Name:

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

Matrikelnummer:	*Punktzahl:*
Ergebnis:	

Freiversuch ☐ *F1* ☐ *F2* ☐ *F3* ☐

Klausur im WS 2007/08 :

☐ ☐ **Programmierkonzepte**
Informatik III

Medieninformatik ☐ *Praktische Informatik* ☐ *Technische Informatik* ☐
Informatik B. Sc. ☐ *Technische Informatik B. Sc.* ☐

Hilfsmittel sind bis auf Computer, Handy etc. erlaubt !	Bitte Aufgabenblätter mit abgeben !
Austausch von Hilfsmitteln mit Kommilitonen ist **nicht** erlaubt !	

Die Lösungen können größtenteils hier auf dem Aufgabenblatt angegeben werden. Sie dürfen aber auch Ihre Lösungen, falls erforderlich, auf separaten Blättern notieren.
Bitte notieren Sie auf **allen** Aufgabenblättern und separaten Blättern Ihren Namen bzw. Ihre Matrikelnummer. Auf eine absolut korrekte Anzahl der Blanks und Zeilenumbrüche braucht bei der Ausgabe nicht geachtet zu werden. Dafür werden keine Punkte abgezogen.
Hinweis: In den folgenden Programmen wird manchmal die globale Variable *datei* verwendet. Hierfür kann der Einfachheit halber die Variable *cout* angenommen werden. Die Variable *datei* diente lediglich bei der Klausurerstellung dem Zweck der Ausgabeumlenkung.

Geplante Punktevergabe

Planen Sie pro Punkt etwas mehr als eine Minute Aufwand ein.

Punktziel	Sonderpunkte	erreicht
Übungsaufgaben: 10 P.		
A1: 16 (2+3+3+8) P.		
A2: 18 (1+1+5+1+1+2+2+4+1) P.		
A3: 20 (2+2+1+3+3+5+4) P.		
A4: 11 (1,5+1,5+6+2) P.		
A5: 19 (2+1+1+6+2+2+1+2+2) P.		
A6: 16 (1+2+1+2+1,5+1+1+2+2+2,5) P.		
Summe 100 P.		

Exercise 1 : StackHeapMemory

approx. 16 (2+3+3+8) pts

a.) Illustrate (graphically) the contents of the stack memory at the end of the following function.

```
void f1() {
   double d = 12.4;
   int i = 41;
   }
```

(*——- Solution here. ———*)

b.) Illustrate (graphically) the contents of the stack memory at the end of the following function.

```
void f3() {
   int i = 41;
   int* pi = &i;
   int* p2 = pi;
   }
```

(*——- Solution here. ———*)

c.) Illustrate (graphically) the contents of the stack **and** heap memory at the end of the following function.

```
void f4() {
   int j = 14;
   int* pj = new int(88);
   }
```

(*——- Solution here. ———*)

The following functions are given:

```
void help(int i, int& j, int* k) {
   int start = i;  /* 1 */
   i = 22;
   j = 33;
   *k = 44; /* 2 */
   }
void f2() {
   int a(4);
   int b = 5;
   int c(6);
   //  help(&*a, &*b, &*c); // correct one
   cout << a <<"," << b <<"," << c << endl;
   }
```

Which of the following 5 calls of function `help` is/are syntactically correct?

```
   help(a, b, &c);
   help(&a, &b, &c);
   help(&a, *b, &c);
   help(a, *b, c);
   help(&a, *b, c);
```

(*——- Mark correct/wrong one in listing ———*)

Illustrate the stack memory contents at time /* 1 */.
(*——- Solution here. ———*)

Illustrate the stack memory contents at time /* 2 */.
(*——- Solution here. ———*)

What is the screen output of function call `f2`, when using (one of) the correct calls of `help`?
(*——- Solution here. ———*)

Exercise 2 : ClassesAndCreation

approx. 18 (1+1+5+1+1+2+2+4+1) pts

Given the following class declarations:

```
class Furniture { // Möbel
public:
   virtual string getType() const
     {return "Unknown,";}
  };

class Chair: public Furniture { // Stuhl
public:
   Chair() {datei << "+C ";}
   ~Chair() {datei << "-C ";}
    virtual string getType() const
     {return "Chair,";}
  };
```

a.) What is the output of function call `f1`?

```
void f1(){
   datei << "Start ";
   Chair c1;
   datei << "End ";
   }
```

(*——- Solution here. ———*)

b.) What is the output of function call `f2`?

```
void f2(){
   datei << "Start ";
   Chair c1;
   Chair c2(c1);
   datei << "End ";
   }
```

(*——- Solution here. ———*)

c.) What is the output of function call `f3`?

```
void f3(){
   datei << "Start ";
   Furniture* f1 = new Chair();
   datei << "End ";
   }
```

(*——- Solution here. ———*)

c2.) What is the output of function call `f4`?

```
void f4(){
   datei << "Start ";
   Chair* c1 = new Chair();
   datei << "End ";
   }
```

(*——- Solution here. ———*)

c3.) What is the output of function call `f5`?

```
void f5(){
   datei << "Start ";
   Chair* c1 = new Chair();
   delete c1;
   datei << "End ";
   }
```

(*——- Solution here. ———*)

c4.) What is the output of function call `f6`?

```
void f6(){
   datei << "Start ";
   Furniture* f1 = new Chair();
   delete f1;
   datei << "End ";
   }
```

(*——- Solution here. ———*)

Class `Table` is another derived class.

```
class Table: public Furniture { // Tisch
public:
   Table(int le) {datei << "+T "; legs = le;}
   ~Table() {datei << "-T ";}
    virtual string getType() const
      {return "Table,";}
   void setColour(string c);
private:
   int legs;
   string colour;
   };
```

d.) Why does the following code fragment result in a syntax error when being compiled?

```
Table tables[12];
```

(*——- Solution here. ———*)

e.) Please implement the method `setColour` in the source code file of the class.
(*——- Solution here. ———*)

f.) Please implement `the copy constructor` of the derived class in the source code file of the class. Avoid to implement needless code.
(*——- Solution here. ———*)

g.) Implement the include guard of the header file.
(*——- Solution here. ———*)

Furthermore we have a container class to manage the class `Furniture` and its derived classes:

```
class Room { // Raum
public:
   Room(): t1(4) {datei << "+R ";}
   ~Room() {datei << "-R ";}
private:
   Table t1;
   Chair chairs[4];
   };
```

h.) What is the output of function call `f7`?

```
void f7(){
   datei << "Start ";
   Room r1;
   datei << "End ";
   }
```

(*——- Solution here. ———*)

i.) Please explain the necessity of the initialization list in the constructor of `Room` (You may get an extra point for specifying the character sequence, which constitutes the initialization list here)?
(*——- Solution here. ———*)

Exercise 3 : STLandTesting

approx. 20 (2+2+1+3+3+5+4) pts

Below we still use the class declarations from the previous exercise.

We use, however, an alternative implementation of the container:

```
class Room1 { // Raum
public:
   Room1() {}
   ~Room1() {}
   void addFurniture(Furniture& f);
   const Furniture* getFurniture(int i) const;
   void print(ostream& datei) const;
private:
   vector<Furniture*> cont;
   };
```

a.) Which advantages do result from the alternative implementation (abstain from the possible additional effort)?

(*——- Solution here. ———*)

b.) Why must we use the container type `vector<Furniture*>` instead of `vector<Furniture>`?

(*——- Solution here. ———*)

c.) Why doesn't the container type `vector<Table* >` make sense?

(*——- Solution here. ———*)

d.) Please implement the remaining lines of method `Room1::print`.

```
// Ausgabe aller Möbel im Container durch
// Aufruf von getType in einer Schleife
// Output of all furniture in the container
// by calling getType in a loop
void Room1::print(ostream& d) const{
   vector<Furniture*>:: const_iterator iter;
```

(*——- Solution here. ———*)

e.) Please implement the method `Room1::addFurniture` (probably only one line of source code).

(*——- Solution here. ———*)

f.) Why is the parameter type `const Furniture&` not allowed here?

(*——- Use additional page. ———*)

g.) Please describe (only describing plain English text) a test for the method `Room1::addFurniture`. Use the method `Room1::getFurniture` for that purpose:

```
// Das i-te Möbelstück im Container wird geliefert .
// return of furniture number i or NULL-Pointer
const Furniture* Room1::getFurniture(int i) const {
   // Type-Cast/Umwandlung to unsigned int
   vector<Furniture*>::size_type no = i;
   if ((no >= 0) && (no < cont.size())) {
      return cont[i];
      }
   else {
      return NULL;
      }
   }
```

(*——- Use additional page. ———*)

h.) Please implement the just described test in C++.

(*——- Use additional page. ———*)

Exercise 4 : VirtualMethods

approx. 11 (1,5+1,5+6+2) pts

Below we still use the class declarations from the previous exercise (in particular the virtual method `getType`).
The relevant class declarations are presented again to keep track:

```
class Furniture { // Möbel
public:
   virtual string getType() const
     {return "Unknown,";}
  };

class Chair: public Furniture { // Stuhl
public:
   Chair() {datei << "+C ";}
   ~Chair() {datei << "-C ";}
   virtual string getType() const
     {return "Chair,";}
  };

class Table: public Furniture { // Tisch
public:
   Table(int le) {datei << "+T "; legs = le;}
   ~Table() {datei << "-T ";}
   virtual string getType() const
     {return "Table,";}
  void setColour(string c);
private:
   int legs;
   string colour;
   };
```

a.) What is the output of function call `f10`? You may ignore outputs resulting from constructors and destructors.

```
void f10() {
   Furniture f1;
   Chair c1;
   datei << f1.getType();
   datei << c1.getType();
   }
```

(*——- Solution here. ———*)

b.) What is the output of function call `f11`? You may ignore outputs resulting from constructors and destructors.

```
void f11() {
   Furniture* f1 = new Furniture();
   Chair* c1 = new Chair();
   datei << f1->getType();
   datei << c1->getType();
   }
```

(*——- Solution here. ———*)

c.) Illustrate (graphically) the contents of the stack **and** heap memory before the `for`-loop in the following program fragment.

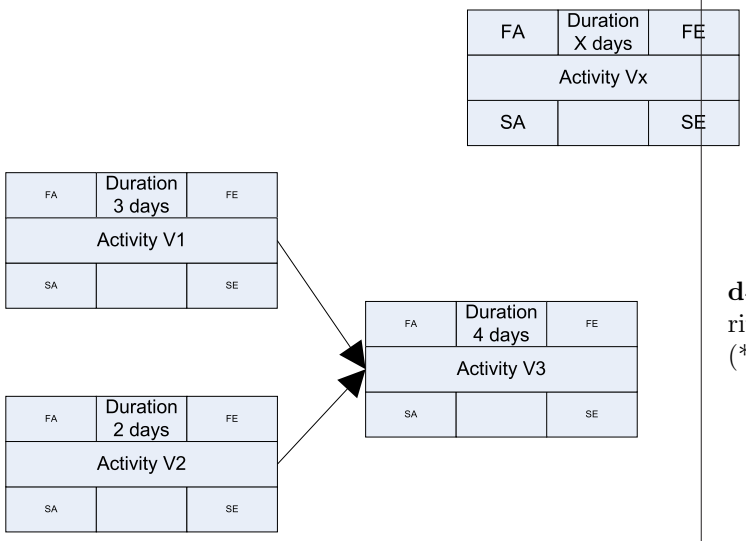
```
void f12() {
   Furniture* pf = new Furniture();
   Chair* pc = new Chair();
   Table t(5);
   Table* pt = &t;

   Furniture* arr[]={pf, pc, &t, pt};
   // *1* Stack and Heap-Memory-Contents ???
   for (int i=0; i<4; ++i) {
      datei << arr[i]->getType();
      }
   }
```

(*——- Solution here. ———*)

d.) What is the output of function call `f12`? You may ignore outputs resulting from constructors and destructors.
(*——- Solution here. ———*)

Exercise 5 : TextQuestions

approx. 19 (2+1+1+6+2+2+1+2+2) pts



a.) Given the network plan above with the activities V1, V2 and V3. Enter the earliest start (FA), the earliest end (FE), the latest start (SA) and the latest end (SE) of each activity into the network plan.
(*——- Enter solution in graphics. ———*)

b.) Which activities are on the critical path?
(*——- Solution here. ———*)

c.) Specify a C++-code fragment, in which the use of the post-increment operator results in a different behaviour than the use of the pre-increment operator (you may use `cout` for demonstration purposes).
(*——- Solution here. ———*)

d.) Please implement **only** the interface of a C++-function which returns both the square and the third power of an input parameter.
(*——- Solution here. ———*)

d2.) Is this function a real function (German: echte Funktion) or a procedure function (assigment function, German: Anweisungsfunktion)?
(*——- Solution here. ———*)

d3.) Please implement a test for the functionality of this function.
(*——- Solution here. ———*)

d4.) Please implement now the C++ function (algorithm) itself.
(*——- Solution here. ———*)

e.) Please implement a **very, very short** C++ code fragment, whose execution allocates both heap memory and stack memory.
(*——- Solution here. ———*)

f.) Please implement a C++ code fragment, which first creates an instance of class X on the program stack and then creates an instance of class X in the heap memory.
(*——- Solution here. ———*)

g.) Is the same also possible with a Java code fragment? Please explain?
(*——- Solution here. ———*)

h.) Please comment the following assertion:
Private C++-methods are useless, because they can never be called.
(*——- Solution here. ———*)

i.) Please specify an example for a declaration (definition not necessary) of a class with a minimal standard interface (German: minimale Standard-Schnittstelle).
(*——- Use additional page. ———*)

Exercise 6 : TextQuestionsTeam

approx. 16 (1+2+1+2+1,5+1+1+2+2+2,5) pts

a.) Which step directly follows the design in the waterfall process model?
(*——- Solution here. ———*)

b.) Why does normally the so-called *truck factor* decrease if the XP base technique *common ownership* is used?
(*——- Solution here. ———*)

c.) Your program contains some code fragments repeatedly. What do you propose to overcome this drawback (one correct word is enough)?
(*——- Solution here. ———*)

d.) Why is redundant code a drawback for the future project success?
(*——- Solution here. ———*)

e.) Four variables are important when planing a software project. One process variable are the *costs*. Which are the other three ones?
(*——- Solution here. ———*)

f.) Explain the heuristics of the *weather of yesterday* in the contents of the XP planning game (iteration planning)?
(*——- Solution here. ———*)

g.) Please specity *Brooks's law*.
(*——- Solution here. ———*)

h.) Explain it, because it may sound a little bit strange at first glance.
(*——- Solution here. ———*)

i.) Which advantages do result from *test first* (specifying the tests before starting the implementation) (a few short sentences)?
(*——- Solution here. ———*)

j.) The team has planned to perform tasks with a planned effort of 12 ideal days during the last iteration (duration of 25 working days). They, however, completed only tasks with an effort of 9 ideal days. Additionally they completed tasks, not planned, with an effort of 4 ideal days. Please calculate the *load factor* (calculation steps required).
(*——- Solution here. ———*)

The new iteration lasts 10 working days. How many tasks (in ideal days) the team should plan to perform if the load factor is used as the basis for decision-making (don't forget the calculation steps)?
(*——- Solution here. ———*)