Lennard Wittenberg und Kostandinos Moshos

Aufgabe 1 Algorithmen:
a.)
1.) Endlich beschrieben
Ein Algorithmus muss eine klar strukturierte Abfolge von Befehlen beziehungsweise Anweisungen haben. Außerdem benötigt ein Algorithmus einen definierten Startpunkte und festen Endpunkt haben, dass heißt ein Algorithmus darf nicht unendlich sein beziehungsweise laufen.

Beispiel für nicht Zutreffen:

Ein Programm welches in einer unendlichen Schleife festhängen bleibt erfühlt das Kriterium nicht, da es die klar definierten Instruktionen zum ausführen nicht besitzt und daher auch keinen Endpunkt hat.

2.) Deterministisch
Ein Algorithmus ist dann deterministisch, wenn bei gleichbleibenden Eingaben immer das gleiche Ergebnis entsteht, mit den immer gleichen Zwischenschritten. Bei deterministischen Algorithmen gibt es keinen Zufallsfaktor.

Beispiel für nicht Zutreffen:

Eine Funktion, die eine Zufalls generierte Variable nutzt, um das Ergebnis beziehungsweise Ausgabe zu bestimmen, könnte bei einer immer gleichen Eingabe andere Ausgaben liefern. Zum Beispiel ein 6 stelliger Würfel oder der Monte Carlo Algorithmus.

3.) Effektiv ausführbares Verfahren

Ein Algorithmus tatsächlich und praktisch mittels verfügbarer Ressourcen ausführbar sein. Ob dies in Form eines Computerprogramms oder händisch ausgeführt wird spielt keine Rolle.

Beispiel für nicht Zutreffen:

Ein Algorithmus welcher eine unpraktikable Menge an Zeit benötigt oder zu viel Speicherplatz in Anspruch nimmt, wird als nicht effektiv ausführbar bezeichnet. Eine solcher Algorithmus könnte zum Beispiel eine Zeitkomplexität von exponentieller Zeit haben. → $O(n^x) \vee O(\mathrm{e}^x)$

b.)

Algorithmus zum Kochen eines Eies:

1. Einen Topf mit Wasser zum Kochen bringen.

2. Ei in das Kochende Wasser legen.

3. Wenn das Ei normal groß, dann 9 Minuten kochen lassen. Wenn das Ei größer ist, dann 10

Minuten.

4. Nach dem Ablauf der Kochzeit wird das Ei aus dem Wasser genommen.

5. Das Ei wird kurz in kaltes Wasser gelegt, um abzukühlen.

Endlich beschrieben? Der Algorithmus hat ein Beginn und ein Ende mit einer endlichen Menge an Schritten. Die Bedingung ist erfüllt.

Deterministisch? Bei der selben Art von Ei mit den immer gleichbleibenden Schritten, ist der Algorithmus immer zu selben Ergebnis kommen. Die Bedingung ist erfüllt.

Effektiv ausführbares Verfahren? Solange die benötigten Ressourcen gegeben sind ist der Algorithmus ausführbar. Die Bedingung ist erfüllt.

2. Zahlenraten:

a.) Binary search

$Beispiel\ für\ n \in \mathbb{N} \wedge n \geq 1 \qquad n = 10$ Gesuchte Zahl = 6

1	2	3	4	5	6	7	8	9	10

Erster Schritt: setzte a auf 1 und b auf 10. Beginne Suche in der Mitte 5. $gesuchte\ Zahl > 5$

1	2	3	4	5	6	7	8	9	10

Zweiter Schritt: setzte a auf Mitte plus 1 → a = 6. b bleibt unverändert. Suche in der Mitte 8.
$gesuchte\ Zahl < 8$

1	2	3	4	5	6	7	8	9	10

$a = 6 \qquad b = 7 \qquad Berechnung\ der\ Mitte: \frac{6+7}{2}$ Abrundung auf den 6 Index. Suche bei 6. Treffer.

b.)
Der hier angewendete Suchalgorithmus heißt Binary Search und hat eine Zeitkomplexität von $O(\log(n))$. Nach jeder Iteration (Runde) verringert sich der Suchbereich um die Hälfte (vorausgesetzt der Spieler wählt jedes mal die Mitte). Da der Suchbereich sich immer so verkleinert, dass die kleineren oder alle größeren Werte herausfallen, kann der gesuchte Wert nicht durch einen Fehler aussortiert werden. Der gesuchte Wert bleibt stets im nicht durchsuchten Bereich, bis er gefunden wurde. Nach einer endlichen Zahlen an Runden hat der Spieler die gesuchte Zahl durch einengen des Suchbereiches gefunden. Die Zeit beträgt im worst case $O(\log(n))$ und im Best case 1.

Aufgabe 3 Erste Schritte in Python:

Lennard Wittenberg und Kostandinos Moshos

REPL-Befehle:

a.) Es wird nichts auf der Konsole ausgegeben. Allerdings wird im Hintergrund im Speicher eine Variable a mit dem Interger Wert 18 erstellt.

b.) Der Befehl help() druckt eine Vielzahl möglicher Befehlsnamen und deren Funktionalität aus. Der Befehl quit() beendet die REPL."

c.) Der Befehl 3 + 5 * 7 == a – 2 vergleicht die beiden Werte miteinander und prüft, ob sie gleich sind. Da 38 nicht gleich 16 ist, gibt die Konsole False aus. Die Variable a wird dadurch nicht verändert."

d.) Der Befehl a = 40 setzt die Variable a auf 40. Der nachfolgende Ausdruck 3 + 5 * 7 == a – 2 prüft, ob die beiden Seiten gleich sind. 38 ist gleich 38, daher gibt die Konsole True aus.

e.) "Der Befehl print("KDP", str(a * 50 + 2*10 + 4 - 1) + ".\n") gibt KDP 2023 aus. Die Funktion str() wandelt den gegebenen Ausdruck $a*50+2*10+4-1$ in einen String um, wobei a gleich 40 ist und somit die Summe 2023 ist. \n sorgt für einen Zeilenumbruch.

f.) Der Befehl True or (False and True) gibt True aus. Zuerst evaluiert der Ausdruck False and True zu False und anschließend evaluiert der Ausdruck True or False zu True.

g.) gibt Nein aus. Der Ausdruck a - 4, wobei a gleich 40 ist, ergibt 36. Da 36 nicht kleiner oder gleich 5 ist, wird der else-Zweig ausgeführt, der Nein druckt.

h.) Der Ausdruck gibt nichts aus, da kein print-Statement verwendet wird.

i.) Der Code gibt die Zahlen 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 aus. Die for-Schleife iteriert über die Variable i von 0 bis 9, da die range-Funktion Zahlen bis, aber nicht einschließlich, des angegebenen Endwerts erzeugt. Bei jeder Iteration wird der Ausdruck 2 * i + 1 ausgewertet, was zu einer Folge ungerader Zahlen führt.

Die range-Funktion hat drei Parameter:

- Der Startwert, der standardmäßig 0 ist, wenn nicht anders angegeben übergeben wird.
- Der Endwert, der angegeben werden muss und bestimmt, wo die Reihe endet (exklusiv).
- Der Schrittwert, der standardmäßig 1 ist und das Inkrement zwischen den Zahlen in der Reihe steuert.

j.) Der Befehl exit() wird verwendet, um die Python-REPL-Sitzung zu beenden. Der Befehl hat den selben Effekt wie quit().

Lennard Wittenberg und Kostandinos Moshos

3.c.)

Die mystery Funktion überprüft ob ein als Parameter übergebener String ein Palindrom ist. Dafür sprechen die Ausgabe für anna, ehe und effe. Wenn diese Strings übergeben werden, wird True ausgegeben. Strings wie banane, ceaser, Test, Hello Python etc. führen zu False. Darüber hinaus kann die Funktion nur dann zu korrekten Ergebnissen kommen, wenn die Strings kleingeschrieben werden. Mit einem kurzen Blick in den Quellcode kann diese Vermutung bewiesen werden.