Klausur Objektorientierte Programmierung WS 2007/2008

Aufgabe 1: (18 Punkte)

Das folgende Quellfile wird gemäß C++-Sprachstandard compiliert und enthält 9 Fehler. Markieren Sie die fehlerhaften Stellen möglichst zeichengenau und schreiben Sie rechts daneben eine kurze stichwortartige Begründung!

Logische oder Laufzeit-Fehler (z.B fehlende Variablen-Initialisierungen, nicht freigegebener Speicher) sind nicht gefragt.

```
#include <string>
#using namespace std;
        class A{
                int s;
                public:
                virtual int x;
        A(string s) {s = string(“abc”) +4;}
        };

        class B :: public A{
                int i;
                public:
                A f(){
                        i=0;
                        return this;
                }
        B(string s){}
        };
void g(){
        B n[3];
        B* b = new A(“abc”);
        A m = n[0];
        m.f();
}
```

Aufgabe 2 (17 Punkte)

Es liegt das folgende fehlerfreie Programm vor (Headerdateien nicht aufgeführt):

```
class Haes{
            public:
              int m;                                  // zur Vereinfachung public
              Haes(){m = 4;}
              int f(int n)                            //Zeile 5
             {
                        n =5;
                        return 6;
              }
              int f(){return m;}
  };

class Plaetzler : public Haes{
  public:
    Plaetzler() {m = 5;}
    int f (int n)                                     //Zeile 16
         {
          n =7;
          return 8;
          }
   };
int main (){
     int i = 1;
     Haes* t = new Plaetzler;
     cout << t->f(i);                                 //Zeile 26
     cout << t->f();                                  //Zeile 27
     cout << i;                                       //Zeile 28
 }
```

Tragen Sie in die folgende Tabelle ein, welche Ausgabe das Programm in Zeilen 26 bis 28 erzeugt, und zwar in der Orginalversion, sowie wenn Zeile 5 und 16 wie angegeben ersetzt werden!

Deklaration in Zeile 5 und 16	Zeile 26:	Zeile 27:	Zeile 28:
Int f(int n) (Orginalversion)			
Int f(int& n)			
Virtual int f(int n)			
Virtual int f(int& n)			

Aufgabe 3 (20 Punkte)

Eine Klasse für die Kalendertage eines Jahres ist zu erstellen. Bevor Sie mit der Lösung beginnen, sollten Sie diese und die folgende Aufgabe durchlesen, um einen Überblick über die geforderten Funktionen zu bekommen. Die für jeden Aufgabenteil nötige Code ist in den zugehörigen Rahmen einzutragen, muss den Prinzibien der Objektorientierten Porgrammierung genügen und ist so zu verfassen, dass er ab Zeile 6 in die Klasse eingefügt werden kann. Zahlenwerte und Variablennamen in den Beispielen sind natürlich nur exemplarisch zu verstehen!
Die Feinheiten des gregorianischen Kalenders wollen wir für die Aufgabe vereinfachen: Alle 12 Monate haben genau 30 Tage.

Folgende Definitionen liegen bereits vor:

```
const int current_month = 2; //aktueller Monat, derzeit Februar
class Date {
int day;                    //Wertebereich 1..30
int month;                  //Januar = 1, Februar = 2 etc
public:
...                         //Zeile 6: hier Methoden 1t. Aufgaben
 };
```

a)
Ein Datumswert soll sich erzeugen lassen unter Angabe von Tag und Monat! Der Monat kann auch weggelassen werden. In diesem Fall wird der Monat aus der Konstanten *current_month* verwendet.

Beispiel:
```
Date ostern(23,3);    //23. März
Date heute(12);       //12. Februar
```

Lösung zu a)

Prinzipiell gibt es zwei verschiedene Möglichkeiten, um ein Datum wie gefordert erzeugen zu können. Beschreiben Sie kurz, welche Alternative es zu Ihrer Lösung gegeben hätte!

b)
Ein Datum soll sich mit einer Methode get auf der Konsole ausgeben lassen. Dabei werden für Tag und Monat jeweils zwei Zeichen verwendet.

Beispiel: *ostern.get(); zeigt an „23. 3.“*

c)
Beim Erzeugen eines Datums mit Tag außerhalb des Bereichs von 1 bis 30 oder mit einem unzulässigen Monat soll eine Exception vom Typ '*IllegalDate*' geworfen werden. Dabei erhält die Exception keine Parameter. Geben Sie den dazu nötigen Code an und machen Sie deutlich, wo er zu platzieren ist!
Eine Exception-Handler brauchen Sie nicht zu implementieren!

d)

Definieren Sie jetzt die erforderliche Exceptionklasse! Sie besitzt insbesondere eine parameterlose Methode '*get_error'* ohne Rückgabewert, deren Aufruf an der Konsole einen Text der folgenden Art ausgibt:

Datumsfehler Nr. 46

Die Nummer ist dabei die Zahl der Fehler dieses Typs, die seit Programmstart aufgetreten sind. Sie ist z.B 0 bei Aufrufen der Methode zwischen Programmstart und dem ersten Auslösen der Exception. Machen Sie auch hier deutlich, wo der nötige Code relativ zur Klassendefinition zu plazieren ist!

Aufgabe 4 (15 Punkte)

Diese Aufgabe verwendet die Klasse *Date* aus Aufgabe 3, ist aber unabhängig von deren Bearbeitung lösbar.
In den folgenden Beispielen sind heute (12.2), ostern (23.3) und silvester (hier 30.12 !) Instanzen der Klasse *Date.*

a) Eine Methode *daynr* der Klasse Date soll die fortlaufende Nummer des Tages seit Jahresanfang liefern, beginnend mit 0 für den 1. Januar.

Beispiel : *silvester.daynr()* ergibt den Wert 359

b) Zwei Datumswerte sollen sich mit dem Operator '<' vergleichen lassen. Der Vergleich liefert genau dann *true*, wenn das linke Datum *früher* ist als das rechte, sonst *false*.
Welchen Code müssen Sie dazu in die Klasse einfügen ? Nutzen Sie die Infrastruktur aus Teil a) dieser Aufgabe !

Beispiel: *ostern* < *heute* ist *false*

c) Es soll ein Array von 3 Datumswerten in einer einzigen Anweisung definiert und mit den bereits zuvor erzeugten drei Datumsinstanzen *silvester, heute* und *ostern* belegt werden, genau in dieser Reihenfolge. Wie lautet diese Anweisung?

d) Das Array aus Teil c) soll in aufsteigender zeitlicher Reihenfolge sortiert werden (Ergebnis also *heute, ostern, silvester*). Wie lautet der Code dafür unter Verwendung der Möglichkeit der Standard Template Library? Selbstverständlich muss Ihr Code so angelegt sein, dass er für ein beliebig besetztes Array mit drei Elementen funktioniert.
Geben Sie auch kurz an, wodurch die gewünschte Sortierreihenfolge zustande kommt!
Was müsste man für eine Sortierung in absteigender Reihenfolge ändern?

Code zur Sortierung:

Wodurch wird die Sortierreihenfolge bestimmt?

Welche Änderung wäre für absteigende Sortierung nötig? (genaue Angabe!)

Aufgabe 5 (18 Punkte)

Beantworten Sie noch folgende Kurzfragen :

a) Wie bezeichnet man die Möglichkeit, Objekte durch einen geeigneten Mechanismus dauerhaft auf einem Datenträger speichern (und wieder abrufen) zu können? (Stichwort)

b) Kreuzen Sie an, welche der angegebenen Funktionen in demselben Gültigkeitsbereich mit der Definition *float p (char* s)* koexistieren können (jede für sich alleine)!

- ☐ float p (char * txt)
- ☐ float p (int i)
- ☐ void p (char * s)

Wie wird dieser Sachverhalt bezeichnet ?

c) Wieviele verschiedene Destruktoren kann eine Klasse besitzen? (Bitte ankreuzen)

- ☐ Einen
- ☐ Genauso viele wie Konstruktoren
- ☐ Beliebig viele

d) Eine Containerklasse *List* wurde wie in der Vorlesung als Template definiert und mit *List*< *X** > *i*nstantiiert, wobei *X* eine Klasse ist. Welche Datentypen kann man dann beim Aufruf der List-Methode add für deren Argument verwenden, ohne dass der Compiler dies beanstandet?

Hinweis: Es wird eine genaue Beschreibung dieser Klassen erwartet. Eine vollständige Antwort ist nicht ganz einfach, aber auch eine teilweise Beantwortung gibt Punkte.

e) Gegeben ist folgende kleine Templatefunktion:

```
template <typename T> void f (T&x, int i) {
        x << i;
}
```

Geben Sie in der rechten Spalte kurz an, ob die Aufrufe links zulässig sind und – wenn ja – was sie bewirken!

int i; .....; f(i, 1);	
double x; ..... ; f(x,1);	
f(cout, 1);	