OOPR SS08

Aufgabe 1 (18 Pkt)

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) sind nicht gefragt.

```
#include <string>
class A {
  boolean flag;            // Datentyp heißt nur bool
public:
  virtual int f() {
    std::string s = "Aufgabe 1";// Zuweisungsoperator fehlte
    return s.size();
  }
  A(int i) {}
  char g(int i = 0, char c) {   // Default-Parameter nur am Ende
    return c;
  }
};
class B : public A {
  double x;
public:
  void B() {          // Konstruktor hat keinen Rückgabewert
  }                   // 2. Fehler hier: A hat keinen passenden
                      // Standardkonstruktor A()
  int h() {return 0;}
  double f() {  // falscher Rückgabetyp der überladenen Methode f
    B b;
    b = A(1);   // Klasse A ist nicht vom Typ Klasse B
    b->h();     // Dereferenzierung falsch, Punkt korrekt
    b::g(0, '.'); // Scope-Operator falsch, Punkt korrekt
    return b.x;
  }
};
```

Aufgabe 2 (32 Pkt)

Im Rahmen einer Anwendung zur Ticketreservierung des EM-Stadions in Wien ist eine Klasse zu erstellen, welche die Zuordnung zwischen Nummer der Eintrittskarte und Name des Zuschauers herstellt.

Der für jeden Aufgabenteil nötige Code ist in den dafür vorgesehenen Rahmen einzutragen und muss die Möglichkeiten von C++ nutzen.

Die angegebenen Beispiele müssen sich mit ihrem Code fehlerfrei kompilieren lassen und sie müssen die geforderten Ausgaben erzeugen.

Folgende Definition liegt bereits vor:

```
class Ticket {
    int ticketnr;     // Nummer des Eintrittstickets
    string name;      // Name des Zuschauers
  public:
    …                 // ab hier Ergänzungen lt. Aufgaben
};
```

a) Ein Ticket-Objekt soll sich erzeugen lassen unter Angabe von Ticketnummer und Name des Zuschauers! Beide zusammen (nicht einzeln!) können auch weggelassen werden. In diesem Fall ist dann die Ticketnummer 0 und der Name „NN“.

Beispiele:

```
Ticket vip(1435, "Beckenbauer");
Ticket frei;
```

Lösung:

```
Ticket() {
  ticketnr = 0;
  name = "NN";
}

Ticket(int _ticketnr, string _name) {
  ticketnr = _ticketnr;
  name = _name;
}
```

b) Die Daten jedes Tickets sollen sich mit einer parameterlosen Methode `list` auf die Konsole ausgeben lassen!

Beispiel:

```
vip.list(); frei.list();
```

geben die folgenden zwei Zeilen aus

```
Ticket  1435:  Beckenbauer
Ticket     0:  NN
```

Lösung:

```
void list() {
  cout << "Ticket ";
  cout << setw(5) << setfill(' ') << right << ticketnr;
  cout << ":  " << name << endl;
}
```

c) Um die Daten zu allen ausgegebenen Karten speichern zu können, wird im Hauptprogramm für die Tickets ein Array der Größe 10.000 angelegt.
Definieren Sie ein solches Array mit dem Namen `wien` und speichern Sie in den ersten beiden Arrayelementen die Instanz `vip` aus Teil a, sowie ein Ticket mit Nummer 2100 und Namen „Maier“, wobei für dieses Ticket keine Variable definiert werden soll.
Wie wird danach die Methode `list` aus Teil b für das allerletzte Arrayelement aufgerufen? Was wird ausgegeben?

Lösung (Code im Hauptprogramm):

```
Ticket wien[10000];
wien[0] = vip;
wien[1] = Ticket(2100, "Maier");
wien[9999].list();
```

Ausgabe: `Ticket  0:  NN`

d) Das gesamte Array `wien` aus Teil c soll jetzt im Hauptprogramm alphabetisch nach den Namen der Zuschauer sortiert werden. Die Sortierung soll mit beliebigem Arrayinhalt funktionieren. Wie lautet der Code dafür im Hauptprogramm unter der Verwendung der C++-Standardbibliothek? Geben Sie auch weiteren Code an, der zu ihrer Lösung zusätzlich erforderlich ist und beschreiben Sie, wo er einzufügen ist!

Lösung:

```
// Im Dateikopf
#include <algorithm>

// in Klasse Ticket
friend bool isbefore(Ticket a, Ticket b);

// nach Klassendefinition von Ticket (vor main() )
bool isbefore(Ticket a, Ticket b) {   // ist a vor b?
  return (a.name < b.name);
}

// in main()
sort(wien, wien+10000, isbefore);
```

e) Im Stadion von Basel sei die Ticketnummer nicht ganzzahlig, sondern besteht aus zwei Zahlen, die durch einen Bindestrich getrennt sind (z.B. 1234-533). Nehmen Sie an, dass bereits eine Klasse `Tickettyp` definiert worden ist, die beide Bestandteile getrennt speichert. Diese Klasse definiert genau einen Konstruktor, der mit linker und rechter Teilnummer aufgerufen wird:

```
Tickettyp::Tickettyp(int leftnr, int rightnr);
```

Wie ist der zu Beginn der Aufgabe vorgegebene Definitionsrahmen der Klasse `Ticket` abzuändern, damit er sowohl in Wien als auch in Basel verwendet werden kann?
Wie wird dann im Hauptprogramm mit dieser geänderten Definition das Array für die Wiener Tickets und zusätzlich ein zweites Array für die Basler Tickets definiert?
Geben Sie noch in Stichworten an, welche Maßnahmen außerdem nötig sind, damit ihre Lösungen zu Teil a und b auch mit der neuen Version der Klasse `Ticket` funktionieren! Welche Methoden muss die Klasse `Tickettyp` besitzen (nur Prototyp)?
Lösung:

```
// Klasse Ticket
template<typedef T> Ticket {
  T ticketnr;
public:
  Ticket(T _ticketnr, string _name) {
    // Inhalt unverändert
  }
  …
}

// Wie werden die beiden Arrays definiert?
Ticket<int> wien[10000];
Ticket<Tickettyp> Basel[10000];

// Was ist nötig, damit der Code aus den Teilen a und noch
// lauffähig  bleibt?
// * Tickettyp braucht einen Typumwandlungskonstruktor für int:
//   Tickettyp::Tickettyp(int i)
//   (nötig für die Zeile „ticketnr = 0“)
// * es wird eine globale Funktion
//   ostream& operator<< (ostream& os, Tickettyp t)
//   benötigt, um Tickettyp mit cout nutzen zu können
// * ideal wäre eine print()-Funktion, welche die Nummer
//   formatiert ausgibt (und von der Operator<<-Funktion
//   genutzt werden kann)
```

Aufgabe 3 (30 Pkt)

Es liegen zwei (fehlerfreie) Klassendefinitionen vor. Um die Aufgabe zu vereinfachen sind alle Elemente öffentlich.

```
class Koerper {
public:
  static int el;                  // Teil c)
  Koerper() { el = 5; cout << el; }
  void f(int i) {i = i + el; el = i; cout << getel(); }  // Z5
  int getel() {return el;}                               // Z6
}

int Koerper::el = 0;              // Teil c), statische Variable
                                  // muss initalisiert werden

class Kugel : public Koerper {
public:
  int getel() {return (el-4);}
};
```

a) Gehen Sie das folgende Hauptprogramm Zeile für Zeile durch, und überlegen Sie sich was das Hauptprogramm auf dem Monitor ausgibt!
Wenn von einer Programmzeile eine Ausgabe erzeugt wird, tragen sie den Ausgabetext auf gleicher Höhe in die linke Tabelle ein, sonst lassen sie die Tabelle dort leer.

```
int main() {
  Koerper k;
  Koerper* p;
  int i = 10;
  k.f(i);
  p = new Koerper;
  k.f(i);
  cout << i;
  cout << p->el;
}
```

Teil a	Teil b &	Teil c static
5	5	5
15	15	15
5	5	5
25	30	15
10	30	10
5	5	15

b) In Zeile 5 soll `int` durch `int&` ersetzt werden. Welche Ausgaben macht nun dasselbe Hauptprogramm? Tragen Sie die Texte jeweils in die mittlere Tabelle zeilengenau ein!

c) Ausgehend von den Originalklassen (d.h. ohne Änderung aus Teil b) soll die Variable `el` statisch werden. Tragen Sie zunächst die dazu nötigen Änderungen in den Code ein. Schreiben Sie danach in die rechte Tabelle wieder alle Ausgaben, die das Hauptprogramm erzeugt.

d) Jetzt ist das nachfolgende Hauptprogramm zu analysieren. Tragen Sie die Ausgaben wieder zeilengenau in die vorbereiteten Tabellen ein. In die linke Tabelle kommen die Ausgaben unter Verwendung der Originalklassen aus Teil a und in die rechte Tabelle die Ausgaben, wenn die Funktion `getel` in Zeile 6 virtuell wäre.

```
int main() {
  Koerper k1;
  Koerper* k2;
  Kugel k3;
  cout << k3.getel();
  k1 = k3;
  cout << k1.getel();
  k2 = &k3;
  cout << k2->getel();
  k2->f(0);
}
// Koeper k4 (für Teil e)
```

Originalklassen	getel virtuell
5	5
5	5
1	1
5	5
5	1
5	1

e) Gibt es eine Änderung der Ausgabe, wenn in Teil d nach dem Hauptprogramm noch eine globale Variable vom Typ `Koerper` definiert wird (siehe Kommentar)?
Wenn ja, welche?
Lösung:

Ja, die globale Variable wird noch vor der main()-Funktion initialisiert, deshalb erfolgt vorher noch die Ausgabe der Zahl 5.

Aufgabe 5

Beantworten Sie noch folgende Kurzfragen:

a) Weshalb muss der Kopierkonstruktor immer mit einem Referenzparameter definiert werden?
Lösung:
Da eine Objekt-Instanz-Parameter normalerweise durch call-by-value kopiert wird, würde dies zu einer endlosen Rekursion des Kopierkonstruktors führen, da zum Kopieren des Objekts ja eben genau dieser benötigt wird.

b) Die Klasse `string` der C++-Standardbilbiothek unterstützt den Operator ‚*', um eine Zeichenfolge an den bestehenden String anzufügen. Wie wir in Übung 4 gesehen haben, lassen sich mit diesem Operator aber keine Zahlenwerte anhängen.
Wie kann man erreichen, dass der Operator ‚*' als rechten Operanden auch Zahlen vom Typ double zulässt, sodass beispielsweise Folgendes möglich wird:

```
string s = "Note: ";
s = s * 1.7;
cout << s; // Ausgabe: ‚Note 1.7'
```

Lösung:

```
string operator*(string os, double d)
{
      ostringstream oss;
      oss << os << d;
      return iss.str();
}
```

c) Das folgende Hauptprogramm benutzt offensichtlich eine Klasse A:

```
int main() {
  A a;
  double x = a.get();
  a = -a;
  x = A::f();
  A* b = new A(1,-1);
}
```

```
class A {
public:
  A();
  A(int, int);
  static double f();
  A operator-();
  double get();
```

Vervollständigen Sie in dem Kasten die notwendige Definition der Klasse, damit das Hauptprogramm korrekt ist! Die Methoden müssen nur mit dem Prototyp angegeben werden.

d) Welche der folgenden Aussagen ist richtig? (Zutreffendes ankreuzen)
Eine Basisklasse definiert einen Konstruktor mit einem Parameter. Jede von ihr abgeleitete Klasse

- o muss dann zwingend einen Konstruktor mit genau einem Parameter definieren
- ✓ muss dann zwingend einen Konstruktor definieren, dessen Parameteranzahl aber beliebig ist
- o kann einen Konstruktor definieren, muss es aber nicht

e) Kreuzen Sie im Folgenden an, was die angegebenen Anweisungen bedeuten!
Hier ist i eine Variable und T ein Datentyp. Wofür a bis d stehen, müssen Sie selbst herausfinden.

	Funktionsaufruf	Funktionsdeklaration (Prototyp)	Variablen-deklaration
`T a(int i);`		x	
`T b(i);`			x
`T c();`		x	
`d();`	x		