FH–Aachen (Standorte Aachen, Jülich, Köln)
Forschungszentrum Jülich
Rechen– und Kommunikationszentrum der RWTH–Aachen

Prof. Dr. W. Hanrath, Prof. Dr. A. Terstegge, B. Sc. M. Politze

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Bachelorstudiengang "*Scientific Programming*"
MatSe–Ausbildung
C++–Klausur, SS 2012

Datum: 13. Juli 2012

Name: _____

Vorname: _____

Matr.-Nr.: _____

Unterschrift: _____

		max. Punktzahl
Aufgabe 1)		(16)
Aufgabe 2)		(14)
Aufgabe 3)		(8)
Aufgabe 4)		(10)
Aufgabe 5)		(10)
Aufgabe 6)		(14)
Aufgabe 7)		(10)
Aufgabe 8)		(18)
Gesamtpunkte:	Note:	

Name: _____ Matr.–Nr.: _____

Aufgabe 1

16 Punkte

a) Definieren Sie einen Zeiger auf ein `double`:

_____

b) Definieren Sie einen Zeiger auf ein konstantes `double`:

_____

c) Definieren Sie einen konstanten Zeiger auf ein `double`:

_____

d) Definieren Sie einen konstanten Zeiger auf ein konstantes `double`:

_____

e) Definieren Sie eine (statische) 3×4–Matrix vom Typ `double`:

_____

f) Definieren Sie eine (statische) 3×4–Matrix von Zeigern auf `double`:

_____

g) Es sei `double **A; int n, m;` gegeben, wobei die Variablen `n` und `m` jeweils positive Werte beinhalten.

- Man schreibe ein Programmfragment derart, dass `A` danach wie eine `n`×`m`–Matrix vom Typ `double` verwendet werden kann:

_____

_____

_____

_____

- Man schreibe ein Programmfragment, mit dem der für `A` im letzten Aufgabenteil allokierte Speicher wieder freigegeben wird:

_____

_____

_____

_____

Aufgabe 2

14 Punkte

Das Objekt `a` vom Typ `A` ist einige 100 Kilobyte groß.

a) `a` soll durch eine Funktion `f` abgeändert werden. Hierzu soll `a` in geeigneter Form als Funktionsargument an `f` übergeben werden. Hierzu gibt es zwei unterschiedliche Möglichkeiten.
Zu diesen beiden Möglichkeiten gebe man jeweils eine mögliche Deklaration von `f` an und einen entsprechenden Aufruf von `f`:

- 1. Möglichkeit:

  Deklaration von `f`: _____

  Aufruf von `f`: _____

- 2. Möglichkeit:

  Deklaration von `f`: _____

  Aufruf von `f`: _____

b) `a` soll an eine Funktion `f` als Funktionsargument übergeben, soll aber innerhalb der Funktion nicht abgeändert werden. Aus Performance–Gründen soll in `f` **keine** Kopie von `a` erzeugt werden.
Skizzieren Sie zwei Möglichkeiten (jeweils Funktionsdeklaration und Aufruf), wie das realisiert werden kann:

- 1. Möglichkeit:

  Deklaration von `f`: _____

  Aufruf von `f`: _____

- 2. Möglichkeit:

  Deklaration von `f`: _____

  Aufruf von `f`: _____

Aufgabe 3

8 Punkte

a) Beantworten Sie folgende Fragen zum Thema *Inline–Funktionen*:

(i) Müssen Inline–Funktionen vor ihrem Aufruf definiert sein?

ja ☐ nein ☐

(ii) Können Inline–Funktionen in einer Headerdatei definiert werden?

ja ☐ nein ☐

(iii) Können Inline–Funktionen in einer Headerdatei definiert werden, und kann diese Headerdatei dann ohne weitere Vorkehrungen in ein– und demselben Quelltext mehrfach eingebunden werden?

ja ☐ nein ☐

(iv) Werden Inline–Funktionen vom Compiler immer so umgesetzt, dass an entsprechender Stelle im ausführbaren Programm die Befehle der Inline–Funktion eingefügt werden, ohne dass es zu einem Funktionssprung kommt?

ja ☐ nein ☐

(v) Sind im Klassenkörper definierte Memberfunktionen automatisch inline?

ja ☐ nein ☐

(vi) Ist es richtig, dass außerhalb des Klassenkörpers definierte Memberfunktionen nicht inline sein können?

ja ☐ nein ☐

b) Beantworten Sie folgende Fragen zum Thema *Standardparameter einer Funktion*:

(i) Standardparameter einer Funktion sind anzugeben bei der

Funktiondefinition: ☐ Funktionsdeklaration: ☐

(ii) Wer kümmert sich um die Umsetzung von Standardparametern einer Funktion:

Laufzeitsystem: ☐ Compiler: ☐

Aufgabe 4

10 Punkte

Eine Klasse `A` sei wie folgt definiert:

```
class A
{ private:
   int A_komp;
  public:
    A(int i);
    ...
};
```

(neben dem aufgeführten parameterbehafteten Konstruktor habe die Klasse `A` keine weiteren Konstruktoren!)

a) Was ist bei der Definition einer von `A` abgeleiteten Klasse `B`

```
class B : public A
{ ...
};
```

im Hinblick auf `B`–Konstruktoren zu beachten?

b) Es sei folgende Klassenhirarchie definiert:

```
class B: public A { ... };
class C: public A { ... };

class D: public B, public C { ... };
```

(i) Wie oft ist in einem `D`–Objekt ein `A`–Bestandteil vorhanden? ☐ 1× ☐ 2×

(ii) `A`–Konstruktoren müssen aufgeführt sein:

☐ in Initialisierungslisten von `B`– und `C`–Konstruktoren

☐ in Initialisierungslisten von `D`–Konstruktoren

c) Es seien folgende Klassenhirarchie definiert:

```
class B: virtual public A { ... };
class C: virtual public A { ... };

class D: public B, public C { ... };
```

(i) Wie oft ist in einem `D`–Objekt ein `A`–Bestandteil vorhanden? ☐ 1× ☐ 2×

(ii) `A`–Konstruktoren müssen aufgeführt sein:

☐ in Initialisierungslisten von `B`– und `C`–Konstruktoren

☐ in Initialisierungslisten von `D`–Konstruktoren

Aufgabe 5

10 Punkte

Werfen Sie einen Blick auf folgende Klassendefinition:

```
class Bruch
{ private:
    int zaehler;
    int nenner;
    int id;              // Identifizierung
    static int gesamt;   // Anzahl aller jemals erzeugten Brueche
    static int aktuell; // Anzahl der aktuellexistierenden Brueche

  public:
    Bruch ( int z = 0; int n = 1);
    Bruch (const Bruch &b);
    ~Bruch();
    static void zeigAnzahl();
    void zeig() const;
    void erweitern(int f);
    void kuerzen();
    int ggT() const;
};
```

und beantworten Sie folgende Fragen:

a) Welche der Funktionen besitzt einen `this`–Zeiger?

	ja	nein
`Bruch`		
`~Bruch`		
`zeigAnzahl`		
`erweitern`		

b) Welchen Typ besitzt der `this`–Zeiger der Funktion `kuerzen`?

_____

c) Welchen Typ besitzt der `this`–Zeiger der Funktion `zeig`?

_____

d) Welcher der folgenden Funktionsaufrufe ist erlaubt, welcher nicht?

	erlaubt	nicht erlaubt
`const Bruch b1(2,3); b1.erweitern(4);`		
`const Bruch b2(9,5); b2.zeig();`		
`Bruch b3(4,6); int g = b3.ggt();`		
`Bruch b4(4,6); b4.kuerzen();`		

Aufgabe 6

14 Punkte

a) Es sei die Klasse A wie folgt deklariert (und definiert):

```
class A {
  public:
  A(int i) { /* ... */ }
  A( const A& a) { /* ... */ }
  ...
};
```

Erläutern Sie zu den durch ① bis ⑦ markierten Stellen in folgendem Programmfragment, was an dieser Stelle geschieht (welche Konstruktoren aufgerufen werden) bzw. was an dieser Stelle falsch ist!

```
...
int main(void)
{
  A a;      //   ①
  A b(7);   //   ②
  A c=b;    //   ③
  A d=8;    //   ④
  A e(b);   //   ⑤
  ...
  e = 4;    //   ⑥
  e = c;    //   ⑦
  ...
}
```

① _____

② _____

③ _____

④ _____

⑤ _____

⑥ _____

⑦ _____

b) Von obiger Klasse `A` werde wie folgt die Klasse `B` abgeleitet:

```
class B : public A {
  public:
  B(void)  { /* ... */ }
  B(int i) { /* ... */ }
   ...
};
```

Erläutern Sie zu den durch ① bis ③ markierten Stellen in folgendem Programmfragment, was an dieser Stelle geschieht (welche Konstruktoren—auch die der Klasse `A`—aufgerufen werden) bzw. was an dieser Stelle falsch ist!

```
...
int main(void)
{
   B a;      //  ①
   B b(7);   //  ②
   B c = 8;  //  ③
...
}
```

① _____

② _____

③ _____

Aufgabe 7

10 Punkte

Erläutern Sie folgende Begriffe:

a) Was ist eine *virtuelle* Methode (wie wird sie definiert, was ist der Unterschied zu *normalen* Methoden)?

b) Was ist eine *rein virtuelle* Methode (wie wird sie vereinbart, was ist deren Bedeutung)?

c) Was ist eine *abstrakte Basisklasse*?

Aufgabe 8

18 Punkte

Entwerfen und implementieren Sie eine `template`–Klasse

```
template <typename T>
class Feld
{ ...
};
```

zur Realisierung eines Feldes vom Type `T` mit Indexüberprüfung.

Genauere Anforderungen:

a) Konstruktor mit `unsigned`–Argument für die Feldlänge, eigentliches Feld ist dynamisch anzulegen und die Klasse soll zur Vererbung geeignet sein.

b) Wegen der *dynamischen Komponente* sind folgende Methoden vernünftig zu implementieren:

(i) Destruktor: `~Feld();`

(ii) Copy–Konstruktor: `Feld(const Feld &o);`

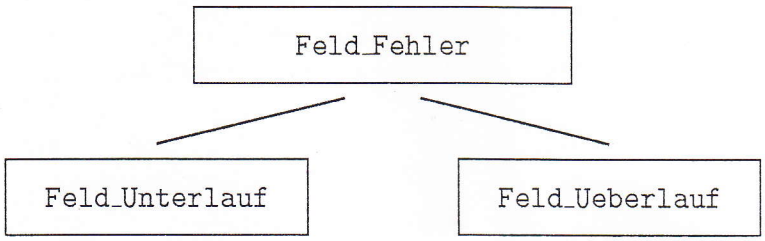
(iii) Move–Konstruktor: `Feld(Feld &&o);`

(iv) Zuweisung mit Copy–Semantik: `Feld & operator=(const Feld &o);`

(v) Zuweisung mit Move–Semantik: `Feld & operator=(Feld &&o);`

c) Feldzugriffs–Operatorfunktion**en** `[]` mit `int`–Argument, bei negativem oder zu großem Index ist eine entsprechende Exception auszulösen! (Exception–Schnittstelle angeben!)

Definieren Sie hierzu eine entsprechende Hierarchie von Fehlerklassen:



und lösen Sie in den Feldzugriffs–Operatorfunktionen entsprechende Exceptions aus!

(Bitte beachten Sie die Zusatzfrage zu dieser Aufgabe auf Seite 12!)

d) Zusatzfrage: Welchen Unterschied in der Fehlerbehandlung haben folgende beiden Ansätze:

(i) globale Fehlerklassen:

```
...
class Feld_Fehler {};
class Feld_Unterlauf: public Feld_Fehler {};
class Feld_Ueberlauf: public Feld_Fehler {};

template <typename T>
class Feld
{ ... }
...
```

(ii) lokale Fehlerklassen:

```
...
template <typename T>
class Feld
{
  public:
    class Feld_Fehler {};
    class Feld_Unterlauf: public Feld_Fehler {};
    class Feld_Ueberlauf: public Feld_Fehler {};
  ...
}
...
```