-Server kann nach dem Eingang einer Verbindung vom Client nur eine begrenzte Datenmenge vom Client verarbeiten, schließt dann die Verbindung und ist für weitere eingehende Verbindungen empfangsbereit.	☐	☐
Das Java Native Interface (JNI) kann verwendet werden, um C-Funktionen aus Java-Programmen aufzurufen oder auch um Java-Objekte aus C-Programmen zu instanzieren und auf Java-Methoden zuzugreifen.	☐	☐
Ein Python-Dictionary ordnet jedem Schlüssel einen eindeutigen Wert zu. Dadurch wird auch die Iteration über die Werte eines Dictionary möglich.	☐	☐
Mit der i386-Assembler-Instruktion <movl %eax, 10(%esi, %ecx, 4)> wird der Wert des %eax-Registers, an die Speicherstelle geschrieben, die sich aus dem Inhalt von %esi + (4 * Inhalt von %ecx) + Displacement von 10 Bytes ergibt.	☐	☐

Aufgabe		**8** Punkte
	Name:	(8 × 1)
2	Matrikel-Nr.:	Σ: .....

I/O Multiplexing in C

Ein C-Programm erwartet auf zwei unterschiedlichen UDP-Ports gleichzeitig Datenpakete, deren Inhalt auf der Standardausgabe ausgegeben werden soll. Da man nicht weiß, wann Daten auf welchem Port ankommen, muss ein Multiplexing zwischen den beiden zugehörigen Sockets `sockfd1` und `sockfd2` mittels `select` durchgeführt werden. Das Hauptprogramm erzeugt und bindet dazu zunächst diese beiden Sockets und ruft anschließend die unten angegebene Funktion `processRequests(...)` auf.

Ergänzen Sie den folgenden Programmauszug an den angegebenen Stellen `(1)` - `(8)` enstprechend:

```
void processRequests(int sockfd1, int sockfd2)
{
  int maxsock;
  fd_set set;

  if (_____________________________________) {                (1)
    maxsock = sockfd2;
  }
  else {

    ____________________________________________              (2)

  }

  while(1) {
    # fd_set set entsprechend setzen

    ____________________________________________              (3)

    ____________________________________________              (4)

    ____________________________________________              (5)


    select(_____________________________, NULL, NULL, NULL);  (6)

    checkFd(sockfd1, set);
    checkFd(sockfd2, set);
  }
}

void checkFd(int sockfd, fd_set set)
{
  int count;
  char buffer[MAXLINE];

  if (_____________________________________________) {        (7)

    count = _____________________________________________     (8)

    if (count > 0) {
      write(STDOUT_FILENO, buffer, count);
    }
  }
}
```

Aufgabe		**8** Punkte
	Name:	
3	Matrikel-Nr.:	Σ: .....

Rekursion in Python

Die Türme von Hanoi sind ein mathematisches Knobel- und Geduldspiel.

Es besteht aus drei Feldern, auf die Scheiben verschiedener Größe gelegt werden können. Zu Beginn sind alle Scheiben auf einem Feld, der Größe nach geordnet, mit der größten Scheibe unten und der kleinsten oben. Bei jedem Zug darf die oberste Scheibe eines beliebigen Feldes auf eines der beiden anderen Felder gelegt werden, vorausgesetzt, dort liegt nicht schon eine kleinere Scheibe. Ziel des Spiel ist es, den kompletten Scheiben-Stapel auf ein anderes Feld zu versetzen.

Schreiben Sie eine rekursive Python-Funktion `hanoi(n, von, nach, temp)`, die die einzelnen Schritte zur Lösung des Problems ausgibt. `n` gibt dabei die Anzahl der zu bewegenden Scheiben an. Die Scheiben sind vom Feld `von` zum Feld `nach` unter Benutzung des Hilfsfeldes `temp` zu verschieben.

Die Ausgabe soll dabei folgendes Format haben (Ausgabe von `hanoi(3,'A','B','C')`):

```
Scheibe 1: A ==> B
Scheibe 2: A ==> C
Scheibe 1: B ==> C
Scheibe 3: A ==> B
Scheibe 1: C ==> A
Scheibe 2: C ==> B
Scheibe 1: A ==> B
```

```
def hanoi(n, von, nach, temp):
```

Aufgabe		8 Punkte
4	Name: ..... Matrikel-Nr.:	Σ: .....

Kontrollstrukuren in Python

Betrachten Sie den u.a. Rahmen eines Python-Programmes. Dort wird in der Variablen `Liste` zunächst eine Liste von 20 Zufallszahlen erzeugt.

Ergänzen Sie nun maximal 6 Zeilen Python-Source (markierte Zeilen (1) bis (6)), so dass nach Durchlaufen der Schleife `for elem in Liste` die Variablen `erster`, `zweiter`, `dritter` jeweils das größte, zweitgrößte und drittgrößte Element der Liste beinhalten.

Ein derartiges Programm könnte z.B. benutzt werden, um aus einer beliebigen Anzahl von Teilnehmern bei einem Wettbewerb den 1., 2. und 3. Platz gemäß Punktestand zu ermitteln.

Beachten Sie ferner folgende Einschränkungen bzw. Annahmen:

- Schreiben Sie jeweils nur ein Python-Statement in eine Zeile, d.h. benutzen Sie keine Semikola um mehrere Statements in eine Zeile zu setzen und fangen Sie nach einem Doppelpunkt (z.B. im if-Statement) eine neue Zeile an.
- Sie dürfen keine Python-Module sowie Klassen und Methoden einbinden, insbesondere nicht zum Sortieren (`Liste.sort()` ist verboten!).
- Das entstehende Programm soll lineare Laufzeit haben (abhängig von der Länge der Liste).
- Die Liste darf nicht verändert werden
- Die Liste enthält kein Element mehrfach.

Hinweise: Denken Sie an Python-Tupel mit 2 oder mehr Elementen, z.B. zum Variablentausch. Es gibt mehrere (verschiedene!) korrekte Lösungen.

```
import random

Liste = []
while len(Liste) < 20:
    rand = random.randint(0,99)
    if not rand in Liste:
        Liste.append(rand)

print Liste

erster, zweiter, dritter = 0, 0, 0

for elem in Liste:
```

```
(1)
(2)
(3)
(4)
(5)
(6)
```

```
print "1. Platz:", erster
print "2. Platz:", zweiter
print "3. Platz:", dritter
```

Aufgabe	Name:	
4	Matrikel-Nr.:	

Das Programm soll mit der angegebenen Beispielliste dann folgende Ausgabe ergeben:

```
[29, 22, 50, 31, 64, 67, 92, 36, 32, 35, 69, 71, 48, 12, 51, 9, 76, 60, 55, 26]
1. Platz: 92
2. Platz: 76
3. Platz: 71
```

Aufgabe		**9** Punkte
	Name:	(9 × 1)
5	Matrikel-Nr.:	Σ: .....

Subroutinen in i386-Assembler

Die folgende C-Funktion `add` wurde unter Beachtung der in der Vorlesung vorgestellten cdecl Calling-Convention "eins zu eins" in Assembler übersetzt. Der Assemblercode auf der nächsten Seite enthält zusätzlich noch den Aufruf der Funktion `add(9, 7, &erg)`. In diesem Code haben sich leider einige semantische Fehler eingeschlichen.

Streichen Sie (auf der nächsten Seite!) alle fehlerhaften Zeilen durch und schreiben Sie den korrekten Befehl rechts daneben. Beachten Sie auch die Kommentare im Assemblercode. Die Kommentarzeilen enthalten keine Fehler, d.h. der Assemblercode ist richtig, wenn er den Kommentaren entspricht.

Für jede gefundene und richtig korrigierte fehlerhafte Zeile gibt es einen Punkt. Im folgenden Assembler-Programm sind genau neun fehlerhafte Zeilen enthalten.

```
void add (int a, int b, int *c)
{
  int x;
  int y;
  int z;

  x = a;
  y = b;

  z = a + b;

  *c = z;
}
```

Aufgabe		
5	Name: ..... Matrikel-Nr.:	

```
.data
erg:    .word   0
.text
.global _start
_start:
        # place parameters on stack
        pushl   $9
        pushl   $7
        pushl   $erg

        # call add(9, 7, &erg)
        call    add
        # adjust stack
        addl    $3,%esp

        # exit process
        movl    $1,%eax
        movl    $0,%ebx
        int     $0x80

# void add(int a, int b, int *c)
add:
        # set up stack frame
        pushl   %esp
        movl    %esp,%ebp

        # reserve space for three ints
        subl    $12,%esp

        # copy a to local variable x
        movl    8(%ebp),%eax
        movl    %ecx,-4(%ebp)

        # copy b to local variable y
        movl    16(%ebp),%eax
        movl    %eax,8(%ebp)

        # calculate x+y in %eax
        movl    -4(%ebp),%eax
        addl    -8(%ebp),%eax

        # copy x+y to local variable z
        movl    %eax,-12(%ebp)

        # copy c to %eax
        movl    16(%ebp),%eax

        # copy x+y to %ecx
        movl    %eax,%ecx

        # copy x+y to *c
        movl    %ecx,%eax

        leave
        ret
```

Aufgabe		**11** Punkte
	Name:	(11 × 1)
6	Matrikel-Nr.:	Σ: .....

Syntax von C++

Entscheiden Sie für jedes der gegebenen Code-Fragmente, ob es, eingesetzt in das folgende Gerüst an der markierten Stelle X, zu einem Compiler-Fehler führt oder nicht.

```
int main() {
  // Stelle X
}
```

Sie können einen beliebigen ISO-C++-standardkonformen Compiler (beispielsweise `gcc` oder `icc`, aber nicht `msvc`) voraussetzen. Vorsicht: Falsche Antworten führen zu Punktabzug. Die Mindestpunktzahl beträgt jedoch 0 (siehe Bemerkung auf Seite 1).

Code-Fragment	compiliert	
	Ja	Nein
`int a,z;` `char a-z = static_cast<char>( a-z );`	☐	☐

`int count;`
`int counter2;`
`counter2 += count;`
	☐	☐

`int i = 0;`
`bool ok?;`
`if ( ok? )`
`  i ++ ;`
	☐	☐

`int j = 0;` `if ( j > 5 );`	☐	☐
`int k = 3, l = 6;` `int m = ( k == l );`	☐	☐
`double d = 5;` `float f = d;`	☐	☐
`char r = 'r', s = 's';` `char t = (r , s);`	☐	☐
`int *******p;`	☐	☐

`while ( true ) {`
`  if ( true )`
`  break;`
`}`
	☐	☐

`struct Mouse`
`{`
`  float size;`
`};`
`Mouse mouse;`
`mouse->size = 3;`
	☐	☐

`int number = 1%1;`	☐	☐

Aufgabe		**4** Punkte
7	Name:	
	Matrikel-Nr.:	Σ: .....

Fallunterscheidungen in C++

Schreiben Sie die im folgenden C++-Code-Fragment gegebene `switch`-Fallunterscheidung in mehrere `if`-Anweisungen um.

```
unsigned int r = rand();
switch( r )
{
    case 0:
    case 1:
    case 2:
    case 3: printf("r kleiner 4");
            break;
    case 4:
    case 5: printf("r ist 4 oder 5");
            break;
    default: printf("r ist zu gross");
}
```

Aufgabe		**9** Punkte
	Name:	(2 + 6 + 1)
8	Matrikel-Nr.:	Σ: .....

Klassen in C++

Gegeben ist folgender Struct zur Speicherung von 2-dimensionalen Punkten:

```
struct Point2D
{
    float x;
    float y;
};
```

Außerdem dazu eine globale Funktion zur Berechnung der Distanz zwischen `p1` und `p2`:

```
float len( const Point2D & p1, const Point2D & p2 )
```

a) Deklarieren Sie eine abstrakte Klasse `Shape2D` in C++, die als Oberklasse für 2-dimensionale Formen dient. Deklarieren Sie außerdem die Methode `perimeter()`, die den Umfang des Objektes liefert.
Hinweis: Sie dürfen in allen Teilaufgaben die Implementierung (= Definition) der Methoden innerhalb der Deklaration einer Klasse schreiben, um Schreibarbeit zu sparen.

Aufgabe		
8	Name: ..... Matrikel-Nr.:	

b) Leiten Sie von `Shape2D` eine Klasse `Polygon` ab und implementieren Sie folgende Methoden:

1. Einen Konstruktor mit den Argumenten `(const Point2D * points, int length)`;
2. einen Destruktor;
3. die Methode `float perimeter() const` zur Berechnung des Umfangs;

Hinweis: Benutzen Sie `new` und `delete` zur Speicherverwaltung. Memory-Leaks führen zu Punktabzug.

c) Deklarieren Sie die Methode `perimeter` in `Shape2D` so, daß sie in der (Unter-)Klasse nicht implementiert werden muß (was natürlich zur Folge hat, daß bei Nicht-Implementierung keine Instanzen der Klasse erzeugt werden können).

Aufgabe		**8** Punkte
9	Name:	(2+6)
	Matrikel-Nr.:	Σ: .....

Templates in C++

Betrachten Sie das in den folgenden drei Quellcodedateien gegebene Programm und beantworten Sie die folgenden Fragen:

Datei foo.h:

```
template<typename T>
T foo( const T & a );
```

Datei foo.cpp:

```
#include "foo.h"

template<typename T>
T foo( const T &a )
{
    return a*a;
}
```

Datei main.cpp:

```
#include "foo.h"

int main( )
{
    int i = 2;
    i = foo(i);
}
```

a) Beschreiben Sie, warum sich aus dem angegebenen Beispiel kein Programm erzeugen lässt.

Aufgabe		
9	Name: ..... Matrikel-Nr.:	

b) Nennen und beschreiben Sie zwei der in der Vorlesung behandelten Ansätze zur Lösung dieses Problems. Notieren sie für jeden dieser Ansätze noch einmal den Quelltext der jeweils geänderten Dateien.

Aufgabe		7 Punkte
	Name:	(2+5)
10	Matrikel-Nr.:	Σ: .....

Arrays und Strukturen in C++

a) Deklarieren Sie in C++ eine Struktur (oder alternativ eine Klasse) `UIntArray`, die ein Array von positiven ganzen Zahlen zusammen mit seiner Länge speichert. **Hinweis**: Bitte verwenden Sie keine Hilfsbibliotheken, wie z.B. die STL in dieser Aufgabe.

b) Schreiben Sie eine Funktion `void remove( unsigned int value, UIntArray aOfA[], unsigned int n )`, die alle Elemente mit dem Wert `value` aus dem gegebenen Array von Arrays löscht. Das Array `aOfA` hat die Länge `n`. **Hinweis**: Falls ein Teilarray ein Element mit dem gesuchten Wert als einziges Element enthält, so soll dieses Teilarray als leeres Teilarray in dem gegeben Array von Arrays verbleiben. Außerdem müssen Sie den in den Arrays verwendeten Speicherplatz nicht freigeben.

Aufgabe		**7** Punkte
	Name:	(4 + 3)
11	Matrikel-Nr.:	Σ: .....

Rekursion in C++

Gegeben sei folgende C++-Funktion:

```
int f( int x, int y )
{
    int z;
    // Stelle X
    if ( x > 0 && y > 0 )
    {
        z = f( x-1, y );
        return f( z, y-1 );
    }
    if ( y > 0 )
        return f( 1, y-1 );
    return x+1;
}
```

a) Zeichnen Sie den Rekursionsbaum bei einem Aufruf mit `f(1,1)`.

Aufgabe		
11	Name:	
	Matrikel-Nr.:	

b) Zeichnen Sie den Zustand des Stacks an Stelle `X` zu dem Zeitpunkt, an dem die Funktion rekursiv mit den Parametern $x = 1$, $y = 0$ aufgerufen wird, wenn der Beginn der Rekursion mit einem Aufruf `f(1,1)` erfolgte. Bitte achten Sie darauf, daß die einzelnen Stackframes in Ihrer Zeichnung erkennbar sind. Sie können Rückgabewert und Framepointer vernachlässigen und für uninitialisierte Variablen den Wert 0 annehmen.

Aufgabe		**6** Punkte
	Name:	(4 + 2)
12	Matrikel-Nr.:	Σ: .....

Statisches und dynamisches Binden in C++

Gegeben sei das folgende C++-Programm:

```
class A
{
public:
    ~A() {}
};

class B : public A
{
public:
    B() {
        m_values = new float[64];
    }
    ~B() {
        delete[] m_values;
    }
private:
    float* m_values;
};

int main() {
    A* x = new B;
    // do something with x
    delete x;
}
```

a) Was passiert an der Stelle, wo `delete x` aufgerufen wird? Welches Problem ergibt sich?

b) Wie muß die Deklaration der Klassen geändert werden, damit das Problem nicht mehr auftritt?