Elementare Zahlentheorie mit Maple (01202) **SS 15**

Klausur am 29.08.2015:

Musterlösungen

Aufgabe 1

Sei $d = \mathrm{ggT}(c, b)$. Dann gilt $d \mid c$ und $d \mid b$. Weiter gilt dann $d \mid a$. Da jeder gemeinsame Teiler von a und b aber den größten gemeinsamen Teiler von a und b teilt, folgt $d = 1$.

Aufgabe 2

Wir nehmen an, dass 3 kein Teiler von n ist. Dann hat n die Form $3k + 1$ oder $3k + 2$ für ein $k \in \mathbb{N}$. Für $n = 3k + 1$ gilt $n^2 = (3k + 1)^2 = 9k^2 + 6k + 1 = 3m + 1$ für ein $m \in \mathbb{N}$. Analog folgt für $n = 3k + 2$ dann $n^2 = (3k + 2)^2 = 9k^2 + 12k + 4 = 3l + 1$ für ein $l \in \mathbb{N}$. In beiden Fällen gilt also $n^2 + 2 = 3t$ für ein $t \in \mathbb{N}$, wobei $t > 1$ gilt, da wegen $n > 1$ gilt, dass $n^2 + 2 > 3$ ist. Da sowohl t als auch 3 ein Teiler von $n^2 + 2$ ist, widerspricht dies der Voraussetzung, dass $n^2 + 2$ eine Primzahl ist. Unsere Annahme war also falsch, und es folgt $3 \mid n$.

Aufgabe 3

Wir unterscheiden zwischen geradem und ungeradem $n \in \mathbb{N}$. Sei n zunächst ungerade. Da $1 + 2 + \cdots + (n - 1) = \frac{n(n-1)}{2}$ gilt und n ungerade ist, folgt, dass $\frac{(n-1)}{2}$ eine ganze Zahl ist. Damit ist $\frac{n(n-1)}{2}$ durch n teilbar und die Kongruenz erfüllt. Sei nun n gerade, also $n = 2m$ für ein $m \in \mathbb{N}$. Dann gilt $1 + 2 + \cdots + (n - 1) = \frac{2m(2m-1)}{2} = m(2m - 1) = m(n - 1) = mn - m \equiv -m \equiv m \not\equiv 0 \pmod{n}$. Daher ist die Kongruenz für gerade n nicht erfüllt.

Aufgabe 4

Eine zahlentheoretische Funktion ist multiplikativ, wenn für alle $m, n \in \mathbb{N}$ mit $\mathrm{ggT}(m, n) = 1$ die Gleichung $f(mn) = f(m)f(n)$ gilt. Wir betrachten $m = 2$ und $n = 5$. Dann gilt $\mathrm{ggT}(2, 5) = 1$ und $f(2 \cdot 5) = f(10) = 2$ wegen $10 \equiv 1 \pmod{3}$. Da $5 \equiv 2 \pmod{3}$ gilt, folgt $f(2)f(5) = 3 \cdot 3 = 9 \neq 2$. Daraus folgt, dass f nicht multiplikativ ist.

Aufgabe 5

(a) $\varphi(16) = \varphi(2^4) = 2^4 - 2^3 = 16 - 8 = 8$.

(b) Es ist $16 = 2^4$.

   (i) Sei $a \in \mathbb{N}$ ungerade. Dann gilt $\mathrm{ggT}(a, 16) = 1$. Mit dem Satz von Euler gilt dann

$$a^8 = a^{\varphi(16)} \equiv 1 \pmod{16}.$$

Also hat a^8 den Rest 1 bei Division durch 16.

(ii) Sei $a \in \mathbb{N}$ gerade. Dann gilt $2 \mid a$. Es folgt $16 = 2^4 \mid a^4$ und wegen $a^4 \mid a^8$ folgt $16 \mid a^8$. Somit hat a^8 den Rest 0 bei Division durch 16.

Aufgabe 6

Sei $p = x^3 + y^3$ mit $x, y \in \mathbb{N}$. Es gilt mit dem Hinweis

$$p = x^3 + y^3 = (x + y)((x - y)^2 + xy).$$

Da p eine Primzahl ist und wegen $x, y \in \mathbb{N}$ die Ungleichung $x + y \geq 2$ gilt, muss $p = x + y$ und $1 = (x - y)^2 + xy$ sein. Wieder wegen $x, y \in \mathbb{N}$ gilt $xy \geq 1$ und $(x - y)^2 \geq 0$, und daher folgt $(x - y)^2 = 0$ und $xy = 1$. Daraus folgt $x = y = 1$ und damit die Behauptung.

Aufgabe 7

1. Bei Übergabe der Liste `[3, 8, 15, 11]` wird geprüft, ob der richtige Datentyp übergeben wurde. Im vorliegenden Fall ist der Datentyp korrekt. Nach Deklaration der lokalen Variablen `i, a` wird zunächst geprüft, ob die übergebene Liste leer ist. Nur im Falle einer nicht leeren Liste wird der nachfolgende Code ausgeführt. Unsere Liste ist nicht leer, also wird der Variablen `a` der erste Wert der Liste `liste`, nämlich 3 zugewiesen. Anschließend tritt die Prozedur in eine `for`-Schleife mit dem Startwert 2 ein. Im ersten Durchlauf wird mittels einer `if`-Abfrage geprüft, ob das zweite Listenelement größer ist als der Wert der Variablen `a`. Da $8 > 3$ gilt, und damit die `if`-Abfrage positiv ausfällt, wird `a` auf den Wert 8 gesetzt. Die `for`-Schleife wird sodann erneut durchlaufen und es wird der Wert des dritten Listenelements mit dem Wert von `a` verglichen. Es gilt $15 > 8$, also wird `a` der Wert 15 zugewiesen. Wegen `nops(liste)` $= 4$ wird im letzten Schleifendurchlauf das letzte Listenelement mit dem Wert von `a` verglichen. Diesmal gilt $11 < 15$, so dass die `if`-Bedingung nicht erfüllt ist und der Wert von `a` bei 15 verbleibt. Die `for`-Schleife ist damit beendet und am Bildschirm wird der Wert von `a`, nämlich 15 ausgegeben. Damit endet auch die Prozedur.

2. Die Prozedur sucht in einer übergebenen Liste das größte Element und gibt dies am Bildschirm aus.

Aufgabe 8

Unsere Prozedur `primfaktoren(z::integer)` benötigt drei lokale Variablen `M, a, i`. Zunächst initialisieren wir `M` als die leere Menge. Anschließend weisen wir der Variablen `a` mit Hilfe des Befehls `ifactors()` die Liste der Primfaktoren zu, ohne das Vorzeichen zu berücksichtigen. Mit einer `for`-Schleife bauen wir die Menge `M` auf, indem wir alle

Elemente der Liste a durchlaufen und das jeweils erste Element der Menge zufügen. Am Ende der Prozedur geben wir die Menge M am Bildschirm aus. Eine mögliche Prozedur könnte folgendermaßen aussehen:

```
primfaktoren := proc(z::integer) # gibt alle Primfaktoren von z
                                   in einer Menge M aus
local M, a, i;
M := {};
a := ifactors(z)[2];
for i from 1 to nops(a) do
  M := M union {op(a[i][1])};
od;
print(M);
end:
```