Name:

Dr.-Ing. Hartmut Helmke
Fachhochschule
Braunschweig/Wolfenbüttel
Fachbereich Informatik

Matrikelnummer:	Punktzahl:
Ergebnis:	

Freischuss ☐ F1 ☐ F2 ☐ F3 ☐

Klausur im WS 2001/02 :

Informatik III

Hilfsmittel sind bis auf Computer, Handy etc. erlaubt !	Bitte Aufgabenblätter mit abgeben !

Die Lösungen sind in den meisten Fällen auf einem separaten Aufgabenblatt anzugeben. Falls der Platz auf dem Aufgabenblatt es zulässt, können die Lösungen auch dort notiert werden.
Bitten notieren Sie auf **allen** Aufgabenblättern und separaten Blättern Ihren Namen.
Auf eine absolut korrekte Anzahl der Blanks und Zeilenumbrüche braucht bei der Ausgabe nicht geachtet zu werden. Dafür werden keine Punkte abgezogen.
Es ist nicht unbedingt erforderlich, dass die Programme hocheffizient sind. Wichtiger ist, dass sie verstehbar sind. Eine gute Dokumentation ist keinesfalls *verboten*.
Hinweis: In den folgenden Programmen wird sehr häufig die globale Variable *datei* verwendet. Hierfür kann der Einfachheithalber die Variable *cout* angenommen werden. Die Variable *datei* diente lediglich bei der Klausurerstellung dem Zweck der Ausgabeumlenkung.

Aufgabe 1 : Kontrollstrukturen, Post/Pre-Increment

ca. 10 Punkte
Was gibt das folgende Programm in die Datei *datei* aus?

```
#include <fstream>
using namespace std;

ofstream datei("test.txt", ios::out);


int main()
{
int i(0);
for (i=0; i<5; ++i) {
   datei << i << " ";
   }
datei << endl;
for (i=0; i<5; i++) {
   datei << i << " ";
   }
datei << endl;
for (i=0; i<5; i++) {
   datei << i++ << " ";
   }
datei << endl;
for (i=0; i<5; i++) {
   datei << ++i << " ";
   }
datei << endl;
i=0;
for ( ;  i++ < 5; ) {
   datei << i << " ";
   }

}
```

/*_____*/
Bitte hier die Ausgabe angeben:
/*_____*/

Aufgabe 2 : Parameterübergabe

ca. 10 Punkte
Gegeben ist der folgende Ausschnitt aus einem Programm:

```
/*-----------------------------------------------*/
void funk1(int   i, int  j);
void funk2(int&  i, int* j);
void funk3(float i, int j[]);

int main()
{
int k(14);
int feld[40]; feld[4]=16;

funk1(k, 14);                          /*1*/
funk2(14, &k);                         /*2*/
funk3(static_cast<float>(14), &k);     /*3*/
funk1(feld[4], k);                     /*4*/
funk2(feld[4], feld);                  /*5*/
funk3(14.3f, &k);                      /*6*/
}
```

Welche der Funktionsaufrufe führt zu einer Compilerfehlermeldung? Korrigieren Sie die entsprechenden Funktionsaufrufe, sodass sich weder eine Fehlermeldung noch eine Warnung ergibt. (Es geht hier

nicht darum, ob die entsprechenden Aufrufe sinnvoll sind, sondern nur, dass sie syntaktisch in Ordnung sind.)
/*——————————————————————*/
Bitte hier die korrekten bzw. fehlerhaften Aufrufe ankreuzen und die fehlerhaften mit so wenig Änderungen wie möglich korrigieren.
/*——————————————————————*/

Nr.	O.K.	falsch	eventuelle Korrektur
1.			
2.			
3.			
4.			
5.			
6.			

Aufgabe 3 : nachträgliches Weihnachstgeschenk?

ca. 1 Punkte
Wie nennt man ein Programm zur Übersetzung von einer Programmiersprache in eine andere?

A Compiler

B Commander

C Complainer

D Composer

/*——————————————————————*/
Bitte hier die Antwort angeben (Nur eine ist richtig.):
/*——————————————————————*/

Aufgabe 4 : Zeiger, Adressen

ca. 6+2+3+2 Punkte
Gegeben ist der untenstehende Ausschnitt aus einem Programm:
Geben Sie die Speicherbelegung der Variablen `i, j, k, a1, a2, a3` zu den im Programm mit `/* 1 */, /* 2 */, /* 3 */, /* 4 */, /* 5 */, /* 6 */` gekennzeichneten Zeitpunkten an.
Gehen Sie dabei davon aus, dass der Speicher auf dem Stack von den niedrigen zu den hohen Adresse wächst, dass also wie gezeigt, die Variable i auf dem Stack eine niedrigere Adresse wie die später angelegte Variable j belegt. Außerdem soll jeder Pointer 4 Bytes im Speicher belegen, d.h. `sizeof(void*)` ergäbe 4.

a.)
Es reicht aus, wenn Sie in jeder Spalte nur die Veränderungen zur vorherigen Spalte eintragen.

Speicherbelegung:

Variable	Adresse	/* 1 */	/* 2 */	/* 3 */	/* 4 */	/* 5 */	/* 6 */
i	1000	20					
j	1004	30					
k	1008	40					
a1	1060						
a2	1064						
a3	1068						

```
#include <fstream>
using namespace std;

ofstream datei("test.txt", ios::out);
int Verdaechtig(int* a1, int a2, int* a3)
{
a2 = a2;              /* 1 */
*a3 = 24;             /* 2 */
++a3;                 /* 3 */
*a3 = 61;             /* 4 */
a3 = a1;              /* 5 */
*a3 = a2;             /* 6 */
return *(--a3);
}
/*-----------------------------------------------*/
int main()
{
int i = 20;
int j = 30;
int k = 40;
/* .. */
datei << Verdaechtig(&k, j, &i) << endl;
datei << i << " " << j << " " << k << endl;
return 0;
}
```

b.)

Was bedeutet die Anweisung `*(--a3)` (return-Anweisung von `Verdaechtig`) in Funktions-Schreibweise (`a - b;` bedeutet in Funktions-Schreibweise `operator-(a, b)`)? Die Lösung dieser Aufgabe könnte auch für die Lösung von Teil a) sehr hilfreich sein.

/*_____*/

Bitte hier die Funktions-Schreibweise angeben:

/*_____*/

c.)

Was gibt das `main`-Programm in die Datei *datei* aus?

/*_____*/

Bitte hier die Ausgabe angeben:

/*_____*/

d.)

Was gibt das Programm in die Datei *datei* aus, wenn die letzte Anweisung der Funktion `Verdaechtig` geändert wird in:? Was gibt das `main`-Programm in die Datei *datei* aus?

```
return *(a3--);  /* statt return *(--a3); */
```

/*_____*/

Bitte hier die Ausgabe angeben:

/*_____*/

Aufgabe 5 : STL, Konstruktoren, Destruktoren, Polymorphie

ca. 4+4+4+7+9+3 Punkte

Gegeben ist das folgende Hauptprogramm:

```
#include <fstream>
#include <list>
#include <algorithm>
using namespace std;

ofstream datei("test.txt", ios::out);

/* Der Teil ab hier wird bei den
   folgenden Aufgaben vorgestellt. */

int main()
{
// Aufgabe a
datei << "funk1\n"; funk1(); datei << endl;
// Aufgabe b
datei << "\nfunk2\n"; funk2(); datei << endl;
// Aufgabe c
datei << "\nfunk3\n";funk3(); datei << endl;
// Aufgabe d und e und f
datei << "\nfunk4\n";funk4(); datei << endl;

return 0;
}
```

a)

Was gibt der Aufruf der folgenden Funktion `funk1` in die Datei `datei` aus. Als Hilfe sind im Anhang zur Wiederholung nochmals die Definition des STL-Algorithmus `for_each` sowie ein Ausschnitt aus dem Destruktor von der STL-Klasse `list` angegeben.

```
void WPrint(int i)  {datei << i << " ";}
void PPrint(int* pi){datei << *pi << " ";}

void funk1()
{
   typedef list<int>  T_IntList;
   typedef list<int*> T_PIntList;

   T_IntList wListe;
   T_PIntList pListe;

   int i1(1), i2(2), i3(3);

   wListe.push_back(i1+10);
   wListe.push_back(i2+10);
   wListe.push_back(i3+10);

   pListe.push_back(&i1);
   pListe.push_back(&i2);
   pListe.push_back(&i3);

   for_each(++wListe.begin(),wListe.end()--,
         WPrint);
   datei << endl;
   for_each(pListe.begin(),pListe.end(),
         PPrint);
}
```

/*_____*/

Bitte hier die Ausgabe angeben:

/*_____*/

b)

```
/***********************************/
class Zahl
{
public:
   Zahl(int w);
   virtual ~Zahl();
   void PrType() const;
   virtual void PrWert() const;
protected:
   int wert;
};

Zahl::Zahl(int w): wert(w) {
   datei << "+Z" << wert << " ";
}
Zahl::~Zahl() {
   datei << "-Z" << wert << " ";
}

/***********************************/
void funk2()
{
   list<Zahl>  zListe;
   Zahl z1(1), z2(2);
   zListe.push_back(z1);
   zListe.push_back(z2);
}
```

Was gibt der Aufruf der Funktion `funk2` in die Datei `datei` aus.

/*_____*/

Bitte hier die Ausgabe angeben:

/*_____*/

c)

```
/*********************************/
void funk3()
{
list<Zahl*> zpListe;
Zahl z1(1);
zpListe.push_back(&z1);
zpListe.push_back(&z1);
}
```

Was gibt der Aufruf der Funktion `funk3` in die Datei `datei` aus.

/*_____*/

Bitte hier die Ausgabe angeben:

/*_____*/

d)

```
/*********************************/
void Zahl::PrType() const {
   datei << "ZAHL" << " ";
}
void Zahl::PrWert() const {
   datei << wert << " ";
}

class Real : public Zahl
{
public:
Real(int w);
virtual ~Real();
void PrType() const;
virtual void PrWert() const;
};

Real::Real(int w): Zahl(w) {
datei << "+R" << wert << " ";
}
Real::~Real() {
datei << "-R" << wert << " ";
}

void Real::PrType() const {
datei << "REAL " << " ";
}
void Real::PrWert() const {
datei << wert << ".0 ";
}

void ZPrint(Zahl* pz){
pz->PrType(); pz->PrWert();
}

void ZPrintConv(Zahl* az){
Zahl z(*az);
Zahl* pz = &z;
pz->PrType(); pz->PrWert();
}
       // siehe Anhang fuer for_each
void funk4()
{
   list<Zahl*> zpListe;
   Zahl z1(1);
   Real* pr2 = new Real(2);
   Real r3(3);
   Zahl* pz=&r3;

   zpListe.push_back(&z1);
   zpListe.push_back(pr2);
   zpListe.push_back( pz);

#ifdef WITH_FOREACH
      datei << endl;
      for_each(zpListe.begin(),zpListe.end(),
               ZPrint);
      datei << endl;
      for_each(zpListe.begin(),zpListe.end(),
               ZPrintConv);
      datei << endl;
#endif /* WITH_FOREACH */
}
```

Was gibt der Aufruf der Funktion `funk4` in die Datei `datei` aus, wenn man davon ausgeht, dass das

Programm ohne die Makrodefinition `WITH_FOREACH` übersetzt wurde, d.h. die Zeilen zwischen den Präprozessorbefehlen `#ifdef WITH_FOREACH` und `#endif` nicht existieren würden.
Tipp: Gehen Sie Anweisung für Anweisung durch.
/*———————————————————————*/
Bitte hier die Ausgabe angeben:
/*———————————————————————*/

e)
Was gibt der Aufruf der Funktion `funk4` **zusätzlich** in die Datei `datei` aus, wenn das Programm mit der Makrodefinition `WITH_FOREACH` übersetzt wurde, d.h. die beiden Zeilen mit den Präprozessorbefehlen `#ifdef WITH_FOREACH` und `#endif` selbst nicht existieren würden.
Tipp: Gehen Sie Anweisung für Anweisung durch.
/*———————————————————————*/
Bitte hier die **zusätzliche** Ausgabe angeben:
/*———————————————————————*/

f)
Wo treten durch Aufruf von Funktion `funk4` Speicherlecks auf? Korrigieren Sie das Programm daraufhin.
/*———————————————————————*/
Ihre Lösung
/*———————————————————————*/

Aufgabe 6 : Zeigersemantik

ca. 15+2 Punkte
Implementieren Sie für die folgende Klasse *Schlange* einen Destruktor einen Copy-Konstruktor und einen Zuweisungsoperator.
Hinweis: Sie können Destruktor, Copy-Konstruktor und Zuweisungsoperator dieser drei Teilaufgaben `inline` implementieren, d.h. es wird angenommen, dass der von Ihnen hier implementierte Code im public-Teil nach dem Konstruktor von `Schlange` folgt.

```
class Schlange {  /* FIFO*/
    int *Data;
    int maxsize;
    int n;
  public:
     Schlange(int siz=20)
  { Data = new int[maxsize=siz]; n = 0; }

    /* hier sollte Ihr Code des
        Destruktors etc. stehen */

    void Einfuegen(int x) { Data[n++] = x; }
    int  Groesse()        { return n; }
    int  GibErstes() {
                int wert = Data[0];   --n;
                for (int i=0; i<n; ++i) {
                     Data[i]=Data[i+1];
                     }
                return wert;
               }
};
```

Nach Ihrer Implementierung soll das folgende Programm lauffähig sein, ohne Speicherlecks zu hinterlassen. Die Ausgabe des Programm soll dann `1 2 3 4` und `1 2 3 4` sein.

```
int main(){
Schlange s1;
s1.Einfuegen(1);
s1.Einfuegen(2);
s1.Einfuegen(3);
s1.Einfuegen(4);

Schlange s2(s1), s3(44);
datei << s2.GibErstes() << " ";
datei << s2.GibErstes() << " ";
datei << s2.GibErstes() << " ";
datei << s2.GibErstes() << " ";

s2 = s3 = s1 = s1;
datei << s2.GibErstes() << " ";
datei << s2.GibErstes() << " ";
datei << s2.GibErstes() << " ";
datei << s2.GibErstes() << " ";
return 0;
}
```

/*———————————————————————*/
a)
Bitte die Lösung auf einen Extrablatt notieren.
/*———————————————————————*/

b)
Welche der Methoden und Elemente der Klasse `Schlange` (mit Ihren zusätzlich implementierten) können als konstant implementiert werden.
/*_____*/
Als konstant können implementiert werden:
/*_____*/

Aufgabe 7 : Templates

ca. 17+3 Punkte
Schreiben Sie eine Klasse `UeberwZahl`, die es ermöglicht, dass einer Variablen nur ganz bestimmte Werte aus einem Intervall zugewiesen werden können. Falls versucht wird, andere Werte zuzuweisen, bleiben die bisherigen Werte erhalten.
Das folgende Programm zeigt eine beispielhafte Anwendung der Klasse, die in der Datei `UeberwZahl.h` definiert wird. Ihre Methoden werden `alle` in der Datei `UeberwZahl.cxx` definiert.
Vergessen Sie nicht, die notwendigen `include`-Anweisungen und einen Include-Wächter zu implementieren. Achten Sie auch darauf, soweit wie möglich das Schlüsselwort `const` zu verwenden.

```
#include <fstream>
using namespace std;
#include "UeberwZahl.h"

ofstream datei("test.txt", ios::out);

int main()
{
// Wert 14, wert aus [0..22]
UeberwZahl<int> wert(14, 0, 22);
// Wert 14.4, d aus [-1..100]
UeberwZahl<double> d(14.4, -1, 100);

wert = 61;
wert = 17;
wert = -1;

d = 11.0;

datei << "*** " << wert << " ***" << endl;
datei << "*** " << d    << " ***" << endl;

datei << "*** wert:" << wert.GetWert()
      << " *** " << endl;
datei << "*** d   :" << d.GetWert()
      << " *** " << endl;
return 0;
}
```

Dieses Programm könnte z.B. die folgende Ausgabe erzeugen:

```
Ueberwachung von 14 in [0..22]
Ueberwachung von 14.4 in [-1..100]
Neuer Wert falsch 61 in [0..22]
Alter Wert 14 bleibt.
Neuer Wert OK 17 in [0..22]
Neuer Wert falsch -1 in [0..22]
Alter Wert 17 bleibt.
Neuer Wert OK 11 in [-1..100]
***  17 in [0..22] ***
***  11 in [-1..100] ***
*** wert:17 ***
*** d   :11 ***
```

In Ihrer Implementierung brauchen die Methoden der Klasse selbst keine Ausgaben zu erzeugen. Die angegebenen Ausgaben, dienen lediglich zur Veranschaulichung der Funktionalität.
a.)
Es ist ausreichend, wenn obiges Programm übersetzt werden kann und die entsprechende Funktionalität (ohne Ausgaben) aufweist. Zusätzliche Methoden der Klasse, die eventuell für weitere Anwendungen sinnvoll sein könnten, sollen **nicht** implementiert werden, d.h. implementieren Sie die erforderlichen Elemente der Klasse, einen geeigneten Konstruktor, einen geeigneten Operator (Welchen?) und die Methode `GetWert`.
b.)
Falls man die Klasse `UeberwZahl` für beliebige Typen, nicht nur für Standardtypen, verwenden will, welche Eigenschaften müssen diese Typen dann mindestens aufweisen? (Welche Methoden etc. muss es für sie geben, auch wenn diese automatisch vom C++-System erzeugt werden sollten?)
/*_____*/
Notieren Sie die Lösungen auf einem Extra-Blatt.
/*_____*/

Anhang 1: Destruktor von `list`

```
template <class T>
list<T>::~list(){
   //...
   erase(begin(), end());
   //...
   }
```

Anhang 2: Schablonenfunktion `for_each`

```
template<class It, class Fn> inline
Fn for_each(It F, It L, Fn Op) {
for (; F != L; ++F)
Op(*F);
return (Op);
}
```

Anhang 3: Geplante Punktevergabe

Punktziel	Sonderpunkte	erreicht
A1: 10 P		
A2: 10 P		
A3: 1 P		
A4: 15 P		
A5: 31 P		
A6: 15 P		
A7: 20 P		
Summe 102 P		