Aufgabe 1:

1. Erweitern Sie die gegebene Implementation einer doppelt verketteten Liste um folgende, in der Vorlesung besprochene, Methoden:

   - len()
   - get(int i)
   - set(int i)
   - insert_atStart (T a)
   - insert_atEnd(T a)
   - remove_at(int i)
   - insert_at (int i, T a)

   Fügen Sie zusätzlich eine toString Methode hinzu, die die einzelnen Inhalte der Liste von vorne nach hinten ausgibt. Greifen Sie dafür auf die toString Methode der einzelnen Objekte zurück.

   Testen Sie Ihre Implementation!

```
public class DoubleLinkedList<T> {
    class Node<U> {
        U data;
        Node<U> pred, succ;
        Node (U data, Node<U> pred, Node<U> succ){
                this.data = data;
                this.pred = pred;
                this.succ = succ;
        }
    }
    Node<T> head, tail;

    DoubleLinkedList (){
        head = new Node<T>(null, null, null);
        head.pred = null;
        head.succ = tail;
        tail.pred = head;
        tail.succ = null;
    }
}
```

2. Gegeben sei eine einfach verkettete Liste mit n Elementen. Entwerfen Sie einen Algorithmus, mit dem sich testen lässt, ob die Liste einen Kreis enthält. Wie ist seine Laufzeit? Ist dies auch in Zeit $\mathcal{O}(n)$ möglich?

Aufgabe 2:
*E*inst schickte Frau Mutter, mit eiligen Worten,
klein Timmi zum Markt: "Es gibt neue Waren!
Strukturen für Daten! Verschiedenste Sorten!
Bring mir eine Queue – doch müssen wir sparen.

Drei Groschen für eine, das sollte schon passen."
So lief Timmi fort, den Marktplatz voraus
doch kaum war er dort, konnt' er sich nicht lassen
"Zwei Groschen reichen doch sicherlich aus!"

Von Gier besiegt und mit Eis in der Hand
erschrak Timmi heftig, als er sich besann
"Drei Groschen die Queue" las er dort am Stand–
er zerbrach sich den Kopf und das Eis zerann.

Mit zwei Stacks im Rucksack kehrt er schließlich heim
–wegen reichlicher Ernte bekam er sie beide–
Doch scholt' ihn die Mutter "Das kann doch nicht sein!
Stacks gehen verkehrt, weshalb ich sie meide!"

"Eine Queue, liebe Mutter, bau ich drumherum:
Enqueue-en werd' ich nur in den ersten der beid'
Und will ich dequeue-en, so füll ich sie um.
Amortisiert wird das klappen – in konstanter Zeit."

Hilf klein Timmi beim Zusammenbau der beiden Stacks zu einer Queue! Seine simulierte Queue soll folgendermaßen funktionieren:

```
dequeue() {
    if (right.isEmpty()) {
        while(!left.isEmpty())
        right.push(left.pop());
    }
    return right.pop();
}
```

Erstellen Sie eine Klasse TimmisQueue, die zwei Stacks als private Attribute hat. Nutzen Sie für die Stacks java.util.Stack. Implementieren Sie:

- Einen Konstruktor,
- dequeue,
- enqueue,
- isEmpty.

Testen Sie Ihre Implementation!