Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion

Priv.-Doz. Dr. Frank Huch, Christoph Daniel Schulze, Sandra Dylus

Institut für Informatik

3. Klausur zur Vorlesung „Informatik für Nebenfächler“

WS 14/15

Hinweise zur Klausur:

- Sie dürfen jegliches Material, allerdings keine mitgebrachten elektronischen Geräte verwenden. Mobiltelefone bitte ausschalten.
- Zur Bearbeitung der Aufgaben sollen Sie Ihren iLearn-Account verwenden, allerdings nicht mit Ihrem normalen Passwort, sondern dem iLearn-Passwort auf Ihrem Klausurdeckblatt.
- Bei technischen Problemen kontaktieren Sie uns bitte frühzeitig. Gegebenenfalls werden wir zum Ausgleich technischer Probleme die Bearbeitungszeit verlängern, also keine Panik.
- Die Einlesezeit beträgt 10 Minuten. Diese Zeit soll ausschließlich dazu dienen, die Klausuraufgaben zu lesen. Sie dürfen in dieser Zeit nicht schreiben.
- Während der Einlesezeit werden wir Ihre Identität anhand Ihres Lichtbild- und Ihres Studienausweises überprüfen. Bitte legen Sie diese gut sichtbar auf Ihrem Tisch bereit. Unterschreiben Sie das Deckblatt erst, wenn wir vorbeikommen.
- Die anschließende Bearbeitungszeit beträgt 150 Minuten.
- Bitte notieren Sie auf jedem Blatt, das Sie abgeben möchten, Ihren vollständigen Namen und Ihre Matrikelnummer. Bei Abgaben im iLearn ist dies nicht erforderlich.
- Da die Klausur im Anschluss direkt noch einmal geschrieben wird, können Sie leider nicht früher abgeben und gehen. Sollte dies Probleme ergeben, kontaktieren Sie uns bitte frühzeitig.
- Am Ende der Bearbeitungszeit werden wir die Klausur an Ihrem Platz zusammenheften und einsammeln. Ihre elektronischen Abgaben werden wir zur Korrektur ausdrucken.
- Die Bekanntgabe der Noten und die Einsicht in die Korrekturen findet am 20. Februar zwischen 10 und 11 Uhr in Raum 715 im Hochhaus statt.
- Sie benötigen 35 Punkte um die Klausur zu bestehen. Die maximal erreichbare Punktzahl beträgt 80 Punkte. Sie können also eine 1,0 erreichen, auch wenn Sie nur 70 Punkte bearbeiten. Werden alle Aufgaben bearbeitet, so können fehlende Punkte aus anderen Aufgaben ausgeglichen werden.
- Teilen Sie sich Ihre Zeit gut ein. Beißen Sie sich nicht an (noch) fehlerhaften Programmen fest.

Aufgabe 1 - Semantik von Ausdrücken

10 Punkte

(a) Geben Sie den folgenden Ausdruck in Termbaumdarstellung, Präfix- und Postfixnotation an:

```
37 + if x >= 0 then x - 1 else x + 2 end
```

(b) Werten Sie den Ausdruck mit Hilfe einer Stackmaschine für die Belegung `x=37` aus.

(c) Ruby wertet diesen Ausdruck anders aus. Erläutern Sie, wie Ruby den Ausdruck auswertet.

Aufgabe 2 - EBNF für Funktionsdefinitionen und Präzedenzen — 12 Punkte

In der Vorlesung haben wir die Syntax von Ruby-Anweisungen mithilfe der folgenden EBNF definiert:

```
Stm ::= Stm Stm
      | Var '=' Exp ';'
      | 'while' Exp 'do' Stm 'end' ';'
      | 'for' Var 'in' Exp '..' Exp 'do' Stm 'end' ';'
      | 'if' Exp 'then' Stm ['else' Stm] 'end' ';'
      | 'puts' '(' Exp ')' ';'
```

Die Nichtterminalsymbole `Var` und `Exp` wurden ebenfalls in der Vorlesung definiert, sind aber für diese Aufgabe nicht weiter wichtig und können im Folgenden als gegeben betrachtet werden.

(a) Erweitern Sie diese EBNF um Funktionsdefinitionen. Achten Sie darauf, dass `return` nur innerhalb von Funktionsrümpfen erlaubt ist.

(b) Geben Sie Beispiele an, anhand derer Sie herausfinden können, wie Ruby Ausdrücke der Form `-a**b` interpretiert, wobei `a` und `b` beliebige ganze, positive Zahle sind. Hierbei geht es insbesondere um die Operatorpräzedenzen.

(c) In der Vorlesung haben wir Ausdrücke mit Operatorpräzedenzen definiert. Für die Operatoren `+` und `*` sieht dies folgendermaßen aus:

```
Exp  ::= Exp  '+' Exp1  | Exp1
Exp1 ::= Exp2 '*' Exp1 | Exp2
Exp2 ::= Num | '(' Exp ')'
```

Erweitern Sie diese EBNF um den zweistelligen Operator `==`. Bilden Sie hierbei die Operatorpräzedenzen, wie sie in Ruby gelten ab.

Aufgabe 3 - Tabellenartige Ausgabe zweidimensionaler Arrays — 10 Punkte

In dieser Aufgabe sollen Sie ein zweidimensionales Array hübsch als Tabelle ausgeben. Hierbei sollen alle Spalten tatsächlich als Spalten ausgegeben werden, in denen der Text linksbündig formatiert steht. Das Verhalten der zu definierenden Prozedur `pretty_print_table(t)` sollte anhand des folgenden Codestücks klar werden:

```
table = [["Frank","Huch","Programmiersprachen"],
         ["Christoph Daniel","Schulze","Echtzeitsysteme"],
         ["Thomas","Wilke","Theoretische Informatik"]]

pretty_print_table(table)
```

Resultierende Ausgabe:

```
-------------------------------------------------
|Frank           |Huch   |Programmiersprachen    |
|Christoph Daniel|Schulze|Echtzeitsysteme        |
|Thomas          |Wilke  |Theoretische Informatik|
-------------------------------------------------
```

Die Länge der Darstellung einer Spalte wird also durch die Länge des längsten in der Spalte vorkommenden Eintrags bestimmt. Der Einfachheit halber gehen wir davon aus, dass die Tabelle nur Strings als Einträge enthält.

(a) Definieren Sie zunächst eine Funktion `longest_strings_in_columns(t)`, welche zu einer gegebenen Tabelle `t` die längsten Strings in der jeweiligen Spalte bestimmt. Angewendet auf unsere Beispieltabelle würden wir das folgende Array als Ergebnis bekommen: `[16,7,23]`

(b) Definieren Sie eine Prozedur `pretty_print_table(t)`, welche unter Verwendung der Funktion aus Aufgabenteil (a) eine Tabelle `t` wie oben abgebildet ausgibt. Für die volle Punktzahl achten Sie bitte auch auf eine komplette Umrandung der Tabelle.

Aufgabe 4 - Suchbaum

13 Punkte

Ein Suchbaum verwendet die Teile-und-Herrsche Idee, um effizient Werte in einer Datenstruktur speichern und wieder nachschlagen zu können. Die Werte werden hierbei in den Knoten eines binären Baums abgelegt. Jeder innere Knoten des Baums hat zwei Kindknoten, die selbst auch wieder Suchbäume sind. Außerdem gilt für jeden inneren Knoten, dass alle Knoten im linken Suchbaum mit kleineren Elementen beschriftet sind und alle Knoten im rechten Suchbaum mit größeren Elementen beschriftet sind. Als Beispiel betrachten wir links einen Suchbaum, welcher die Elemente `1,3,5,7,9` enhält, und rechts den Suchbaum, der entsteht, wenn man zusätzlich die 6 einfügt:

Ein Suchbaum kann mithilfe verschachtelter Arrays realisiert werden. Hierbei kodieren wir einen Knoten als dreielementiges Array, wobei die Elemente der Reihe nach der linke Suchbaum, der Wert in dem Knoten und der rechte Suchbaum sind. Leere Suchbäume werden durch leere Arrays repräsentiert. Der linke Suchbaum wird zum Beispiel folgendermaßen als Array kodiert:

```
[[[[],1,[]],3,[[],5,[]]],7,[[],9,[]]]
```

Deutlicher sollte die Definition anhand der folgenden Implementierung in Ruby werden:

```ruby
def empty_tree ()
  return []
end

def insert!(t,n)
  if t == [] then
    return t[0,0] = [[], n, []]
  else
    if t[1] > n then
      insert!(t[0], n)
    else
      insert!(t[2], n)
    end
  end
end
```

(a) Bestimmen Sie das Laufzeitverhalten von `insert!(t,n)` im Best-, Average- und Worst-Case.

(b) Die Funktion `lookup(t, n)` liefert `true` zurück, falls die Zahl `n` im Suchbaum `t` vorhanden ist, und `false`, falls sie das nicht ist. Implementieren Sie die Funktion auf effiziente Weise. Sie soll dabei nicht rekursiv arbeiten, sich also nicht selbst aufrufen.

(c) Implementieren Sie auf effiziente Weise die Funktion `lookup(t, n)` so, dass sie rekursiv arbeitet.

Aufgabe 5 - Physikalische Größen 12 Punkte

Rechnet man mit physikalischen Größen, müssen die Maßeinheiten mit in Betracht gezogen werden. In dieser Aufgabe sollen Sie eine Klasse `Measurement` programmieren, die einen Wert inklusive seiner Maßeinheit modelliert und arithmetische Grundoperationen zur Verfügung stellt.

- Die Maßeinheit soll sich aus zwei Strings zusammensetzen, wobei der erste String den Zähler und der zweite String den Nenner eines Bruchs modellieren. Die Maßeinheit `m/s` würde zum Beispiel als `"m", "s"` modelliert.
- Der Konstruktor soll den Wert als Zahl sowie die Maßeinheit als zwei Strings übernehmen.
- Definieren Sie drei getter-Methoden zum Zugriff auf den Wert, sowie den Zähler und den Nenner der Maßeinheit.
- Fügen Sie Ihrer Klasse eine Methode `show_measurement()` hinzu, welche den aktuellen Wert inklusive der Maßeinheit auf der Konsole ausgibt.
- Fügen Sie Ihrer Klasse zwei Methoden `multiply(m)` und `divide(m)` hinzu, welche als Parameter `m` eine weitere `Measurement`-Instanz übergeben bekommen und eine neue `Measurement`-Instanz zurückliefern. Denken Sie daran, die Maßeinheit der neuen Instanz entsprechend zu wählen. (In einer realen Anwendung würde man den Bruch auch noch kürzen, also `(mm)/(ms)` zu `m/s` vereinfachen; das brauchen Sie hier aus Zeitgründen nicht zu tun.)
- Fügen Sie Ihrer Klasse eine Methode `add(m)` hinzu, die ähnlich wie die vorigen Methoden eine weitere `Measurement`-Instanz übergeben bekommt. Falls die Maßeinheiten der beiden zu addierenden Instanzen übereinstimmen, soll das Ergebnis eine neue `Measurement`-Instanz sein. Falls die Maßeinheiten nicht übereinstimmen, soll die Methode `nil` zurückliefern. Denken Sie daran, dass die Maßeinheiten auch dann übereinstimmen, wenn in Zählern und Nennern die Buchstaben in anderer Reihenfolge auftreten!

Beispiel

```
s = Measurement.new(50, "m", "")
s.show_measurement()
t = Measurement.new(2, "s", "")
t.show_measurement()

v = s.divide(t)
v.show_measurement()

v2 = Measurement.new(5, "m", "s")
v3 = v2.add!(v)
v3.show_measurement()
```

Für dieses Programm soll folgende Ausgabe erzeugt werden:

```
50.0 m
2.0 s
25.0 (m)/(s)
30.0 (m)/(s)
```

Aufgabe 6 - Mergesort 13 Punkte

In dieser Aufgabe sollen Sie einen Algorithmus entwickeln, welcher zwei sortierte Arrays nimmt und zu einem neuen sortierten Array kombiniert.

(a) Definieren Sie eine nicht-mutierende Funktion `merge(a, b)`, welche zwei aufsteigend sortierte Arrays von Zahlen `a` und `b` nimmt und ein neues Array konstruiert, welches genau dieselben Elemente enthält und ebenfalls sortiert ist.

```
puts(merge([2,5,8,12,18],[1,3,5,13,22,30]))
# liefert folgende Ausgabe:
[1, 2, 3, 5, 5, 8, 12, 13, 18, 22, 30]
```

Sie dürfen bei Ihrer Implementierung keine bereits bekannten Funktionen, Prozeduren oder Methoden zum Sortieren von Arrays verwenden.

(b) Verwenden Sie die Funktion aus Aufgabenteil (a), um nun den Sortieralgorithmus `merge_sort` zu implementieren. Der Algorithmus läuft wie folgt ab:

- Der Algorithmus arbeitet auf einem Array von Arrays, wobei alle inneren Arrays immer sortiert sind.
- In einem Schritt werden in diesem Array je zwei benachbarte Arrays mit Hilfe der Mergefunktion aus Aufgabenteil (a) zu einem sortierten Array gemischt.
- Das Mischen von Arrays erfolgt so lange, bis sich nur noch ein Array im Array befindet.
- Der Algorithmus beginnt mit einem Array, in dem jedes Element in ein einelementiges Array gepackt wird.

An folgendem Beispiel sollte das Verhalten des Algorithmus klar werden:

```
merge_sort([4,7,9,3,5,2,6,8])

   [4, 7, 9, 3, 5, 2, 6, 8, 1]
-> [[4], [7], [9], [3], [5], [2], [6], [8], [1]]
-> [[4, 7], [3, 9], [2, 5], [6, 8], [1]]
-> [[3, 4, 7, 9], [2, 5, 6, 8], [1]]
-> [[2, 3, 4, 5, 6, 7, 8, 9], [1]]
-> [[1, 2, 3, 4, 5, 6, 7, 8, 9]]
-> [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Aufgabe 7 - Mutierende und nicht mutierende Funktion 10 Punkte

(a) Wir betrachten das folgende Ruby-Programm:

```
a = [42,7]
b = [a,a]

b[0][0] = 73

p(a)  #-> [73,7]
p(b)  #-> [[73,7],[73,7]]
```

Erklären Sie, warum sich das Programm die oben als Kommentar notierten Ausgaben erzeugt. Skizzieren Sie hierzu auch die Objektstruktur im Speicher.

(b) Methoden auf Objekten können mutierend oder nicht mutierend sein. Wir betrachten die folgende nicht mutierende Funktionsdefinition zum Duplizieren aller Elemente in einem Array:

```
def improve(a)
  for i in 0..a.size-1 do
    if a[i] == 42 then
      a = a[0,i] + [73] + a[i+1,a.size]
    end
  end
  return a
end
```

Auf den ersten Blick erscheint diese Funktion korrekt, was aber nicht der Fall ist. Erläutern Sie, weshalb die Funktionsdefinition nicht die Konvention für nicht mutierende Funktionen aus der Vorlesung erfüllt.

Geben Sie eine Programmstück an, welches diese Funktion verwendet und woran der Fehler ersichtlich wird. Korrigieren Sie das Programm.