Autor = Schenk Philipp, 7093700

Aufgabe 1

Eigenschaften eines Algorithmus:

* Ausführbarkeit (verständlich formuliert und ausführbar)
* Allgemeingültigkeit (nicht nur ein Einzelproblem)
* Determiniertheit (gleiche Startwerte entspricht gleichem Ergebnis)
* Determinismus (maximal eine Möglichkeit der Programmfortsetzung)
* Statische Finitheit (endlicher Quellcode)
* Dynamische Finitheit (Menge an Daten inklusive Zwischenspeicherungen sind endlich)
* Terminiertheit (kontrollierter Abbruch)

Gemäß dem Verfahren aus der Aufgabe:

* Ausführbarkeit: Wurde erfüllt. Die Beschreibung ist für den Leser verständlich, genau wie die Ausführung eines Algorithmus machbar ist.
* Allgemeingültigkeit: Größtenteils erfüllt. Ein Gegenbeispiel wäre ein Brötchen statt Toast, das nicht in den Toaster passen würde.
* Determiniertheit: Wurde erfüllt. Bei erneutem Durchlauf mit den gleichen Parametern (Toastzeit, Marmelade, …) entsteht das gleiche Ergebnis.
* Determinismus: Wurde erfüllt.
* Statische Finitheit: Wurde erfüllt.
* Dynamische Finitheit: Wurde generell erfüllt. Dennoch kann beispielsweise durch das Hängen der Toastertaste eine Endlosschleife entstehen.
* Terminiertheit: Wurde generell erfüllt. Dennoch kann beispielsweise durch einen leeren Brotbeutel ein unkontrollierter Abbruch auftreten, da kein Brot mehr vorhanden ist.

Aufgabe 2

Handlungsvorschrift 1:

* Terminierend, da sich b immer mehr (pro Durchlauf in der while-schleife) der 0 annähert. Dadurch, dass „Rest“ gerechnet wird und dieser dann h und später b entspricht pro Durchlauf wird h (und auch b) jedes Mal näher an 0 definiert und somit bricht es früher oder später kontrolliert ab.
* Determiniert, da gleiche Startwerte gleichem Ergebnis entsprechen. In diesen Rechnungen verändern sich bei gleichem Input auch die Rechnungen nicht und somit auch die Ergebnisse der Zwischenrechnungen. Keine Rechnungen mit Zufallswerten sind eingebaut.
* Deterministisch, da es eine fortlaufende Möglichkeit und keine Threads existieren. Bis die Bedingung erfüllt ist, läuft der Code in der while-schleife, danach geht es erst weiter.

Code:

a = float(input('First number: '))  
b = float(input('Second number: '))  
while b != 0:  
 h = a % b  
 a = b  
 b = h  
 print(a)  
print(a)

Handlungsvorschrift 2:

* Nicht terminierend: Der Algorithmus könnte rein technisch endlos laufen, für den Fall, dass a beispielsweise vor der while-schleife gleich 0 ist.
* Determiniert, da gleiche Startwerte gleichem Ergebnis entsprechen. Der Input ist nur indirekt zufällig, nämlich durch den Nutzer bestimmt. (Da wir nicht importieren dürfen, somit auch kein package „random“, somit gibt der Nutzer die Zahl an.) Wenn der Nutzer erneut diese Zahl eingibt, entseht durch die exakt gleichen Rechnungen auch der gleiche Ablauf uns somit das gleiche Ergebnis.
* Deterministisch, da es eine fortlaufende Möglichkeit und keine Threads existieren. Bis die Bedingung erfüllt ist, läuft der Code in der while-schleife oder rekursiv in einer Funktion, danach geht es erst weiter.

Literaturverzeichnis:

**[1]** Willers, Darwin, „Typisierung in Programmiersprachen“, *Software-System-Entwicklung*, S. 10, 15. März 2020.