Aufgabe 1

Erkläre, wofür das Silbenkurzwort *Deque* genau steht.

Aufgabe 2

Beschreibe sechs verschiedene sinnvolle Operationen für den abstrakten Datentyp *Deque*.

Aufgabe 3

Schreibe eine syntaktisch korrekte Funktion `palinChecker(text)`, der überprüft, ob die Zeichenkette `text` ein Palindrom ist und entsprechend den Wert `True` bzw. `False` zurückliefert.

Du kannst auf eine Implementation der Klasse `Deque` zurückgreifen, die Deque-Objekte mit den üblichen Methoden erzeugen. Die Namen dieser Methoden können „frei“ gewählt werden, so lange sie die betreffende Operation unmissverständlich beschreiben.

Aufgabe 4

Stelle die Node-Objekte (Zeilen 19–23) mit ihren Daten und den allfälligen Verbindungen zu anderen Knoten nach der Ausführung des folgenden Codes grafisch dar. Anleitung: Stelle jeden Node durch ein zweiteiliges Rechteck für Daten und Zeiger sowie die Verbindungen (Referenzen) durch Pfeile (in richtiger Richtung). Zeiger, die auf kein Objekt zeigen, sind durch ein Kreuz zu kennzeichnen.

```
1  class Node:
2
3      def __init__(self, data):
4          self.data = data
5          self.next = None
6
7      def getData(self):
8          return self.data
9
10     def setData(self, newData):
11         self.data = newData
12
13     def getNext(self):
14         return self.next
15
16     def setNext(self, newNext):
17         self.next = newNext
18
19 a = Node(5)
20 b = Node(3)
21 c = Node(7)
22 c.setNext(a)
23 b.setNext(c)
```

Aufgabe 5

Beschreibe den logischen Fehler im Code auf den Zeilen 24–26, der einen neuen Knoten an den Anfang einer einfach verketteten Liste einfügen soll.

```
1  class Node:
2
3      def __init__(self, data):
4          self.data = data
5          self.next = None
6
7      def getData(self):
8          return self.data
9
10     def setData(self, newData):
11         self.data = newData
12
13     def getNext(self):
14         return self.next
15
16     def setNext(self, newNext):
17         self.next = newNext
18
19 class LinkedList:
20
21     def __init__(self):
22         self.head = None
23
24     def add(self, data):
25         temp = Node(data)
26         self.head = temp
```

Aufgabe 6

Bestimme die Laufzeitkomplexität $O(\dots)$ der folgenden, im Unterricht implementierten Operationen, auf einer einfach verketteten Liste mit n Elementen.

(a) Entfernen eines Elements an beliebiger Position

(b) Hinzufügen eines Elements am Anfang der Liste

(c) Durchsuchen der Liste nach einem Element

(d) Bestimmen der Länge der Liste

(e) Testen, ob die Liste leer ist

Aufgabe 7

Erkläre die Bedeutung des Begriffs *Garbage Collection* in der Informatik.

Aufgabe 8

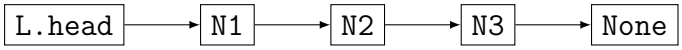
Stelle die einfach verkettete Liste, die am Ende des folgenden Codeabschnitts vorliegt, grafisch dar.

```
from linkedlist import Linkedlist

mylist = Linkedlist()
mylist.add(4)
mylist.add(1)
mylist.add(9)
mylist.remove(1)
mylist.add(2)
```

Aufgabe 9

Beschreibe, wie in der folgenden graphischen Darstellung der Knoten `N2` aus der einfach verketteten Liste `L` entfernt wird.



Aufgabe 10

Scheibe syntaktisch korrekten Code für die Methode `search(self,item)`, um herauszufinden, ob sich der Wert `item` in der einfach verketteten Liste befindet. Abhängig vom Resultat der Suche, soll die Methode `True` bzw. `False` zurückliefern. Verwende als Grundlage folgenden Code:

```
class Node:

    def __init__(self, data):
        self.data = data
        self.next = None

    def getData(self):
        return self.data

    def setData(self, newData):
        self.data = newData

    def getNext(self):
        return self.next

    def setNext(self, newNext):
        self.next = newNext

class Linkedlist:

    def __init__(self):
        self.head = None
```