

2. Hausaufgabenblatt

1. RAM-Programm

 20 Punkte

RAM

Schreiben Sie ein kommentiertes RAM-Programm, das die folgende Funktion $f: \mathbb{N}^5 \rightarrow \mathbb{N}$ berechnet.

$$f(x_0, x_1, x_2, x_3, x_4) = \begin{cases} \left\lfloor \frac{x_0}{5} \right\rfloor & \text{falls } x_0 \leq 4 \\ 0 & \text{sonst} \end{cases}$$

2. Bestimmen der berechneten Funktion

 20 Punkte

Wählen Sie *eine* der folgenden Funktionsdeklarationen aus, bestimmen Sie die berechnete Funktion und geben Sie diese mathematisch exakt an. Begründen Sie Ihre Aussage!

```
def f6(x):  
    y = 0  
    while (x > 0):  
        x = ((x - y) - y) - 1  
        y = (y + 1)  
    return y
```

```
def f7(x,y):  
    z = 1  
    if ((x>0) and (y>0)):  
        z = 0  
        for i in range(0,x):  
            z = (z + f7(x,(y-1)))  
    return z
```

3. Listencodierung

 60 Punkte

while

Verwenden Sie die im Skript auf den Seiten 9–10 beschriebene Listencodierung und schreiben Sie ein kommentiertes *While-Programm* (mit Syntaxprüfer testen!), das folgende Funktionen zum Erstellen und Abfragen von codierten Listen zur Verfügung stellt.




- ListCreate(): Liefert den Code der leeren Liste $\langle \rangle$.
- ListGetLength(l): Falls l der Code einer Liste ist, so wird die Anzahl der Elemente der Liste zurückgegeben. Andernfalls darf sich die Funktion beliebig verhalten.
- ListGetElement(l, i): Falls $i \geq 1$ und l der Code einer Liste mit mindestens i Elementen ist, so wird das i -te Listenelement von links zurückgegeben. Andernfalls darf sich die Funktion beliebig verhalten.
- ListAppendElement(l, e): Falls l der Code einer Liste ist und $e \geq 0$, so wird das Element e rechts an die Liste angehängt und der Code der neu entstandenen Liste zurückgegeben. Andernfalls darf sich die Funktion beliebig verhalten.

Abgabetermin: **Montag, 04.05.2026, 12:00 Uhr**

Lösungen müssen von einer Person je Abgabegruppe im WueCampus hochgeladen werden.

Jede Aufgabe in eine eigene Datei, die die Namen aller an der Lösung beteiligten Personen enthält.

Begründen Sie Ihre Behauptungen. Verwenden Sie nur Definitionen, Notationen, Resultate und Verfahren aus der Vorlesung. Verwenden Sie keine externen Quellen (z.B. Wikipedia, ChatGPT usw.).

Aufgaben-Rating: ohne Chili = Wissen,  = Anwenden,  = Verallgemeinern,  = schwierig

Lösungshinweise

Aufgabe 1:

Für $x \in \mathbb{R}$ ist $\lfloor x \rfloor$ die größte ganze Zahl, die kleiner oder gleich x ist. Falls Ihnen die Aufgabe leicht fällt, können Sie nach einem möglichst kurzen RAM-Programm suchen, das die Funktion f berechnet.

Aufgabe 2:

- Achten Sie auf typische Spezialfälle wie negative Eingaben, 0 und 1.
- Begründen Sie möglichst genau, warum das Programm die von Ihnen angegebene Funktion berechnet. Sie können beispielsweise wie folgt vorgehen:
 - Zu **f6**: Bestimmen Sie wieviel in den ersten i Schleifendurchläufen von x abgezogen wird. Sie können die Gaußsche Summenformel benutzen, um $\sum_{i=1}^n (2i - 1) = n^2$ für alle $n \in \mathbb{N}$ zu zeigen.
 - Zu **f7**: Hier lässt sich die Vermutung durch vollständige Induktion zeigen.

Aufgabe 3:

- Sie können sich beispielsweise folgende Hilfsfunktionen programmieren, wobei $|\text{bin}(n)|$ die Anzahl der Ziffern von $\text{bin}(n)$ bezeichnet.
 - $\text{binLength}(n)$: Falls $n \geq 0$, so wird $|\text{bin}(n)|$ zurückgegeben. Andernfalls darf sich die Funktion beliebig verhalten.
 - $\text{binTestBit}(n, i)$: Falls $n \geq 0$ und $1 \leq i \leq |\text{bin}(n)|$, so wird die i -te Ziffer von $\text{bin}(n)$ zurückgegeben. Andernfalls darf sich die Funktion beliebig verhalten.
- Jede der genannten Funktionen/Hilfsfunktionen lässt sich mit ca. 10 Zeilen programmieren.
- Verwenden Sie den Syntax-Checker, um die syntaktische Korrektheit Ihres While-Programms sicherzustellen.
- Testen Sie Ihr Programm, indem Sie die Liste $\langle 0, 1, \dots, n \rangle$ erzeugen und dann die einzelnen Elemente wieder auslesen. (Frage am Rande: Bis zu welchem n führt Ihr Programm dies innerhalb einer Minute durch?)
- Tests: $\langle \rangle = 2$, $\langle 0 \rangle = 34$, $\langle 1 \rangle = 46$, $\langle 5 \rangle = 718$, $\langle 0, 1 \rangle = 558$, $\langle 0, 1, 2 \rangle = 35762$, $\langle 0, 1, 2, 3 \rangle = 2288830$, $\langle 0, 1, 2, 3, 4 \rangle = 585940674$, $\langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \rangle = 10307977362219945691918$

Zusatzaufgaben


1. schnelle Paritätsbestimmung

 **20 Punkte**

while

Gesucht ist ein “schnelles” kommentiertes While-Programm, das testet, ob die Eingabe eine ungerade ganze Zahl ist. Bis zu welchen Eingabewerten (z.B. 10^4 , 10^6 , 10^8 , 10^{10} , 10^{100} , 10^{1000}) liefert Ihr Programm das Ergebnis innerhalb einer Minute? Schätzen Sie die ungefähre Anzahl der Rechenschritte in Abhängigkeit der Eingabelänge n ab. (Die Länge einer Eingabe $x \geq 0$ sei hier die Anzahl der Ziffern von $\text{bin}(x)$.)

2. schnelle Listencodierung

 **40 Punkte**

while

Gesucht ist ein While-Programm, das die Funktionen zur Listencodierung aus Aufgabe 3 “schnell” berechnet. Bis zu welchem m kann Ihr Computer innerhalb einer Minute die Liste $\langle 1, 2, \dots, m \rangle$ mittels `ListAppendElement` erzeugen und danach mittels `ListGetElement` vollständig auslesen? Schätzen Sie die ungefähre Anzahl der Rechenschritte in Abhängigkeit der Listengröße n ab. (Ist l Code einer Liste, so sei die Listengröße die Anzahl der Ziffern von $\text{bin}(l)$.)