Klausuraufgaben

Aufgabe 1 Folgende C++ Programme enthalten Fehler, die vom Compiler entdeckt werden. Erklären Sie den Fehler und machen Sie einen Korrekturvorschlag.

a)

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

class Person{
 public:
   string name; string vorname;
   Person(string name,string vorname)
    :name(name),vorname(vorname){}
   string getFullName(){return vorname+" "+name;}
};

int main(){
  Person p("Andrak","Manuel");
  cout << p->getFullName();
}
```

b)

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

class Person{
 public:
   string name; string vorname;
   Person(string name,string vorname)
    :name(name),vorname(vorname){}
   string toString(){return vorname+" "+name;}
};

template <typename at>
at t(at x){cout << x.toString();return x;}

int t(int x){cout << x; return x;}

int main(){
  char* n ="Zerlett"; string v ="Helmut";
  Person p = t(Person(n,v));
  int i = t(42);
  t(new Person(n,v));
}
```

Aufgabe 2 Rechnen Sie folgendes Programm auf dem Papier und geben Sie die Ausgabe auf dem Bildschirm an. Erklären Sie kurz, wie es zu der Ausgabe kommt.

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

template <class At>
class Box{
public:
  At content;
  Box<At>(At& x):content(x){x[1]='o';};
};

int g(int& x){x=x+4; return x;}
int f(int x){return g(x);}

int main(){
  string s = "hallo";int y = 42; int z = (f(y));
  Box<string> b(s);
  cout << s << endl;
  s[1]='e';
  cout << z << endl; cout << y << endl;
  cout << b.content << endl; cout << s << endl;
}
```

Aufgabe 3 Gegeben sei folgende Headerdatei für die rekursiv definierten einfach verketteten Listen aus der Vorlesung:

```
#include <iostream>

template <typename a>
class L{
private:
  a hd;
  L<a>* tl;
  bool isEmpt;
public:
  bool isEmpty(){return isEmpt;}
  a head(){return hd;}
  L<a>* tail(){return tl;}

  L():isEmpt(true){}
  L(a x,L<a>* xs):hd(x),tl(xs),isEmpt(false){}
};
```

Schreiben Sie Implementierungen für die folgenden Methoden, die der Klasse `L` zugefügt werden sollen:

a) `L<a>* everySecond();` zurückgegeben werden soll eine neue Liste, die jedes zweite Elemente der `this`-Liste enthält, also für die Liste `[1,2,3,4,5,6,7,8]` soll die Liste `[1,3,5,7]` erzeugt werden.

b) `L<a>* take(int i);` soll die Liste der ersten `i` Elemente der `this`-Liste zurückgeben.

c) `int howOftenCotains(a x)` soll die Anzahl, die das `x` in der `this`-Liste enthalten ist, zurückgeben.

Lösung

```
#include <iostream>

template <typename a>
class L{
private:
  a hd;
  L<a>* tl;
  bool isEmpt;
public:
  bool isEmpty(){return isEmpt;}
  a head(){return hd;}
  L<a>* tail(){return tl;}

  L():isEmpt(true){}
  L(a x,L<a>* xs):hd(x),tl(xs),isEmpt(false){}

  //a)
  L<a>* everySecond(){
    if (isEmpty()) return new L<a>();
    if (tail().isEmpty()) return new L<a>(head(),new L<a>());
    return new L<a>(head(),tail()->tail()->everySecond());
  }
  //b)
  L<a>* take(int i){
    if (isEmpty()||i==0) return new L<a>();
    return new L<a>(head(),tail()->take(i-1));
  }
  //c)
  int howOftenContains(a x){
    if (isEmpty()) return 0;
    if (head()==x) return 1+tail()->howOftenContains(x);
    return tail()->howOftenContains(x);
  }
};

int main(){
  using namespace std;
  L<string>* xs
    =new L<string>("stand"
    ,new L<string>("and"
    ,new L<string>("deliever"
    ,new L<string>("your"
    ,new L<string>("money"
    ,new L<string>("or"
    ,new L<string>("your"
```

```
46      ,new L<string>("live"
47      ,new L<string>()))))))));
48
49    cout<<xs->howOftenContains("your")<<endl;
50    cout<<xs->howOftenContains("money")<<endl;
51  }
```

Aufgabe 4 Schreiben Sie folgende generischen Funktionen für die STL:

```
1  template <typename Iterator,typename ElemType>
2  ElemType groesstes(Iterator begin,Iterator end
3                     ,bool(* groesser)(ElemType,ElemType));
```

`groesstes` soll das nach der mit übergebenen Vergleichsfunktion größte Element im Iteratorbereich sein. Falls der Iteratorbereich kein Element enthält, soll die Funktion eine Ausnahme werfen.

Schreiben Sie eine Beispielanwendung Ihrer Funktion, die für eine Sammlung von Strings den längsten dieser Strings berechnet.

Lösung

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  template <typename Iterator,typename ElemType>
6  ElemType groesstes(Iterator begin,Iterator end
7                     , bool(* groesser)(ElemType,ElemType)){
8    if (begin==end) throw "no element found";
9    ElemType result = *begin;
10   begin++;
11   for (;begin!=end;begin++){
12     if (groesser(*begin,result)) result=*begin;
13   }
14   return result;
15 }
16
17 template<typename a>
18 bool gr(a x,a y){return x>y;}
19
20 bool longestString(string x,string y){return x.size()>y.size();}
21
22 int main(){
23     int ys []= {321,432,5,532,5321,5432,5432,4321,32};
24     std::cout<< groesstes(ys,ys+9,gr<int>)<<std::endl;
25     vector<string> xs;
26     xs.insert(xs.end(),"stand");
27     xs.insert(xs.end(),"and");
```

```
28      xs.insert(xs.end(),"deliever");
29      xs.insert(xs.end(),"your");
30      xs.insert(xs.end(),"money");
31      xs.insert(xs.end(),"or");
32      xs.insert(xs.end(),"your");
33      xs.insert(xs.end(),"life");
34      std::cout<< groesstes(xs.begin(),xs.end(),gr<string>)<<std::endl;
35      std::cout<< groesstes(xs.begin(),xs.end(),longestString)<<std::endl;
36  }
37
```

Aufgabe 5 Schreiben Sie in dieser Aufgabe Klassen für die Darstellung von XML Dokumenten.

a) Schreiben Sie eine Klasse XMLNode. Sie soll eine Methode `theChildren` haben, die ein `vector`-Objekt mit dem Kinderknoten des `this`-Knotens zurückgibt. Mit einer Methode `addChild(XMLNode* c)` soll ein zusätzlicher Knoten in die Kinderliste eingefügt werden. Desweiteren soll es die Methoden `std::string getName` und `std::string getvalue()` geben.

b) Schreiben Sie eine Unterklasse `TextNode` der Klasse `XMLNode`. Sie soll in einem Konstruktor den Text des Textknotens als String übergeben bekommen. `getName` soll so überschrieben sein, daß `"xml:text"` zurückgegeben wird, und `getValue`, so daß der im Konstruktor übergebene String zurückgegeben wird. Der Aufruf von `addChildren` soll zu einen Fehler führen

c) Schreiben Sie eine Unterklasse `ElementNode` der Klasse `XMLNode`. Sie soll in einem Konstruktor einen Tagnamen übergeben bekommen. `getName` soll so überschrieben sein, daß der Tagname zurückgegeben wird, und der Aufruf von `getValue`, soll zu einem Fehler führen.

d) Fügen Sie Ihrer XML-Knotenklasse eine Funktion `boolean containsTag(std::string tagName)` hinzu, die angibt, ob es einen Elementknoten mit dem Tag `tagName` im Dokument gibt.

e) Schreiben Sie einen kleines Testbeispiel für die Benutzung Ihrer XML-Knotenklasse, in dem für folgendes kleine Dokument ein Baum aufgebaut wird:

```
1  <aufgabe><punkte>12</punkte>Schreiben Sie einen <b>Test</b></aufgabe>
```

Lösung

```
1   #include <vector>
2   #include <string>
3   #include <iostream>
4
5   using namespace std;
6
7   class XMLNode{
8     vector<XMLNode*> children;
9   public:
10    vector<XMLNode*>& theChildren(){return children;}
11    virtual void addChild(XMLNode* n){
12      children.insert(children.end(),n);
```

```
13    }
14    virtual string getName(){return "";}
15    virtual string getValue(){return "";}
16    bool containsTag(string tag){
17      if (tag == getName()) return true;
18      for (int i=0;i<theChildren().size();i++)
19        if (theChildren()[i]->containsTag(tag)) return true;
20      return false;
21    }
22  };
23
24  class TextNode:public XMLNode{
25    string value;
26  public:
27    TextNode(string value):value(value){}
28    virtual void addChildNode(XMLNode* n){
29      throw "cannot add child to text node";
30    }
31    virtual string getName(){return "xml:text";}
32    virtual string getValue(){return value;}
33  };
34
35  class ElementNode:public XMLNode{
36    string name;
37  public:
38    ElementNode(string name):name(name){}
39    virtual string getName(){return name;}
40    virtual string getValue(){throw "no value for element nodes";}
41  };
42
43  int main(){
44    ElementNode* aufgabe= new ElementNode("aufgabe");
45    ElementNode* note   = new ElementNode("note");
46    ElementNode* b   = new ElementNode("b");
47    TextNode* t0 = new TextNode("12");
48    TextNode* t1 = new TextNode("Schreiben Sie einen ");
49    TextNode* t2 = new TextNode("Test ");
50    note->addChild(t0);
51    b->addChild(t2);
52    aufgabe->addChild(note);
53    aufgabe->addChild(t1);
54    aufgabe->addChild(b);
55    cout << aufgabe->containsTag("b")<<endl;
56    cout << aufgabe->containsTag("em")<<endl;
57  }
58
```