W06H01 - Listige Launemacher

Hausaufgabe Easy

Points: 8 of 8 Assessment: manual

Submission due: 8 months ago

Complaint due: 8 months ago

Resume practice in exercise

100% (8 months ago) GRADED

Recent results:

75% (9 months ago)	100% (9 months ago)	100% (8 months ago)	100% (8 months ago)
GRADED	GRADED	GRADED	GRADED

Show all results

Tasks:

Listige Launemacher

Die Clownuine der lokalen Pinguinschule haben sich das erste Mal seit längerer Zeit wieder getroffen. Leider haben sie in der Zwischenzeit vergessen, wie sie andere Pinguine am besten zum Lachen bringen können, weshalb sie sehr deprimiert sind. Um aus diesem Loch zu kommen, haben sie beschlossen, ihrem zweitliebsten Hobby zu folgen: Programmieren. Ihr Ziel war die Implementierung einer objektorientierten, rekursiven Liste für Ganzzahlen. Die einfachsten Methoden, die du auch im Template findest, haben sie schnell geschafft. Doch leider ist jetzt die Luft raus und sie verzweifeln an den letzten Methoden. Du musst ihnen dringend helfen, sonst könnte die Freude im Pinguinland für immer verschwinden!

Allgemein

- Die Basisimplementierung der doppelt verketteten Liste bestehend aus `RecIntListElement`s findest du in `RecIntList`
- Alle Teilaufgaben müssen in der Klasse `RecIntList` implementiert werden
- Alle Teilaufgaben müssen rekursiv implementiert werden, d.h. zur Lösung einer Teilaufgabe muss die Methode selbst oder eine genutzte Hilfsmethode sich selbst aufrufen. Schleifen sind nur in der `main`-Methode und den vorgegebenen `toString`s erlaubt.
- Es dürfen Hilfsmethoden in `RecIntList` und `RecIntListElement` hinzugefügt werden.
- Andere Änderungen am Template geschehen auf eigene Gefahr und können zu Punktverlust führen, falls dadurch die Tests nicht mehr funktionieren! Insbesondere `static` Variablen können zu Problemen führen.
- Wie immer bei einfachen Aufgaben gilt: Punkte werden mit den Public Tests verteilt und es gibt pro Teilaufgabe einen Public Test. Nach der Deadline kann es nur dann Punktabzug geben, wenn die Aufgabe nicht rekursiv gelöst wurde.

Aufgaben

countThresh No results

Implementiere die Methode `countThresh`, die einen `int` übernimmt und die Summe der Werte in der Liste, die kleiner, gleich groß, bzw. größer dem übergebenen Wert sind. Die Summen sollen in einem Array in entsprechender Reihenfolge (kleiner, gleich, größer) zurückgegeben werden. Die Methode kann auf jeder möglichen Liste aufgerufen werden, aber es kommt in unseren Tests nicht zu `long`-Overflows. Außerdem sollte die Liste beim Aufruf nicht zerstört werden. Beispiel:

```
Input: [1,2,3,4,5], threshold = 3
Output: [3,3,9]
```

kinguinSort No results

Implementiere die Methode `kinguinSort`, die die Liste mit Hilfe des Lieblingssortieralgorithmus des Kinguins Konrad sortiert: KinguinSort. Dabei wird die Liste einmal von vorne nach hinten durchlaufen und jedes Element, das der Sortierung widerspechen würde, wird einfach entfernt. Die Methode übernimmt zusätzlich einen Parameter, der angibt, ob aufsteigend (`true`) oder absteigend (`false`) sortiert werden soll. Gleich große Werte sind in jedem Fall ok und sollen nicht entfernt werden. Die Methode kann auf jeder möglichen Liste aufgerufen werden. Beispiel:

```
Input: [3,2,4,7,1,6,5,9,8]
Ergebnis (increasing = true): [3,4,7,9]
denn: 3>2 -> widerspricht der Sortierung; 3<4 -> OK; 4<7 -> OK; 7<1 -> widerspricht der Sortierung; usw.
Ergebnis (increasing = false): [3,2,1]
denn: 3>2 -> OK; 2<4 -> widerspricht der Sortierung; 2<7 -> widerspricht der Sortierung; 2>1 -> OK; usw.
```

reverse No results

Implementiere die Methode `reverse`, die die Reihenfolge der Liste umdreht, d.h. das ursprünglich letzte Element soll jetzt das erste sein und andersrum. Wichtig dabei ist, dass keine neuen `RecIntListElement`-Instanzen erstellt werden und die `value`-Attribute der bestehenden Instanzen nicht verändert werden. Außerdem soll das Ganze effizient passieren, die Liste darf also nur einmal durchlaufen werden. Die Methode kann auf jeder möglichen Liste

aufgerufen werden. Beispiel:

```
Input: [1,2,3,4,5]
Output: [5,4,3,2,1]
```

(Auf Nachfrage in Zulip: Getestet wird mit 10000 Elementen, aber denk daran: Artemis Build Clients sind langsam und ausschlaggebend für die Tests)

zip No results

Implementiere die `static` Methode `zip`, die zwei Listen übernimmt und diese wie folgt zu einer Liste zusammenführt. Das Ergebnis soll in der ersten übergebenen Liste gespeichert werden, die zweite Liste darf bei der Ausführung zerstört werden. Auch hier sollen wie bei `reverse` keine neuen `RecIntListElement`-Instanzen erstellt oder `value`-Attribute verändert werden. Es können alle möglichen Listen übergeben werden und du darfst davon ausgehen, dass die übergebenen Listen unterschiedlich und nicht `null` sind.
Liste l: $l_0, l_1, l_2, \ldots$
Liste m: $m_0, m_1, m_2, \ldots$
Liste l nach zip(l, m): $l_0, m_0, l_1, m_1, l_2, m_2, \ldots$
Die Listenelement wechseln sich also immer zwischen den Listen ab. Sollten einer Listen die Elemente ausgehen, werden nur noch die übrigen Elemente der anderen Liste verwendet. Beispiel:

```
Input:
 l1: [1,3,5,7,8]
 l2: [2,4,6]
Ergebnis l1: [1,2,3,4,5,6,7,8]
(nach 6 ist l2 leer, daher werden die restlichen Elemente von l1 ohne Änderungen der Reihenfolge angehängt)
```

Vergiss nicht, auch die `prev`-Zeiger der einzelnen Elemente korrekt zu setzen!

Hinweis:

Sollten alle "xxx - public feedback" Tests vor der Deadline passen, gibt es 100% der Punkte (sofern Rekursion zum Lösen der Aufgaben verwendet wurde).

Hidden Tests

Diese Tests werden nach der Deadline Feedback anzeigen und sind inhaltlich gleich mit den Pubilc Tests.

- **countThresh Hidden Tests** No results
- **kinguinSort Hidden Tests** No results
- **reverse Hidden Tests** No results
- **zip Hidden Tests** No results

Lösungsvorschlag
Tests

FAQ

Q: Wofür ist das FAQ?

A: Um häufig gestellte Fragen zu sammeln. Wer trotzdem noch fragt macht Pinguine traurig.

Q: Was bedeutet es eine neue Instanz zu erstellen?

A: Man erstellt mit `new` immer eine neue Instanz. Lokale Variablen sind erlaubt, solange sie nur Referenzen auf bereits existierende Objekte speichern.

Exercise details	
Release date:	Nov 24, 2022 18:30
Submission due:	Dec 4, 2022 18:00
Assessment due:	Dec 11, 2022 18:00
Complaint due:	Dec 18, 2022 18:00

Every student is allowed to complain once per exercise. In total 1000 complaints are possible in this course. You still have **998** complaints left.

How useful is this feedback to you?

★ ★ ★ ★ ★