Elementare Zahlentheorie mit Maple (01202) WS 14/15

Klausur am 28.02.2015:

Musterlösungen

Aufgabe 1

Es gibt $x, y \in \mathbb{Z}$ mit $b = ax$ und $c = ay$. Daraus folgt $bc = (ax)(ay) = a^2(xy)$, und damit die Behauptung.

Aufgabe 2

Aus $p \mid a$ folgt $a = px$ für ein $x \in \mathbb{Z}$, und aus $p \mid (a^2 + b^2)$ folgt $a^2 + b^2 = py$ für ein $y \in \mathbb{Z}$. Es gilt

$$\begin{aligned} & a^2 + b^2 = py \\ \Rightarrow \quad & p^2x^2 + b^2 = py \\ \Rightarrow \quad & b^2 = py - p^2x^2 \\ \Rightarrow \quad & b^2 = p(y - px^2). \end{aligned}$$

Das heißt $p \mid b^2$. Für jede Primzahl p gilt: wenn p ein Teiler von ab ist, dann folgt $p \mid a$ oder $p \mid b$. In unserem Fall folgt also aus $p \mid b^2$ die Behauptung $p \mid b$.

Aufgabe 3

17 ist eine Primzahl und der Satz von Wilson besagt $16! \equiv -1 \equiv 16 \pmod{17}$. Wegen $\mathrm{ggT}(16, 17) = 1$ (Rechenregeln für Kongruenzen) können wir diese Kongruenz durch 16 teilen und erhalten $15! \equiv 1 \pmod{17}$.

Aufgabe 4

Für $n \in$ mit der Primfaktorzerlegung $n = \prod_{i=1}^{r} p_i^{e_i}$ gilt $\tau(n) = \prod_{i=1}^{r}(e_i + 1)$. Es gilt

$$\tau(n) = 10 = \begin{cases} 10 & = e_1 + 1 = 9 + 1 \quad \text{oder} \\ 2 \cdot 5 & = (e_1 + 1)(e_2 + 1) = (1 + 1)(4 + 1). \end{cases}$$

Mögliche Primfaktorzerlegungen sind demnach $n = p_1^4 p_2$ mit beliebigen Primzahlen $p_1 \neq p_2$ oder $n = p^9$ mit beliebiger Primzahl p.

Aufgabe 5

Mit Euklids Klassifikationssatz gibt es $m, n \in \mathbb{N}$ mit $m > n, \mathrm{ggT}(m, n) = 1, m \not\equiv n \pmod{2}$, so dass $a = 2mn, b = m^2 - n^2, c = m^2 + n^2$ gilt. Eine der beiden Zahlen ist gerade, so dass $mn = 2k$ für ein $k \in \mathbb{N}$ folgt. Daraus folgt $a = 2mn = 4k$, also $4 \mid a$ und damit auch $4 \mid ab$. Falls $3 \mid m$ oder $3 \mid n$ gilt, so folgt $12 \mid ab$ sofort. Wir nehmen an, dass $3 \nmid m$ und $3 \nmid n$ gilt. Daraus folgt zunächst $\mathrm{ggT}(3, m) = \mathrm{ggT}(3, n) = 1$. Mit dem kleinen Satz von Fermat erhalten wir $b = m^2 - n^2 \equiv 1 - 1 \equiv 0 \pmod{3}$, und somit $3 \mid b$. Wegen $\mathrm{ggT}(3, 4) = 1$ folgt also $12 \mid ab$.

Aufgabe 6

Mit dem Zwei-Quadrate-Satz folgt

(a) (i) $22 = 2 \cdot 11$. Der Primfaktor $11 \equiv 3 \pmod 4$ kommt mit ungeradem Exponenten in der Primfaktorzerlegung vor. Daher ist 22 keine Summe zweier Quadrate.

(ii) $26 = 2 \cdot 13$. Es kommt kein Primfaktor $p \equiv 3 \pmod 4$ vor, so dass $26 = 1^2 + 5^2$ als Summe zweier Quadrate darstellbar ist.

(b) Die Zahl 5 ist in den ganzen Zahlen eine Primzahl. Eine Gauß'sche Primzahl ist sie aber nicht. Denn wegen $5 \equiv 1 \pmod 4$ kann $5 = z\bar{z}$ geschrieben werden mit zwei nicht-assoziierten Gauß'schen Primzahlen z und $\bar{z}$, so dass $N(z) = N(\bar{z}) = 5$ gilt (Klassifikation der Gauß'schen Primzahlen). Somit kann 5 in $\mathbb{Z}[i]$ in zwei Faktoren zerlegt werden und ist deshalb keine Gauß'sche Primzahl.

Aufgabe 7

Eine mögliche Prozedur sieht folgendermaßen aus:

```
vollkommen := proc() # findet alle geraden vollkommenen Zahlen
bis 500
  local i, liste, exponent, a;
  liste := [];
  for i from 6 by 2 to 500 do
    a := i;
    exponent := 0;
    while a mod 2 = 0 do
      exponent := exponent + 1;
      a := a/2;
    od;
    if a = 2^(exponent + 1) - 1 then
      if isprime(exponent + 1) and isprime(a) then
        liste := [op(liste), i];
      fi;
    fi;
  od;
  print(liste);
end:
```

Zuerst deklarieren wir die lokalen Variablen und initialisieren im nächsten Schritt eine leere Liste. In einer `for`-Schleife werden alle geraden Zahlen von 6 bis 500 auf ihre Vollkommenheit untersucht. Um den Laufindex `i` nicht innerhalb der Schleife zu verändern, verwenden wir die Variable `a`. Mittels einer `while`-Schleife werden alle 2er Potenzen aus `a` heraus dividiert und der Wert des entsprechenden Exponenten `exp` gespeichert, um in der anschließenden `if`-Abfrage prüfen zu können, ob der „Rest" eine Mersenne'sche Primzahl ist. Ist dies der Fall, so wird die überprüfte Zahl in die Gesamtliste aufgenommen. Nach der `for`-Schleife wird die Gesamtliste am Bildschirm ausgegeben.

Aufgabe 8

(a) Die Prozedur berechnet die Anzahl aller Primzahlen $\leq$ der Eingabe n und gibt diesen Wert aus.

(b) Eine Prozedur könnte folgendermaßen aussehen:

```
klausur2:=proc(n::posint)
  local z, i;
  z := 0;
  i := 2;
  while i <= n do
    z := z + 1;
    i := nextprime(i);
  od;
  print(z);
end:
```