Algorithmen und Programmierung 2, SS 2016 — 5. Übungsblatt

Abgabe bis Freitag, 20. Mai 2016, 12:00 Uhr

24. Arithmetische Ausdrücke, 12 Punkte, Programm `berechne` aus der Vorlesung[1]
   (a) (4 Punkte) Kann der Stapel bei der Auswertung eines arithmetischen Ausdrucks zwischendurch jemals den Inhalt `( 1 + 2 * ( 3 + 4 * (` haben? Kann er jemals den Inhalt `1 + 2 * 3` (ohne Klammern!) haben? Wie ist es mit den Inhalten `( 1 * * (` und `( 1 * 2 + ( 3 * 4 + (`? Wenn ja, zeigen Sie eine Eingabe, die zu diesem Inhalt führt, wenn nein, begründen Sie Ihre Antwort.
   (b) (4 Punkte) Gibt es ungültige arithmetische Ausdrücke, die nicht zu einer Fehlermeldung führen? (Wie bei *allen Aufgaben* ist die Antwort zu begründen.)
   (c) (4 Punkte) Das Programm wurde in der Vorlesung ohne großes Nachdenken entwickelt. Argumentieren Sie mit Hilfe einer passenden Schleifeninvariante (zum Beispiel: Wie kann der Stapelinhalt aussehen?), dass das Programm korrekt ist, oder finden Sie ein Beispiel, bei dem das Programm versagt.

25. Partielle Korrektheit, 8 Punkte
   (a) (4 Punkte) Die folgende Funktion[2] verteilt die Elemente von zwei Listen neu. Beweisen Sie mit dem Hoare-Kalkül, dass die beiden Listen am Ende dieselbe Summe haben, wenn das Programm für diese Eingabelisten terminiert.

```
def Ausgleich(s1, s2):
   """Gleiche zwei Listen aus, sodass sie die gleiche Summe haben.
   s1 und s2 sind Listen mit positiven ganzen Zahlen."""
   assert all(x>0 and isinstance(x,int) for x in s1+s2)
   while sum(s1)!=sum(s2):
       if sum(s1) > sum(s2):
            a = s1.pop()
            s2.append(a)
       else:
            a = s2.pop()
            s1.append(a)
   return s1, s2
```

   (b) (4 Punkte) Warum ist in dem Programm trotzdem der Wurm drin?

26. Endrekursion, Programmieraufgabe, 10 Punkte

   Eliminieren Sie die Rekursion aus folgender Funktion[3] ohne Verwendung eines Stapels. Gehen Sie dabei schematisch nach der Methode aus der Vorlesung vor. Erstellen Sie als Zwischenschritt eine Lösung mit `goto`-Anweisungen, bevor Sie als Endprodukt ein gültiges Python-Programm schreiben.

```
def fak(n, s=1):
    if n<0:
        raise ValueError("negatives Argument",n)
    if n==0:
        return s
    return fak(n-1, s*n)
```

[1] http://www.inf.fu-berlin.de/lehre/SS16/ALP2/ausdruecke.py
[2] http://www.inf.fu-berlin.de/lehre/SS16/ALP2/Ausgleich.py
[3] http://www.inf.fu-berlin.de/lehre/SS16/ALP2/Fakultaet.py