JOHANNES GUTENBERG UNIVERSITÄT MAINZ

JG|U

Institut für Informatik

Michael Wand
Christian Ali Mehmeti-Göpel

Wintersemester 2024/25

Midterm / Rekursion

Übung 6

Einführung in die Programmierung

Über dieses Übungsblatt

Dieses Übungsblatt dient zur Selbstevaluation, da **die erste Aufgabe im exakten Wortlaut die Zwischenklausur aus dem letzten Semester ist**. Das so genannte "Midterm Exam" sollte in 30 Minuten geschrieben werden, ungefähr mit ihrem jetzigen Wissenstand. Sie können die Möglichkeit nutzen um ihren Fortschritt zu evaluieren. Zweitens beschäftigen wir uns in diesem Übungsblatt vermehrt mit dem Konzept der Rekursion.

Aufgabe Midterm

Letzte Änderung: 15. June 2023, 12:38 Uhr
20 Punkte – im Detail

Ansicht: |

1a) Schreiben Sie ein Python-Programm, dass die Zahlen von 1 bis 10 in römischen Ziffern auf der Konsole ausgibt.

- Tipp 1: Tabelle arabische → römische Ziffern: 1 → I, 2 → II, 3 → III, 4 → IV, 5 → V, 6 → VI, 7 → VII, 8 → VIII, 9 → IX, 10 → X
- Tipp 2: Vor Bearbeitung auch schon Mal Aufgabe (b) lesen

1b) Schreiben Sie ein Python-Unterprogramm `sum_roman(x,y)`, dass zwei ganze Zahlen x und y (Typ int) als Parameter bekommt und dann die Summe der beiden Zahlen als römische Zahl auf der Konsole ausgibt, sofern kein Fehler (s.u.) vorliegt. Falls ein Fehler vorliegt, soll es einfach den Text "Fehler" ausgeben. Damit kein Fehler vorliegt, müssen die folgenden Bedingungen erfüllt sein

- Beide Zahlen müssen zwischen 1 und 10 liegen
- Die Summe darf nicht größer als zehn sein.

Beispiel:

```
def sum_roman(x, y):
# ... hier steht ihr Code...
sum_roman(2, 4) # -> Ausgabe "VI"
sum_roman(12, 3) # -> Ausgabe "Fehler"
```

2) Die Folge der Tribunacci-Zahlen (nicht zu verwechseln mit Fibunaccizahlen) ist wie folgt definiert:

$$trib(n) = trib(n-1) + trib(n-2) + trib(n-3), \quad \text{fuer } n \geq 4$$
$$trib(1) = 1, trib(2) = 1, trib(3) = 2$$

Schreiben Sie ein Python-Unterprogramm trib(n), welches die n-te Tribunacci-Zahl berechnet. Sie können dabei annehmen, dass der Parameter immer ≥ 1 ist.

3) Schreiben Sie ein Python-Unterprogramm `search_max(some_list)` welches eine Liste von ganzen Zahlen (Liste von int) `some_list`

als Parameter bekommt und die größte Zahl aus dieser Liste als Ergebnis zurückgibt

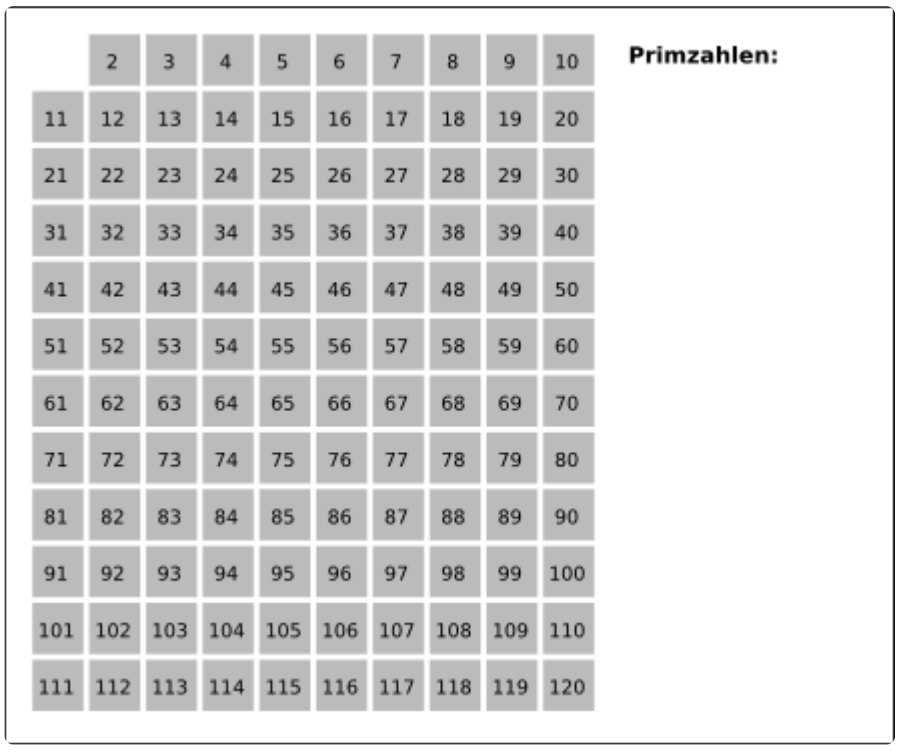
Achtung: Lösen Sie diese Aufgabe rein rekursiv – sie dürfen in ihrer Lösung keine Schleifen (while,for) benutzen! Eine Lösung mit Schleifen gibt nur maximal 5 Punkte. Die Verwendung der Python-Funktionen von `min()`, und `max()` ist **nicht** erlaubt.

Tipp: Sie sollten (wahrscheinlich) weitere Unterprogramm definieren und verwenden.

Aufgabe Sieb des Eratosthenes

Letzte Änderung: 15. June 2023, 12:38 Uhr
20 Punkte – im Detail

Ansicht: |



Bildquelle, klick mich für eine Animation.

Das Sieb des Eratosthenes ist ein einfacher Algorithmus um die ersten n Primzahlen zu bestimmen. In der klassischen Variante malt man ein 10x10 Gitter auf den Boden und beschriftet es mit den Zahlen von eins bis 100. Dann fängt man bei der Zahl zwei an und streicht alle ihre vielfachen aus dem Gitter, da sie ja durch zwei teilbar sind und somit keine Primzahlen sein können. Dann macht man weiter mit der nächsten Zahl größer zwei, die noch nicht gestrichen ist und streicht ihre Vielfache usw. Der Algorithmus endet, wenn alle vielfache von Zahlen kleiner 100 gestrichen sind und übrig bleiben alle unteilbaren Zahlen, also alle Primzahlen kleiner 100. **Das beiliegende .gif vermitelt die Idee sehr gut.**

Wir werden den Algorithmus zunächst iterativ, dann rekursiv implementieren.

1) Implementieren Sie das Sieb des Eratosthenes iterativ und geben alle Primzahlen kleiner n aus.

Tipps:

- Erstellen Sie anfangs ein Array mit n Bool-Variablen, initialisiert mit dem Wert True. Der Wert an der i-ten Stelle soll angeben ob die i-te Zahl eine Primzahl ist, deshalb nehmen wir anfangs an, dass alle Zahlen Primzahlen sind und streichen sukzessive mehr Zahlen heraus.
- Das initiale Array kann mit einfach `primes = [True] * n` erstellt werden.
- Iterieren Sie anschließend ein Mal über alle Zahlen von zwei bis n und markieren Sie alle ihre Vielfachen (kleiner n natürlich) als nicht-Primzahlen
- Wenn eine Zahl schon als nicht-prim markiert ist, sind es auch schon ihre Vielfachen. Man kann diese Zahl also z.B. in einer Schleife mit `continue` überspringen.
- Debuggen Sie ihr Programm mit dem Debugger (Siehe Blatt 1) oder mithilfe von print-Befehlen bis am Ende die richtige Ausgabe erscheint

2a) Implementieren Sie das Sieb des Eratosthenes mit einem **rekursiven** Funktionsaufruf `def get_primes(primes, current)` (diese Funktion **muss sich an einer Stelle also selbst aufrufen**), wobei `current` die Zahl darstellt, deren Vielfache gestrichen werden sollen und

geben Sie die ersten n Primzahlen aus. In ihrer Funktion darf noch eine `for` oder `while` Schleife vorkommen, solange ein rekursiver Funktionsaufruf vorkommt.

2b) Implementieren Sie das Sieb des Eratosthenes **komplett rekursiv**, also ohne das Verwenden von `for` oder `while`-Schleifen. Sie können dazu beispielsweise eine weitere rekursive Funktion `def rm_multiples(primes, nbr, current)` verwenden, um die Vielfachen von `nbr` aus der Liste zu streichen.