Name: Matrikelnummer:

Bachelorprüfung Programmieren 3

SS10

Erlaubte Hilfsmittel: Stift

Lösung ist auf den Klausurbögen anzufertigen. (eventuell Rückseiten nehmen)

Bitte legen Sie den Studentenausweis auf den Tisch.

Bearbeitungszeit: 90 Minuten

Unterschrift

Benotung

Aufgabe:	1	2	3	4	5		Gesamt	Note
Punkte:	15	21	20	20	24		100	
erreicht:								

Aufgabe 1 (15 Punkte)
(C++ Syntax und Semantik)
Welche Ausgaben machen die folgenden Programme. Erklären Sie, wie es zu den Ausgaben kommt. Benutzen Sie dabei möglichst die entsprechenden Fachbegriffe.

a) Aufgabe1a.cpp

```
1  #include <iostream>
2  class O {
3    public:
4      int x;
5      O(int x):x(x){}
6      virtual void print(){ std::cout << x << std::endl;}
7  };
8
9  class U: public O{
10   public:
11     U(int x):O(x){}
12     virtual void print(){ std::cout << (x*2) << std::endl;}
13 };
14
15
16 void print(O& o){
17   o.x = o.x+4;
18   o.print();
19 }
20
21 int main(){
22   U u(17);
23   print(u);
24   std::cout << u.x << std::endl;
25   O o = u;
26   print(o);
27   std::cout << o.x << std::endl;
28   return 0;
29 }
```

b) Aufgabe1b.cpp

```
#include <iostream>
class C{
 public:
  int counter;
  C():counter(42){};

  int next(){
    counter=counter+1;
    if (counter > 43){
      counter = 0;
      throw 17;
    }
    return counter;
  }
};
int main(){
  C c;
  try { std::cout << c.next() << std::endl;
        std::cout << c.next() << std::endl;
        std::cout << c.next() << std::endl;
        std::cout << c.next() << std::endl;
  }catch (std::string s){
    std::cout << s << std::endl;
    std::cout << c.next() << std::endl;
  }catch (int i){
    if (i == 42){
      std::cout << c.next() <<std::endl;
      std::cout << c.next() << std::endl;
    }else{
      std::cout << "unknown error: "<< i << std::endl;
      std::cout << c.next() << std::endl;
    }
  }
}
```

c) Aufgabe1c.cpp

```cpp
#include <iostream>

int main(){
  int i = 17;
  int* i1 = &i;
  int& i2 = i;
  i2=(*i1)+4;
  int i3 = *i1+i2;
  *i1=2*i2;
  std::cout<<i<<std::endl;
  std::cout<<i1<<std::endl;
  std::cout<<i2<<std::endl;
  std::cout<<i3<<std::endl;
  return 0;
}

```

Aufgabe 2 (21 Punkte)
(rekursive Strukturen)
Gegeben sei die folgende generische Javaklasse für Baumstrukturen:

T.java

```
1  import java.util.*;
2  class T<A>{
3   public A element;
4   final public List<T<A>> children;
5
6   public T(A element ){
7     this(element,new ArrayList<T<A>>());
8   }
9
10  public T(A element,List<T<A>> children){
11    this.element = element;
12    this.children = children;
13  }
14 }
```

a) Schreiben Sie für die Klasse `T` eine Methode
`int highestNumberOfChildren()`
Es soll die höchste Anzahl zurückgegeben werden, die ein Baumknoten als direkte Kinderknoten hat.

b) Schreiben Sie für die Klasse `T` eine Methode
`boolen contains(A a)`
Die Methode soll genau dann wahr zurückgeben, wenn ein Element, das gleich dem Parameter ist, im Baum gespeichert ist.

c) Gegeben sei zusätzlich folgende Schnittstelle:

F.java

```
1 interface Transformer<A>{
2   void transform(A x);
3 }
```

Schreiben Sie für die Klasse `T` eine Methode
`void map(Transformer<A> f)`

Die Methode soll für alle Elemente, die in den Baumknoten gespeichert sind, die Methode `transform` des übergebenen Objekts des Typs `Transformer` anwenden.

Aufgabe 3 (20 Punkte)
(XML API)
Schreiben Sie mit dem DOM API folgende Methoden:

a) `public static int maxDepth(org.w3c.dom.Node node);`
Es soll die maximale Pfadlänge aller Pfade in dem übergebenen DOM-Baum zurückgeben werden.

b) `static void collectText(org.w3c.dom.Node node,StringBuffer result)`
Es soll mit der Methode `append` der Text aller Textknoten an den Parameter `result` angehängt werden.

Aufgabe 4 (20 Punkte)
(Standard Template Library)
Schreiben Sie im Stil der Standard Template Library folgende Funktionen, die das Iterator-Konzept benutzen sollen. Schreiben Sie jeweils einen Test, in dem Sie die Funktion aufrufen.

a)
```
template <typename Iterator, typename ElementType>
int containsHowOften(Iterator anfang, Iterator ende,ElementType element);
```

Die Funktion soll zurückgeben, wie oft das übergebene Element im Iterationsbereich enthalten ist.

b)
```
template <typename Iterator, typename ElementType>
void fuerAlle(Iterator anfang,Iterator ende,ElementType f(ElementType));
```

Die Funktion soll auf alle Elemente des Iterationsbereichs die Funktion `f` anwenden und dann die Elemente durch das Ergebnis dieser Funktionsanwendung ersetzen.

Aufgabe 5 (24 Punkte)
(Zusammenhänge)

a) Was passiert technisch, wenn ein Objekt nicht als Pointer oder Referenz als Parameter an eine Methode übergeben wird. Was folgt daraus für das late-Binding?

b) Warum hat man in C++ nicht wie in Java ein eigenes Konzept für Interfaces eingefügt.

c) Wie werden in C++ Template-Klassen vom C++-Compiler übersetzt?

d) In Java gibt es anders als in C++ keine Defaultwerte für Parameter. Wie würden Sie in Java stattdessen ausdrücken, dass standardmäßig bestimmte Parameter mit einen Defaultwert belegt sind?

e) Wenn Sie in C++ eine Methode nicht als virtual markieren, existiert für diese kein Late-Binding. Worin kann dabei der Vorteil liegen?

f) Im Gegensatz zu den Collection-Klassen in `java-util` ist die Standard-template Library nicht objektorientiert modelliert. Woran erkennt man das.