Abschlußklausur

Objektorientierte Programmierung in C++

Bitte notieren Sie auf allen Lösungsblättern Ihren Namen und einmalig Ihre Einschreibnummer. Anhand der letzteren werden Anfang August die Ergebnisse der Klausur durch Aushang (Axel-Springer-Str. 54a, 1. Etage rechts, Schaukasten) öffentlich bekanngegeben. Alle dort entsprechend markierten Studenten sind aufgefordert, sich für eine Prüfungszulassung zu zusätzlichen Konsultationen anzumelden (Dr. Ahrens, Tel. 20 181 238)

1. Implementieren Sie einen Datentyp ‚`String`‘ zur effizienten Repräsentation von Zeichenketten mit folgenden Eigenschaften:

   - beim Kopieren/Zuweisen von `String`-Objekten ist statt einer Kopie der eigentlichen Zeichenkette (`char*`) ein Referenzzähler zu erhöhen, gleiche `String`-Objekte „teilen“ sich eine Zeichenkette,
   - falls ein `String` als C-Zeichenkette (`char*`) verwendet wird, soll eine automatische Typumwandlung vorgenommen werden
   - die Ausgabe von `String`-Objekten über beliebigen `ostream`‘s soll möglich sein

   Hinweis: Durch Einschluß des Systemheaderfiles `string.h` kann man die Funktionen `strlen(char* str)` [Länge der Zeichenkette ohne ‘\0‘] und `strcpy (char* source, char* dest)` [Kopie incl. ‘\0‘] verwenden. Man verwende eine separate Klasse für die eigentliche Repräsentation der C-Zeichenkette und führe einen Referenzzähler, der beim Kopieren, Zuweisen und Löschen von `String`-Objekten aktualisiert wird:

   ```
   class String {
   private:
        class StringRep {
             char *rep;
             int ref;
             ...} *rep;
        ...
   };
   ```

   [10 Punkte]

2. Welche Ausschriften erzeugt das folgende C++-Programm ?

   ```
   #include <iostream.h>

   class A {
   public:
        A() {cout << "A";}
        ~A() {cout << "~A";}
   };

   class B: public virtual A {
   public:
        B() {cout << "B";}
        ~B() {cout << "~B";}
   };
   ```

```
class C: public virtual B {
public:
     C() {cout << "C";}
     ~C() {cout << "~C";}
};
class D: public C, public virtual B, public virtual A {
     B b;
public:
     D(){ cout << "D";}
     ~D() {cout << "~D";}
};

A a;
B * pb = new B;

main() {
     cout << "{";
     C c;
     D d;
     cout << "}";
}
```

Warum sollte man `cout` i.allg. nicht innerhalb von Konstruktoren verwenden ?

[8 Punkte]

3. Die folgende Implementationen der Funktion `middle` ist korrekt. Erstellen Sie mit den Mitteln von C++ eine möglichst (laufzeit-)effiziente ‚Reimplementation'. Begründen Sie die Vorteile Ihrer Implementation.

```
class X {
     char x [2000];
public:int i;
     X(int n): i(n){}
};

int middle (X a, X b) {
     return (a.i+b.i)/2 ;
}
```

[6 Punkte]

4. Implementieren Sie einen abstrakten Datentyp Stack von `int`-Werten, dessen Kapazität (potentiell) unbeschränkt ist. Organisieren Sie dazu die einzelnen Einträge in Form einer dynamisch verketteten Liste.

```
class Stack {  // of int
     ....
public:
     Stack();           // leerer Stack
     void push(int i);  // i ‚einkellern'
     int pop();         // oberstes Kellerelement ‚auskellern'
};
```

[8 Punkte]

5. Implementieren Sie zwei Klassen `class Celsius` und `class Fahrenheit`, die zur (dimensionierten) Erfassung von Temperaturen dienen sollen. Objekte der beiden Klassen sollen bei Bedarf automatisch geeignet ineinander ‚umgerechnet' werden, z.B.

```
Celsius c = 20;    // 20ºC
Fahrenheit f = c;  // die gleiche Temperatur, aber in ºF (68ºF)
Celsius c1 = f;    // wieder 20ºC
```

Die Umrechnungsformel lautet: $\text{temp}\,[^{o}\text{C}] \;=\; (\text{temp}\,[^{o}\text{F}] - 32) \cdot 5/9$

Bei der Augabe von Objekten beider Klassen soll die Dimension mit ausgegeben werden (‘º‘ sei ein druckbares Zeichen):

```
cout << "Temperatur = " << c << endl;
// ---> Temperatur = 20ºC
```

[8 Punkte]

6. Entwerfen Sie ein C++-Sprachmuster (*idiom*) nach dem von einer Klasse nur ein einziges Objekt existieren kann.

Entwerfen Sie ein C++-Sprachmuster (*idiom*), nach dem dynamisch aus einem polymorph referenzierten Objekt eine exakte Kopie erstellt wird („virtueller Konstruktor“) :

```
class A { ... } a, *pa;
class B: public A { ... } b;
class C: public A { ... } c;
class D: public B { ... } d;

pa=&a; pa=pa->dup(); // pa zeigt auf ein neues A als Kopie von a
pa=&b; pa=pa->dup(); // pa zeigt auf ein neues B als Kopie von b
pa=&c; pa=pa->dup(); // pa zeigt auf ein neues C als Kopie von c
pa=&d; pa=pa->dup(); // pa zeigt auf ein neues D als Kopie von d
```

[8 Punkte]

7. Welche Ausschriften erzeugt das folgende C++ -Programm ?

```
#include <iostream.h>

class A {
public:
     virtual void f() { cout << "A::f()\n"; }
     void g() { cout << "A::g()\n"; f(); }
     A () { f(); }
};

class B : public  A {
public:
     void f() { cout << "B::f()\n"; }
     void g() { cout << "B::g()\n"; f(); }
     B() { f(); }
};
```

```
class C : public B {
public:
     virtual void g() { cout << "C::g()\n"; f(); }
};

main() {
     A *p;
     p = new A; p->f(); p->g();
     p = new B; p->f(); p->g();
     p = new C; p->f(); p->g();
}
```

[8 Punkte]

8. Finden Sie möglichst viele Fehler in dem folgenden vermeintlichen C++-Programm. Versuchen Sie dazu eine korrekte Implementation zu erstellen. Klassifizieren Sie die gefundenen Fehler nach
   (1) Syntaxfehler (d.h. Verletzungen der Syntaxregeln der Sprache)
   (2) Semantikfehler (Typ-, Schutzverletzungen, fehlende Deklarationen/Definitionen)
   (3) logische Fehler (Fehler, die [nach geeigneten Korrekturen von (1) und (2)] zur Laufzeit dazu führen, daß nicht das angegebene Programmverhalten zustandekommt)
   (4) memory leaks (Fehler, die [nach geeigneten Korrekturen von (1), (2) und (3)]trotz korrektem Programmverhalten zu Problemen führen können, weil dynamischer Speicher „verschwendet“ wird)

   Hinweis: Die Funktion ‚`sqrt`‘ ist in übersetzter Form in der C-Standardbibliotheken verfügbar, sie verlangt ein Argument des Typs ‚`double`‘ und liefert ein ‚`double`‘-Resultat.
   Nicht alles, was auf den ersten Blick als Fehler erscheint, ist auch einer!

```
/*
  Das folgende Programm soll die Nullstellen von quadratischen Gleichungen
  bestimmen, deren Koeffizienten interaktiv erfragt werden; die Eingabe von
  Null (oder einer Zeichenfolge, die keine reelle Zahl darstellt) fuer den
  Koeffizienten a soll das Programm beenden (bei b und c wird korrekte
  Eingabe vorausgesetzt)
*/

/* 1*/   extern double sqrt(double);
/* 2*/
/* 3*/   class P {
/* 4*/      double a, b, c;
/* 5*/   //*
/* 6*/   P(x) = a*x^2 + b*x + c
/* 7*/   */
/* 8*/
/* 9*/      P (double pa=0, double pb, double pc)::
/*10*/           a(pa), b(pb); c(pc) {}
/*11*/   public
/*12*/      int D(void) {return b*b - 4*a*c;}
/*13*/      double x1() {
/*14*/           retrun (-b+sqrt(D()) / 2*a;
/*15*/      };
/*16*/      double x2(void)
/*17*/   };
```

```
/*18*/   main(); {
/*19*/      double a=0, b=0, c=0;;
/*20*/      for (;;) {
/*21*/           cout<< "Bitte die Koeffizienten eingeben\n";
/*22*/           cout<< "a = "; cin >> a; if (a=0) break;;
/*23*/           cout<< "b = "; cin >> b;
/*24*/           cout<< "c = "; cin >>>c;
/*25*/
/*26*/           P *p = new P(a,b,c);
/*27*/           if p.D < 0 cout << "keine reellen Loesungen !\n;
/*28*/           else
/*29*/                cout << "x1 = " << p->x1() << endl;
/*30*/                cout << "x2 = " << (*p).X2(); << endl;
/*31*/   }
/*32*/
/*33*/   inline double P:x2() {
/*34*/      return (-b-sqrt(D()) / 2*a;
/*35*/   }
```

[10 Punkte]