1. Wir betrachten wieder die Klasse `LinkedList` aus dem Praktikum 7. Implementieren Sie einen Kopier-Konstruktor und (falls noch nicht vorhanden) einen Destruktor für die Klasse `LinkedList`. Überprüfen Sie die Implementierung mit dem folgenden Hauptprogramm.

```
void ausgabe(const char* text){
  std::cout << text << std::endl;
}

// Test der LinkedList-Klasse
int main(){
  using namespace std;
  LinkedList liste;

  liste.append("Element 1");
  liste.insert("Element 2", 2);
  LinkedList liste2 = liste;
  cout << "Liste:" << endl;
  liste.visit_all(ausgabe);
  liste.remove(2);
  cout << "Liste:" << endl;
  liste.visit_all(ausgabe);
  cout << "Liste2:" << endl;
  liste2.visit_all(ausgabe);

  return 0;
}
```

   Das Programm muss dann die folgende Ausgabe erzeugen.

```
Liste:
Element 1
Element 2
Liste:
Element 1
Liste2:
Element 1
Element 2
```

2. Ergänzen Sie nun die Klasse `LinkedList` von Blatt 7 um einen Iterator. Der Iterator soll ähnlich nutzbar sein, wie die Ihnen bereits bekannten Iteratoren für Java-Sammlungen.[1] Schreiben Sie die Klasse `Iterator` in Form einer reinen Schnittstelle

[1] Hinweis: Die Iteratoren der C++-Standardbibliothek arbeiten nach einem anderen Muster. Für die Implementierung würden wir die Operatorüberladung benötigen.

(abstrakten Klasse) mit den folgenden Elementfunktionen[2]:

- `bool hasNext()` : Liefert den Wert `true` genau dann, wenn mindestens ein Element in der Liste ist, welches noch nicht mit der Funktion `next()` betrachtet wurde.
- `const char* next()` : Liefert den nächsten Text aus der Liste.

Erweitern Sie die Klasse `LinkedList` um die Elementfunktion `Iterator* iterator()`. Implementieren Sie dann diese Schnittstelle mit einer konkreten Klasse `ListIterator`. Diese Klasse können Sie als normale Klasse oder als innere Klasse der Klasse `LinkedList` implementieren. Ändern Sie dann die Elementfunktion `visit_all()` der Klasse `LinkedList`. Hier soll nun der Iterator verwendet werden, um die komplette Liste zu durchlaufen. Testen Sie diese Implementierung mit dem Hauptprogramm aus der Aufgabe 1.

3. Nehmen Sie die Klassen `LinkedList`, `Iterator` und `ListIterator` in den Namensraum `fhdo_pk2` auf. Passen Sie das Hauptprogramm entsprechend an.

[2] In C++ können Sie den elementaren Datentyp `bool` verwenden. Die Wahrheitswerte lauten `true` und `false`. Bei Bedarf findet eine Typkonvertierung zwischen `bool` und `int` statt.