Elementare Zahlentheorie mit Maple (01202) SS 2013

Klausur am 24.08.2013:

Musterlösungen

Aufgabe 1

Sei $n \in \mathbb{N}$ und $\mathrm{ggT}(n, 6) = 1$. Für n kommen also nur die Zahlen mit den Resten 1 und 5 modulo 6 in Betracht. Daher gilt $n = 6k \pm 1$ für ein $k \in \mathbb{Z}$. Daraus folgt $n^2 = (6k \pm 1)^2 = 36k^2 \pm 12k + 1 = 24k^2 + 12k(k \pm 1) + 1$, und damit $n^2 - 1 = 24k^2 + 12k(k \pm 1)$. Da k und $k \pm 1$ zwei aufeinanderfolgende Zahlen sind, ist das Produkt daraus gerade. Damit folgt $24 \mid n^2 - 1$.

Aufgabe 2

Sei $n + i$ mit $2 \leq i \leq p$ eine der Zahlen. Wenn i eine Primzahl ist, dann gilt $i \mid n + i$ und $n + i$ ist zusammengesetzt. Falls i keine Primzahl ist, so ist i durch eine Primzahl p_0 mit $p_0 < i$ teilbar. Dann gilt aber auch $p_0 \mid n + i$, und $n + i$ ist ebenfalls zusammengesetzt.

Aufgabe 3

31 ist eine Primzahl und es gilt $\mathrm{ggT}(2, 31) = 1$. Aus dem Kleinen Satz von Fermat folgt $2^{30} \equiv 1 \pmod{31}$. Wir erhalten

$$2^{990} = (2^{30})^{33} \equiv 1^{33} \equiv 1 \pmod{31}.$$

Damit gilt

$$2^{1000} = 2^{990} \cdot 2^{10} \equiv 1 \cdot (2^5)^2 \equiv 1^2 \equiv 1 \pmod{31}.$$

Aufgabe 4

1. Für ein $n \in \mathbb{N}$ mit der Primfaktorzerlegung $n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ gilt $\tau(n) = \prod_{i=1}^{r} (e_i + 1)$. Daraus folgt

$$\tau(60) = \tau(2^2 \cdot 3 \cdot 5) = (2 + 1)(1 + 1)(1 + 1) = 3 \cdot 2 \cdot 2 = 12.$$

2. Mit der Primfaktorzerlegung für n aus 1. gilt $\sigma(n) = \prod_{i=1}^{r} \frac{p_i^{e_i+1} - 1}{p_i - 1}$. Daraus folgt dann

$$\sigma(60) = \sigma(2^2 \cdot 3 \cdot 5) = \frac{2^3 - 1}{1} \cdot \frac{3^2 - 1}{2} \cdot \frac{5^2 - 1}{4} = 7 \cdot 4 \cdot 6 = 168.$$

3. Eine zahlentheoretische Funktion $f \colon \mathbb{N} \to \mathbb{N}$ heißt multiplikativ, wenn für $m, n \in \mathbb{N}$ mit $\mathrm{ggT}(m, n) = 1$ gilt

$$f(mn) = f(m)f(n).$$

Aufgabe 5

Sei (a, b, c) ein pythagoreisches Tripel. Wir nehmen zuerst an, dass $k = c$ gilt. Dann folgt $a, b \leq k - 1$. Hieraus erkennen wir sofort, dass es nur endlich viele Möglichkeiten für a und b gibt. Betrachten wir nun $a = k$, dann gilt $k^2 = c^2 - b^2 \geq c^2 - (c-1)^2 = 2c - 1$, denn es gilt $b \leq c - 1$. Da mit der Ungleichung auch $c \leq \frac{k^2+1}{2}$ gilt, gibt es für b und c nur endlich viele Möglichkeiten. Der Fall $b = k$ wird analog gelöst.

Aufgabe 6

Es gilt $27 = 3^3$. Da $3 \equiv 3 \pmod 4$ gilt und der Exponent ungerade ist, ist 27 nicht als Summe von zwei Quadraten darstellbar. Weiter gilt $45 = 3^2 \cdot 5$. Da hier der Exponent von 3 gerade ist und $5 \equiv 1 \pmod 4$ gilt, ist 45 als Summe von zwei Quadraten darstellbar. Schließlich gilt $125 = 5^3$ und wegen $5 \equiv 1 \pmod 4$ ist 125 unabhängig vom Exponenten Summe von zwei Quadraten.

Aufgabe 7

Die Prozedur lautet mit der richtigen Reihenfolge der Zeilen folgendermaßen:

```
> teilersumme:=proc(n::posint)
  local i, sum;
  i:= 1;
  sum:= 0;
  while i <= n do
    if n mod i = 0 then
      sum:=sum+i;
    fi;
    i:=i+1;
  od;
  print(sum);
  end:
```

Aufgabe 8

Wir werden in dieser Prozedur mit einem Zähler und drei `while`-Schleifen arbeiten. Wir überprüfen die Summen aller Quadratzahlen, die größer oder gleich 0 und kleiner als n sind. Es werden der Reihe nach die Quadratzahlen $0^2, 1^2, 2^2, \ldots$ aufsummiert, wobei der zweite Summand nicht kleiner als der erste, der dritte nicht kleiner als der zweite etc. sein soll. Auf diese Weise wird sichergestellt, dass nur eine Kombination der Summanden gezählt wird. Jede Summe wird auf Gleichheit mit n geprüft. Auf ein Abbruchkriterium für den Fall, dass die Summe der drei Summanden größer als n ist, wurde aufgrund der Kürze der Prozedur verzichtet. Am Ende der Prozedur wird die Variable `zaehler` ausgegeben.

Eine mögliche Prozedur könnte folgendermaßen aussehen:

```
> dreiquadrate:=proc(n::posint) # berechnet, auf wie viele Arten n als
                                  Summe von drei Quadraten dargestellt
                                  werden kann
  local i, j, k, zaehler;
  zaehler:= 0;
  i:= 0;
  while i^2 < n do
   j:= i;
   while j^2 < n do
    k:= j;
    while k^2 <= n do
     if i^2 + j^2 + k^2 = n then
      zaehler:= zaehler + 1;
     fi;
     k:= k+1;
    od;
   j:= j+1;
   od;
   i:= i+1;
  od;
  print(zaehler);
  end:
```