Name:

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

Matrikelnummer:	*Punktzahl:*
Ergebnis:	

Freiversuch ☐ *F1* ☐ *F2* ☐ *F3* ☐

Klausur im WS 2005/06 :

Informatik III

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

Die Lösungen können in einigen Fällen 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 sehr häufig 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

Punktziel	Sonderpunkte	erreicht
Hausaufgaben: 10 P.		
A1: 27 P.		
A2: 15 P.		
A3: 32 P.		
A4: 12 P.		
A5: 14 P.		
Summe 100 P.		

Aufgabe 1 : Softwareentwicklung / Testen

ca. 27 Punkte

Implementieren Sie eine Funktion *Berechne* mit den beiden Eingangsparametern `a` und `e`. Die Funktion soll die folgenden Werte berechnen und **alle** diese Werte zurückliefern, d.h. nicht auf den Bildschirm ausgeben, sondern an die aufrufende Funktion zurückliefern:

- `a` plus `e`,
- `a` mal `e`,
- Die Summe der vierten Potenzen von `a` bis `e`, d.h. $\sum_{j=a}^{e} j^4$

Gehen Sie bei der Lösung der Aufgabe in den Schritten vor, wie sie in der Vorlesung vorgestellt wurden. Sie dürfen sich hierbei auf die folgenden Schritte beschränken:

a.) Definieren Sie drei wirklich **verschiedene** Tests für die Aufgabenstellung, d.h. für den Test der Funktion (Es geht hier um die Beschreibung/Auflistung von konkreten Ein- und Ausgaben der Tests – noch nicht um deren Implementierung in C++).

b.) Implementieren Sie **einen** der zuvor definierten Tests *Test1*, *Test2* und *Test3* in C++.
Anmerkung: Auf die Implementierung der Datei *Test.h* dürfen Sie verzichten.

c.) Beschreiben Sie mit Worten, was die Funktion *Berechne* macht, d.h., was die Eingangs- und Ausgangsvariablen sind und **wie** (im Aufgabenteil e.)) die Funktion aus den Eingangsparametern die Ausgangsparameter ermittelt.

d.) Geben Sie die Headerdatei *Berechne.h* mit Include-Wächter an, d.h. u.a. den Prototyp der Funktion *Berechne*.

e.) Implementieren Sie **nun erst** die C++-Funktion *Berechne* selbst.

Beachten Sie, dass die Implementierung von *Berechne* im Teil e.) nur ein kleiner Teil der Lösung ist.

Die zugehörige Hauptfunktion *main*, die die Tests ausführt und damit indirekt *Berechne* aufruft, ist bereits vorgegeben.

Listing 1: (./Code1/main.cpp)

```
#include <iostream>
using namespace std;
#include <cstdlib>// wegen EXIT_SUCCESS
#include "Test.h"

int main()
{
   if (Test1()  &&
         Test2() &&
         Test3() ){
      cout << "Alle Tests erfolgreich\n";
      return EXIT_SUCCESS;
      }
   else {
      cout << "Tests gescheitert\n";
      return EXIT_FAILURE;
   }
}
```

(*———————————————————————*)
Notieren Sie die Lösung auf einem Extra-Blatt.
(*———————————————————————*)

Falls die Aufgabenstellung Ihrer Meinung nach mehrere Möglichkeiten zulässt, wählen Sie eine aus und machen Sie Ihre Entscheidungen explizit (niederschreiben). Dies gehört zum Thema der Beseitigung von Missverständnissen und Unklarheiten.

Aufgabe 2 : Textfragen agile Software-Entwicklung

ca. 15 Punkte

a.) Nennen Sie einen Unterschied zwischen den beiden Prozessmodellen *eXtreme Programming* und dem *Wasserfallmodell* (ein Unterschied reicht).

b.) In der Softwareentwicklungsfirma *Müller & Clever* gibt es häufig Wechsel der Mitarbeiter. Durch welche Basistechnik(-en) von eXtreme Programming (XP) kann verhindert/abgemildert werden, dass dies zum Scheitern eines Softwareprojektes in der Firma *Müller & Clever* führt?

c.) Sie gucken sich gerade fertigen und korrekt ablaufenden Code an. Trotz des korrekten Codes entscheiden Sie sich sinnvollerweise für ein Code-Refactoring. Was könnten Gründe hierfür sein (3 Aufzählungen reichen)?

d.) Bei der Planung von Software-Projekten spielen neben `den Kosten/Ressourcen` drei weitere Variablen eine Rolle. Welche?

e.) Was besagt `Brooks Gesetz`? Erklären Sie es, d.h. warum ist es gültig?

Aufgabe 3 : Konstruktoren und Destruktoren, Polymorphie

ca. 32 Punkte
Gegeben sind die Klasse `Person`

Listing 2: (./Code1/Poly/Person.h)

```
#include <iostream>
#include <string>
using namespace std;

class Person {
public:
   Person(string n);
   virtual ~Person();
   void Setze(string n);
   virtual void Print() const;
private:
   string name;
};

ostream& operator<<(ostream& s, const Person& p);
```

Listing 3: (./Code1/Poly/Person.cpp)

```
#include "Person.h"

#include <fstream>
using namespace std;
extern ofstream datei;

Person::Person(string n): name(n){
   datei << "+P␣" << name << endl;
}

Person::~Person() {
   datei << "-P␣" << name << endl;
}
void Person::Setze(string n) {
   name=n;
}
void Person::Print() const{
   datei << name;
}

ostream& operator<<(ostream& s, const Person& p) {
   p.Print();
   return s;
}
```

sowie die Klasse `Mann`:

Listing 4: (./Code1/Poly/Mann.h)

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

#include "Person.h"

class Mann : public Person{
   int age;
public:
   Mann(string n, int w);
   virtual ~Mann();
   Mann(const Mann& m2);
   void Setze(string w);
   virtual void Print() const;
};

ostream& operator<<(ostream& s, const Mann& m);
```

Listing 5: (./Code1/Poly/Mann.cpp)

```
#include "Mann.h"

#include <fstream>
using namespace std;
extern ofstream datei;

Mann::Mann(string n, int a): Person(n){
   age = a;
   datei << "+M␣" << a << endl;
}

Mann::Mann(const Mann& m2): Person("Kai"){
   age = m2.age;
   datei << "+MCopy␣" << age << endl;
}


Mann::~Mann() {
   datei << "-M␣"  << age << endl;
}

void Mann::Print() const{
   Person::Print();
   datei << ":␣" << age;
}

void Mann::Setze(string a) {
   datei << "Geht␣nicht\n";
}

ostream& operator<<(ostream& s, const Mann& m) {
   m.Print();
   return s;
}
```

Im Folgenden können Sie Ihre Ausgaben direkt unter den Funktionen oder auf einem Extra-Blatt notieren.

a.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk1`?

```
void funk1(){
   datei << "funk1" << endl;
   Person p("Paul");
   Mann   m("Paula", 4);
}
```

b.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk2`?

```
void funk2(){
   datei << "funk2" << endl;
   Person* p1 = new Mann("Fritz", 4);
   Mann* p2 = NULL;
   datei << "Ende\n" << endl;
   p1 = p2;
}
```

c.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk3`? Erklären Sie Ihre Lösung durch Zeichnungen, aus denen die Speicherbelegungen auf dem Heap und auf dem Stack zu den Zeitpunkten /*1*/, /*2*/ und /*3*/ hervorgehen.

```
void funk3(){
   datei << "funk3" << endl;
   Person p("Hans");
   Mann m("Paul", 4);                    /*1*/
   p = m;                                /*2*/
   datei << m << ",␣" << p << endl;
   p.Setze("Fritz");                     /*3*/
   datei << m << ",␣" << p << endl;
}
```

d.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk5`?

```
void funk5(){
   datei << "funk5" << endl;
   auto_ptr<Mann> p(
      new Mann("Martin", 2));
}
```

e.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk6`?

```
void funkWert(Mann m) {
   datei << m << endl;
   m.Setze("Fritz");
}
void funk6(){
   datei << "funk6" << endl;
   Mann* m = new Mann("Karl", 8);
   funkWert(*m);
}
```

f.) Welche Ausgabe erzeugt der Aufruf von der Funktion `funk7`? Erklären Sie Ihre Lösung durch Zeichnungen, aus denen die Speicherbelegungen auf dem Heap und auf dem Stack zu den Zeitpunkten /*1*/, /*2a*/ und /*2b*/ (zweiter Aufruf) hervorgehen.

```
void funkRef(Person& pers) {
   pers.Setze("Fritz");
   datei << pers << endl;   /*2a, 2b */
}
void funk7(){
   datei << "funk7" << endl;
   Mann*   m1 = new Mann("Karl", 8);
   Person* m2 = new Mann("Hans", 3);  /*1*/
   funkRef(*m1);
   funkRef(*m2);
}
```

Aufgabe 4 : Fehler und schlecher Programmierstil

ca. 12 Punkte
Was ist im folgenden Codefragment sehr wahrscheinlich falsch?

```
bool Test1() {
   int erwartet[]={1, 2, 3, -1};
   int berechnet[4];
   /* . . . */
   if (erwartet == berechnet) {
     return true;
   }
   else {
     return false;
   }
}
```

Warum führt der Aufruf von *Testmatrix* im folgenden Programmausschnitt sehr wahrscheinlich zu einem Programmabsturz? Korrigieren Sie das Problem entsprechend.

```
class Matrix {
public:
   Matrix(int i)     {  /* mache nichts */}
   ~Matrix() {delete [] feld; }
   int GetZeile()    {return z;}

private:
   int z, sp;
   double* feld;
};

void TestMatrix(){
   Matrix m(2);
}
```

Was sollte bei der Deklaration der Methode *Matrix::GetZeile* unbedingt verbessert werden?

Können von der Klasse Matrix Felder (Arrays) definiert/erzeugt werden? Begründen Sie Ihre Entscheidung.

Warum ist die folgende Funktion *TestPut* nicht so sehr für einen **automatisch** ablaufenden Test geeignet?

```
void TestPut() {
   MeineMatrix mat(5, 4);
   mat.Put(1,2, 28.4);
   cout << mat.Get(1,2);
}
```

Was sollte bei der Deklaration der Attribute der Klasse *BloedeMatrix* unbedingt verbessert werden?

```
class BloedeMatrix {
public:
   BloedeMatrix(int z, int s);
   ~BloedeMatrix();
   int z, sp;
   double* feld;
};
```

Aufgabe 5 : STL

ca. 14 Punkte

a.) Welche Ausgabe erzeugt der Aufruf von *funk1* in `datei`?

```
void Print(int i){
   datei << i << "␣";
}

void funk1(){
   list<int> li1;
   for (int i=1; i<7; ++i) {
      li1.push_back(i);
   }
   for_each(li1.begin(), li1.end(),Print);
   datei << endl;

   list<int>::iterator iter =
      find(li1.begin(), li1.end(),4);
   for_each(++iter, li1.end(),Print);
   datei << endl;

   *iter = 61;
   for_each(li1.begin(), li1.end(),Print);
   datei << endl;
}
```

b.) Welche Ausgabe erzeugt der Aufruf von *funk2* in `datei`?

```
class Zufall{
   int last;
public:
   Zufall(int w): last(w) {}
   int operator()(int x){
      --last;
      datei <<(x - last) << "␣";
      return (x - last);
   }
};

void funk2(){
   list<int> li1;
   for (int i=1; i<4; ++i) {
      li1.push_back(i);
   }
   for_each(li1.begin(), li1.end(),Zufall(3));
   datei << endl;

   Zufall z(4);
   for_each(li1.begin(), li1.end(), z );
   datei << endl;
}
```

c.) Implementieren Sie für die Klasse Zufall einen sinnvollen Vergleichsoperator und einen Kopierkonstruktor in der Datei *Zufall.h*, d.h. die beiden Methoden/Funktionen sollten `inline` implementiert werden (sinnvollen Gebrauch des Schlüsselwortes `const` nicht vergessen!).