Institut für Informatik
Scientific Computing and Bioinformatics
Prof. Dr. Andreas Hildebrandt
M.Sc. Jennifer Leclaire

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Einführung in die Programmierung - WS 2016/2017

5. Übungsblatt

Abgabe: 02.12.2016 - 10:00

Aufgabe 1: Korrektheitsbeweis *(4 Punkte)*

Sie erhalten folgendes Python-Programm:

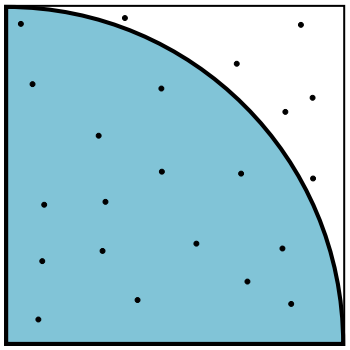
```
1 a = 2
2 n = 10
3 result = 0
4 for i in range(n):
5     result += a
```

Zeigen Sie mit einem totalen Korrektheitsbeweis, dass die Schleife $a \cdot n$ berechnet.

Aufgabe 2: Monte Carlo Simulation *(4 Punkte)*

Bei einer Monte Carlo Simulation versucht man mit Hilfe einer großen Anzahl an Zufallsexperimenten ein komplexes Problem numerisch zu lösen. In der letzten Vorlesung haben Sie *Gambler's Ruin* kennengelernt. Eine anderes beliebtes Beispiel für eine Monte Carlo Simulation ist die numerische Approximierung der Zahl *Pi* (π). Eine Anleitung dazu finden Sie im folgenden:

Stellen Sie sich ein Quadrat mit der Kantenlänge eins vor. In dieses Quadrat ist ein Viertelkreis eingeschrieben, wie in der Abbildung unten gezeigt. Erzeugen Sie jetzt gleichmäßig auf das gesamte Quadrat verteilt eine große Anzahl N an zufälligen Punkten. Dabei zählen Sie die Anzahl A der Punkte, die sich im Inneren des Kreises befinden. Die relative Häufigkeit $\frac{A}{N}$ der Punkte, die sich im Inneren des Kreises befinden ist dann ungefähr proportional zur Fläche des Viertelkreises (und diese ist $\frac{\pi}{4}$). Insgesamt gilt also:



$$\frac{4 \cdot A}{N} \approx \pi$$

Schreiben Sie ein Programm, das eine Millionen zufällige Punkte wie oben beschreiben erzeugt und so die Kreiszahl π approximiert und auf der Konsole ausgibt. Die Koordinaten (x, y) eines zufälligen Punktes bestimmen Sie dabei unter Verwendung der Funktion `uniform(a,b)` des Moduls `random`, so dass die Punkte insgesamt gleichverteilt in dem Einheitsquadrat erzeugt werden.

Aufgabe 3: Steinscher Algorithmus *(9 Punkte)*

In der Vorlesung haben Sie den euklidischen Algorithmus kennengelernt, um den ggT von zwei ganzen Zahlen a und b zu ermitteln. Sie lernen nun einen anderen Algorithmus kennen, der für zwei positive, ganze Zahlen den ggT bestimmt und dabei statt eine beliebige Division mit Rest durchzuführen ausschließlich Divisionen durch die Zahl 2 verwendet. Der Algorithmus arbeitet wie folgt:

a) Sind beide Zahlen gerade, so teile beide durch zwei.

b) Ist genau eine Zahl gerade, so teile diese durch zwei.

c) Ist keine Zahl gerade, ersetze die größere der beiden Zahlen durch (größereZahl - kleinereZahl)/2.

Wiederhole diese Schritte so oft, bis eine der Zahlen 0 wird und merke dir die Anzahl k, wie oft der erste Schritt durchgeführt wurde. Der ggT entspricht dann der Zahl, die nicht 0 wurde, multipliziert mit 2^k.

a) Entwerfen Sie ein Flow-Chart zu dem oben beschriebenen Algorithmus. Beachten Sie dazu die Hinweise zum Erstellen von Flow Charts, die auf dem letzten Übungsblatt gegeben wurden.

b) Implementieren Sie anschließend das Programm. Das Programm sollte zum Flow-Chart passen!

Die Eingabe und Ausgabe Ihres Programms soll zum Testen in Sauce folgendermaßen aussehen: Zunächst lesen Sie die Anzahl der Testfälle ein. Für jeden Testfall lesen Sie zwei Zahlen ein, für die Sie den ggT bestimmen wollen. Für jedes Paar von Zahlen (a, b) schreiben Sie den $ggT(a, b)$ in eine neue Ausgabezeile.

Eingabe:	*Ausgabe:*
`<Anzahl Testfälle>`	
`ZahlA ZahlB`	`ggT(ZahlA, ZahlB)`
`ZahlA ZahlB`	`ggT(ZahlA, ZahlB)`
`....`	`....`

Aufgabe 4: Schleifen *(3 Punkte)*

Das folgende Programm enthält drei verschiedene Schleifenkonstrukte, die jedoch alle drei zur Ausgabe des gleichen "Zahlendreiecks" führen. Erklären Sie die Unterschiede in den drei Versionen in jeweils zwei bis drei Sätzen.

```
1  # version a
2  for i in range(1, 11):
3      for j in range(1, 11):
4          if j > i:
5              continue
6          print(" ", j, end=" ")
7      print("\n")
8
9
10 # version b
11 for i in range(1, 11):
12     for j in range(1, 11):
13         if j > i:
14             break
15         print(" ", j, end=" ")
16     print("\n")
17
18
19 # version c
20 for i in range(1, 12):
21     for j in range(1, i):
22         print(" ", j, end=" ")
23     print("\n")
```

Hinweis: Der Aufruf der Funktion `print(''Hallo'', end='' '')` mit dem Parameter `end='' ''` sorgt dafür, dass nach der Ausgabe von *Hallo* kein Zeilenumbruch sondern ein Leerzeichen gesetzt wird.