**Informatik I** **Wintersemester 2015/16**
Prof. Dr. Carsten Damm Georg-August-Universität Göttingen
Dr. Henrik Brosenne Institut für Informatik

# Übungsblatt 10

## E-Learning

**Absolvieren Sie die Tests bis Di., 26.01.2016, 20 Uhr.**

Die Tests sind in der Stud.IP-Veranstaltung *Informatik I* unter *Lernmodule* hinterlegt.
Sie können einen Test **nur einmal durchlaufen**. Sobald Sie einen Test starten steht Ihnen nur eine **begrenzte Zeit** zu Verfügung, um den Test zu bearbeiten.
Alle Punkte, die Sie beim Test erreichen, werden ihnen angerechnet.

### ILIAS 4-Minuten-Aufgaben – 8 Punkte

Absolvieren sie die Tests

- *Informatik I - ILIAS 10 Teil 1*,
- *Informatik I - ILIAS 10 Teil 2*.

(8 Punkte)

### LON-CAPA – 30 Punkte

**Stacks und Queues**

Absolvieren Sie im Test *Informatik I - LON-CAPA* die *Übung 10*.
(30 Punkte) + (6 Zusatzpunkte)

# Übung

**Abgabe bis Di., 26.01.2016, 18 Uhr.**

Werfen Sie Ihre Lösung in den Zettelkästen Ihrer Gruppenübung. Für die Übungen im Nordbereich stehen die Zettelkästen im Sockelgeschoß (Ebene -1) des Instituts für Informatik.

Wenn Ihre Übung im Südbereich stattfindet, klären Sie mit Ihrem Tutor wo die Lösungen abzugeben sind.

Achten Sie darauf, dass Ihr **Name**, Ihre **Gruppe** und Ihre **Matrikelnummer** auf **jedem** Blatt stehen!

Falls Ihre Lösung mehrere Blätter umfasst, heften Sie diese bitte zusammen.

## Aufgabe 1 – 29 Punkte

### Vollständige Induktion

**Frage**
Wieviele Möglichkeiten $b(n)$ gibt es aus $n$ Personen 4 für eine Bridgerunde auszuwählen?

**Antwort**
Für alle $n \geq 4$ gilt $b(n) = \frac{n(n-1)(n-2)(n-3)}{24}$.

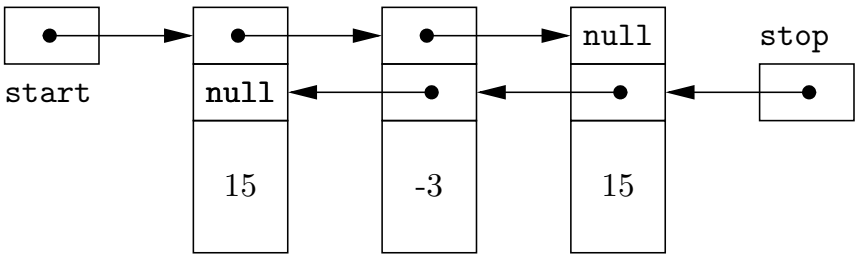
Beweisen Sie die Korrektheit der Antwort mit vollständiger Induktion. Gliedern Sie Ihre Beweisführung wie in Abschnitt *7.8 Vollständige Induktion* des Skripts vorgegeben.
(29 Punkte)

Hinweis
Formeln, deren Korrektheit in Abschnitt 7.8 des Skripts bewiesen wurde, können Sie verwenden.

## Aufgabe 2 – 8 Punkte

### Listen

Betrachten Sie folgende graphische Darstellung einer doppelt (zweifach) verketteten `int`-Liste mit Referenzen auf Listenanfang (`start`) und Listenende (`stop`).



Auf der Liste gibt die Operationen

- `prepend`, am Anfang einspeichern.
- `append`, am Ende einspeichern
- `deleteFirst`, entferne den ersten Knoten.
- `deleteLast`, entferne den letzten Knoten.

Führen Sie nacheinander die folgenden Operationen aus und stellen Sie jedes Zwischenergebnis graphisch dar.

```
deleteFirst();
deleteLast();
deleteLast();
prepend(-3);
deleteLast();
prepend(15);
append(15);
append(4);
```

(8 Punkte)

# Praktische Übung

**Abgabe der Prüfsumme bis Di., 26.01.2016, 20 Uhr.**
**Testat von Mi., 27.01.2016, 18-20 Uhr bis Di., 02.02.2016, 8-10 Uhr.**

**Hilfe** zum Bearbeiten der praktischen Übung können Sie grundsätzlich jeden Tag in den Rechnerübungen bekommen. Insbesondere in den Rechnerübungen **Di., 18-20 Uhr** und **Mi., 8-10 Uhr**, in denen **keine Testate** stattfinden.

## Abgabe der Prüfsumme

Beim Testat werden nur die Dateien testiert, deren Prüfsumme **exakt** mir der von Ihnen übermittelten Prüfsumme entspricht.

Folgendes Vorgehen wird empfohlen.

1. Legen Sie **im Rechnerpool des Instituts für Informatik** ein neues Verzeichnis an, z.B. `uebung10`, in dem Sie alle Dateien ablegen, die Sie beim Testat präsentieren möchten, wenn nötig in verschiedenen Unterverzeichnissen. Fahren Sie dann mit einer der beiden nachfolgenden Varianten fort.

   Variante 1

   Erstellen Sie eine Kopie des Verzeichnisses, z.B. `uebung10.testat`, entziehen Sie sich und allen anderen die Schreibrechte für dieses Verzeichnis und berechnen Sie die Prüfsumme des Verzeichnisses mit folgendem Befehl.
   ```
   tar c uebung10.testat | cksum
   ```

   Variante 2

   Erstellen Sie ein Archiv des Verzeichnisses mit folgendem Befehl.
   ```
   tar cf uebung10-testat.tar uebung10
   ```

   Entziehen Sie sich und allen anderen die Schreibrechte für das Archiv und berechnen Sie die Prüfsumme des Archivs mit folgendem Befehl.
   ```
   cksum uebung10-testat.tar
   ```

   Hinweise

   - Die Prüfsumme ist die erste Zahl in der Ausgabe des `cksum`-Befehls.
   - Prinzipiell sollte die Erstellung des Archivs und die Berechnung der Prüfsumme auf jedem Linux-System dasselbe Ergebnis liefern wie im Rechnerpool, im Einzelfall muss das aber getestet werden.
   - Entpacken können Sie das Archiv mit folgendem Befehl.
     ```
     tar xf uebung10-testat.tar
     ```

2. Übermitteln Sie die Prüfsumme durch Absolvieren des in der Stud.IP-Veranstaltung *Informatik I* unter *Lernmodule* hinterlegten Tests *Informatik I - ILIAS 10 Testat*.

   Hinweise

   - Sie können den Test mehrfach absolvieren, d.h. mehrere Prüfsumme übermitteln, beim Testat wir ausschließlich die letzte übermittelte Prüfsumme betrachtet.
   - Nachdem Sie die erste Prüfsumme übermittelt haben, ist bei einem erneuten Aufruf des Tests aus Stud.IP neben der Schaltfläche *Neuen Testdurchlauf starten* die Schaltfläche *Testergebnisse anzeigen* vorhanden.

     Auf der Seite *Testergebnisse* können Sie sich im Dropdown-Menü *Aktionen* zu jedem Testdurchlauf *Details anzeigen* lassen.

     Der Fragentitel *Prüfsumme - Praktische Übung* führt zu einer Seite, auf der Sie die von Ihnen übermittelte Prüfsumme einsehen können.

## Testat

Lassen Sie die Lösung der praktischen Übung während einer **Rechnerübung** testieren, dazu **müssen** Sie sich, nach Ablauf des Abgabezeitraums für die Prüfsumme, über Stud.IP einen **Termin reservieren**. Den Link zur Terminvergabe finden Sie in den Ankündigungen der Stud.IP-Veranstaltung Informatik I. **Pro Tag ist aber nur eine begrenzte Anzahl an Testaten möglich.**

Die Testate finden im **Rechnerraum IfI -1.110** statt, die für die Testate vorgesehenen Rechner sind gekennzeichnet. Die erste Hälfte der von Ihnen reservierte Testatzeit dient dazu, dass Sie an einem der Rechner **ohne Tutor das Testat vorbereiten** (melden Sie sich an; öffnen Sie eine Konsole und wechseln in das richtige Verzeichnis; starten Sie einen Browser und laden die benötigten Webseiten; etc.). Das eigentliche Testat findet in der zweiten Hälfte der Testatzeit statt, dazu kommt der Tutor zu Ihnen.

Testieren einer Aufgabe bedeutet, dass Sie dem Tutor die Aufgabe Schritt für Schritt demonstrieren können und auch in der Lage sind auf die Lösung bzw. den Lösungsweg bezogene Fragen zu beantworten.

**Bereiten Sie sich auf das Testat vor.** Das Testat eine Aufgabe kann vorzeitig beendet und ein entsprechender Punktabzug vorgenommen werden, wenn Sie z.B. auf in der Aufgabenstellung formulierte Fragen keine Antworten geben können. Beginnen Sie z.B. erst während des Testats mit der Bearbeitung der Aufgabe wird das Testat abgebrochen und Sie bekommen für die Aufgabe keine Punkte.

Öffnen Sie die **Webseite mit den übermittelten Prüfsummen** `https://www.stud.informatik.uni-goettingen.de/info1/uebung/uebung10/uebung10cksum.html` und suchen Sie die Zeile mit der von Ihnen übermittelten Prüfsumme in der ersten Hälfte der Testatzeit.

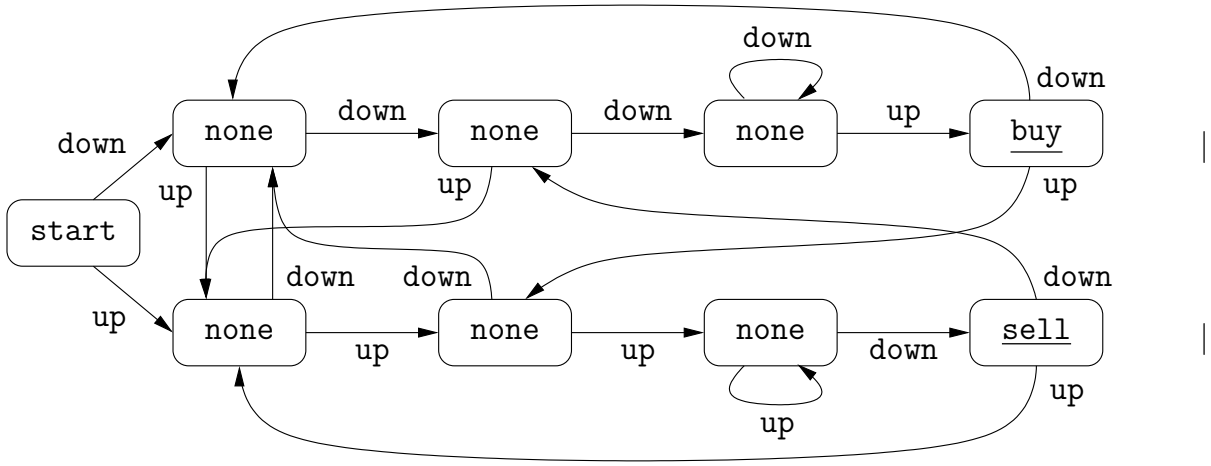
## Aufgabe 1 – 25 Punkte

### Automatisches Day-Trading

Für automatisches Day-Trading werden Muster in den Kursbewegungen einer Aktie gesucht. Zwei einfache Muster sind `buy` und `sell`.

- `sell`. Der Kurs ist mindestens dreimal hintereinander gestiegen und anschließend einmal nicht gestiegen.
- `buy`. Der Kurs ist mindestens dreimal hintereinander nicht gestiegen und anschließend einmal gestiegen.

Zum Erkennen dieser Muster kann man folgenden *endlichen Automaten* (engl. *finite state machine*) verwenden.




Der Automat besteht aus Knoten, die mit `start` (für den Knoten bei dem die Mustererkennung beginnt), mit dem aktuell erkannten Muster (`buy`, `sell`) oder der Information, dass noch kein Muster erkannt wurde (`none`), markiert sind. Jeder Knoten hat genau zwei ausgehende Kanten, die auf andere Knoten oder den Knoten selbst verweisen. Eine Kante für steigenden Kurs (`up`) und eine Kante für nicht steigenden Kurs (`down`).

Die Mustererkennung in einer Liste von Aktienkursen läuft wie folgt ab.

1. Setze als aktuellen Knoten den mit `start` markierten Knoten. Setze als aktuellen Kurs den ersten in der Liste aufgeführten Aktienkurs.
2. Lese den nächsten Aktienkurs aus der Liste und vergleiche diesen mit dem aktuellen Aktienkurs. Ist der Aktienkurs gestiegen wird der Knoten, auf den die mit `up` markierte Kante des aktuellen Knoten verweist zum aktuellen Knoten, sonst der Knoten, auf den die mit `down` markierte Kante verweist.

   Setze als aktuellen Kurs den aus der Liste gelesen Aktienkurs.
3. Ist der aktuelle Knoten mit `buy` oder `sell` markiert wurde das jeweilige Muster erkannt, sonst wurde (noch) kein Muster erkannt.
4. Sind noch Aktienkurs in der Liste vorhanden fahre fort mit Schritt 2., sonst ist die Mustererkennung beendet.

1. Die Applikation `Pattern` soll die Muster `buy` und `sell`, mit Hilfe des oben angegebenen endlichen Automaten, in einer Liste von Aktienkursen erkennen.

   Beachten Sie folgende Anforderungen.

   a) Verwenden Sie als Grundlagen die Klassen `Pattern` und `DayTrader`, sowie für Tests die Listen `example15.csv`, `xing20160106.csv` und `bmw20160106.csv`. Die zugehörigen Dateien finden Sie in der Stud.IP-Veranstaltung *Informatik I* unter *Dateien→Übung→uebung10*.

      Die Applikation `Pattern` ist lauffähig und kann, z.B. mit `example15.csv`, getestet werden.

      ```
      > java Pattern < example15.csv
      ```

      Bemerkung. Um `Pattern` zu übersetzen wird die Klasse `StdIn` benötigt.

   b) Die Klasse `Pattern` darf, bis auf Kommentare, nicht verändert werden.

   c) Erweitern Sie `DayTrader` um symbolische Konstanten (`public static final`) für die möglichen Muster (`BUY`, `SELL`), sowie die Information, dass noch kein Muster (`NONE`) erkannt wurde.

   d) Erstellen Sie eine Klasse `Node` für die Knoten des endlichen Automaten.

      - Es gibt eine `private` Instanzvariable für den aktuellen Status (z.B. `state`), die zu den in `DayTrader` definierten symbolischen Konstanten passt.
      - Es gibt `private` Instanzvariablen für die Referenzen auf die bei steigenden und nicht steigenden Aktienkurs nachfolgenden Knoten (z.B. `up` und `down`).
      - Sehen Sie die nötigen `get`- und `set`-Methoden für die Instanzvariablen vor.

        Bemerkung. Ein kombinierte `set`-Methoden (z.B. `setUpDown`) könnte sinnvoll sein.
      - Implementieren Sie einen Konstruktor, der alle Instanzvariablen mit passenden default-Werten belegt.

   e) Erweitern Sie `DayTrader`.

      - Sehen Sie einen Konstruktor vor, der aus `Node`-Objekten den oben angegeben Automaten erzeugt.
      - Es gibt `private` Instanzvariable für den aktuellen Knoten (z.B. `current`) und den aktuellen Aktienkurs (z.B. `price`).
      - Implementieren Sie die Methode `setPrice`, die den aktuellen Knoten und den Aktienkurses aktualisiert.
      - Implementieren Sie die Methode `toString`, die `buy`/`sell` zurückliefert, wenn das entsprechende Muster gefunden wurde (siehe Status des aktuellen Knotens) und sonst eine leer Zeichenkette.

   f) Versehen Sie alle Klassen mit ausführlichen Kommentaren.

   g) Ergänzung Sie, wenn nötig, `DayTrader` und `Node`, sodass der Test von `Pattern` mit `example15.csv` folgende Ausgabe liefert.

```
> java Pattern < example15.csv
    1     59.75
    2     58.00
    3     57.25
    4     57.00
    5     57.50    buy
    6     61.25
    7     63.50
    8     59.00    sell
    9     58.00
   10     57.00
   11     57.25    buy
   12     57.50
   13     59.75
   14     58.00    sell
   15     58.00
```

Weiterhin soll `Pattern` auch in `xing20160106.csv` und `bmw20160106.csv` die Muster `buy` und `sell` erkennen.

(18 Punkte)

2. Schreiben Sie eine Applikation `Invest`, die, basierend auf den Mustern `buy` und `sell` in einer Liste von Aktienkursen, den An- und Verkauf von Aktien simuliert.

   Beachten Sie folgende Anforderungen.

   a) `Invest` startet mit 10000 Euro Kapital und keinen Aktien.

   b) Nutzen Sie `Pattern` als Vorlage zum Einlesen der Aktienkurse und `DayTrader` um die Muster `buy` und `sell` zu erkennen.

      Beim Muster `buy` wird das gesamte Kapital in Aktien investiert, beim Muster `sell` werden alle Aktien verkauft.

      Bemerkung. Bei den Berechnungen können die von Java bereitgestellten Gleitkommaoperationen benutzt werden, extra Überlegungen zu Rundung etc. sind nicht nötig.

   c) Geben Sie für jeden Aktienkurs den Preis der Aktie, das vorhandene Kapital, die Anzahl der gekauften Aktien, den Gesamtwert des Depots (Kapital und Aktien) und das bei diesem Kurs gefundene Muster aus.

d) Der Test von `Invest` mit `example15.csv` sollte (abgesehen vom Format) folgende Ausgabe liefern.

```
> java Invest < example15.csv
period    price          cash    shares         value  trade
-------------------------------------------------------------
     1    59.75      10000.00      0.00      10000.00
     2    58.00      10000.00      0.00      10000.00
     3    57.25      10000.00      0.00      10000.00
     4    57.00      10000.00      0.00      10000.00
     5    57.50          0.00    173.91      10000.00    buy
     6    61.25          0.00    173.91      10652.17
     7    63.50          0.00    173.91      11043.48
     8    59.00      10260.87      0.00      10260.87   sell
     9    58.00      10260.87      0.00      10260.87
    10    57.00      10260.87      0.00      10260.87
    11    57.25          0.00    179.23      10260.87    buy
    12    57.50          0.00    179.23      10305.68
    13    59.75          0.00    179.23      10708.94
    14    58.00      10395.29      0.00      10395.29   sell
    15    58.00      10395.29      0.00      10395.29
```

e) Der Test von `Invest` mit den tatsächlichen Kursen der XING- (`xing20160106.csv`) und der BMW-Aktie (`bmw20160106.csv`) vom 06.01.2016 sollte für XING einen Gewinn von 359,00 Euro und für BMW einen Gewinn von 134,84 Euro ergeben (inklusive Rundungsfehler).

(7 Punkte)