Aufgabe 1: In der Vorlesung wurde das Lemma von Euklid angegeben:

Lemma (von Euklid)**.** *Ist p eine Primzahl, so folgt aus $p|n_1n_2$, $p|n_1$ oder $p|n_2$.*

Es gilt auch die Umkehrung:

Lemma. *Falls für eine natürliche Zahl $n \geq 2$ gilt:*

$$n|n_1n_2 \Rightarrow n|n_1 \text{ oder } n|n_2,$$

dann ist n prim.

Beweis: Es sei $n = tt'$ mit $1 \leq t, t' \leq n$. Nach Voraussetzung gilt nun: $n|t$ oder $n|t'$. Angenommen $n|t$, da $t \leq n$ ist, haben wir $t = n$ und $t' = 1$. Eine Zahl deren einzigste Teiler 1 und die Zahl selbst sind, ist eine Primzahl. □

Wir haben mit den beiden Lemmata ein schönes Korollar:

Korollar. *Folgende Bedingungen für ein ganzes $n \geq 2$ sind äquivalent:*

1. *n ist prim*
2. *aus $n|n_1n_2$ mit $n_1, n_2 \in \mathbb{Z}$ folgt $n|n_1$ oder $n|n_2$.*

Allgemeine Bemerkung: Wenn man allgemeinere Ring betrachtet, dann wird die zweite Eigenschaft zur Definition von Primelementen herangezogen. Die Eigenschaft über welche der Primzahlbegriff in der Vorlesung definiert wurde (p ist prim, falls nur 1 und p Teiler von p sind), wird zur Definition von irreduziblen Elementen benutzt.
Im Ring der ganzen Zahlen sind irreduzible Elemente und Primelemente gleich.

Beweis der Aufgabe: $\forall a, b : ka \equiv kb \pmod{m} \Leftrightarrow \exists q \in \mathbb{Z} : (ka - kb) = qm$ oder auch $m|(a-b)k$. Nach Voraussetzung folgt aus $m|(a-b)k$, dass $m|(a-b)$ gilt. (Ebenfalls nach Voraussetzung: $m \not| k$.) Nach dem obigen Korollar ist das schon äquivalent zu der Aussage: m ist prim. □

Aufgabe 2: Der ISBN Code besteht aus 10 Zeichen $Z_1Z_2 \ldots Z_9Z_{10}$, mit $Z_i \in \{0, \ldots, 9, X\}$, also den 10 Ziffern und einem weiteren Symbol, welches dem Wert 10 entspricht. Die ersten 9 Ziffern sind die Nummer des Buches, die letzte Ziffer ist die Prüfziffer. Die Prüfziffer wird so gewählt, dass $\sum_{j=1}^{10}(11-j)Z_j \equiv 0 \pmod{11}$ ist (vgl. den Eintrag zu „Internationale Standard-Buchnummer ISBN“ im Bronstein).

Das folgende Programm berechnet zuerst die Summe mit x an der unleserlichen Stelle. Dann wird das x bestimmt als 7 und eine Amazon-Suche ermittelt das gesuchte Buch.

```
ISBN     = [ 0, 5, 1, x, 1, 4, 9, 2, 5, 7];  \\ letzte Ziffer ist Prüfziffer
Gewichte= vector(10,i,11-i);                 \\ die Gewichte 10,9,...,2,1

print( Gewichte*ISBN~ ); \\ Skalar-Produkt dieser "Matrizen", das ist die gewichtete Summe
                         \\ ISBN~ ist die transponierte Matrix von ISBN, also ein Spaltenvektor
/*
        7*x + 138
    Soll mod 11 == 0 sein, also in Pari-Objekten:
       Mod(7,11)*x*Mod(1,11)+Mod(138,11) = 0
    => x = -Mod(138,11) * Mod(7,11)^(-1)
          oder auch x = Mod(-138/7, 11)
*/

print("x= ", -Mod(138,11)*Mod(7,11)^-1 );
/*
  x= Mod(7, 11),
  Also die unleserliche Stelle ist 7,
```

```
   Amazon Suche:

     The Ultimate Hitchhiker's Guide to the Galaxy, A Trilogy in Five Parts
     (Gebundene Ausgabe) von Douglas Adams
*/
```

Aufgabe 3:

```
/* Die Anzahl der Dukaten erfüllt diese Bedingungen :

Dukaten = 17*q_1 + 3  oder  Dukaten kongruent 3 mod 17
Dukaten = 11*q_2 + 4  oder  Dukaten kongruent 4 mod 11
Dukaten =  6*q_3 + 5  oder  Dukaten kongruent 5 mod  6

Mit den Symbolen des chinesischen Restsatzes aus der Vorlesung:
    r = 3
    x = Dukaten
    a1 = 3, m1 = 17
    a2 = 4, m2 = 11
    a3 = 5, m3 =  6

Das ist in Pari: x kongruent zu Mod(a_i, m_i), i=1,2,3.
*/

print( chinese([Mod(3,17),
                Mod(4,11),
                Mod(5,6)])  );

/*  Mod(785, 1122)

    Also sind 785 + n * 1122 , (n >=0) mindestens jedoch 785  Dukaten im Topf.
*/
```

Aufgabe 4:

```
{
  for(x=0,1000,
    if(Mod(x^2+25*x+1,55)==0, print(x," \t", Mod(x,55)););
  );
}

/*
\\ Ausschnitt der Ausgabe:
2       Mod(2, 55)
13      Mod(13, 55)
17      Mod(17, 55)
28      Mod(28, 55)
57      Mod(2, 55)
68      Mod(13, 55)
72      Mod(17, 55)
83      Mod(28, 55)
112     Mod(2, 55)
123     Mod(13, 55)
127     Mod(17, 55)
138     Mod(28, 55)

Es gibt mod 55 die vier Lösungen:

        Mod( 2, 55),    Mod(13, 55),
        Mod(17, 55),    Mod(28, 55)
*/
```

Es gilt der Satz: Sei f ein Polynom mit ganzzahligen Koeffizienten. Wenn $a \equiv a' \pmod{m}$ ist, dann ist auch $f(a) \equiv f(a') \pmod{m}$.
(Das folgt aus wiederholter Anwendung von: $a \equiv a' \pmod{m}$ und $b \equiv b' \pmod{m} \Rightarrow ab \equiv a'b' \pmod{m}$ und $a + b \equiv a' + b' \pmod{m}$.)

In der obigen Schleife wäre es also ausreichend gewesen, x nur von 0 bis 54 laufen zu lassen. Es ist klar, dass alle weiteren $55 \leq x \leq 1000$ in den bereits gefundenen Restklassen liegen.

Alternative

Man bestimmt die Lösungen von $x^2 + 25x + 1 \equiv 0 \pmod{5}$ und $x^2 + 25x + 1 \equiv 0 \pmod{11}$ z.B. durch Probieren als $x_1 := 2 \pmod{5}$, $x_2 := 3 \pmod{5}$ und $y_1 := 2 \pmod{11}$, $y_2 := 6 \pmod{11}$.

Nun sucht man mittels Chinesischem Restsatz Lösungen $a, b, c, d \bmod 55$, so dass $a \equiv x_1 \pmod{5}$, $a \equiv y_1 \pmod{11}$ und $b \equiv x_1 \pmod{5}$, $b \equiv y_2 \pmod{11}$ und $c \equiv x_2 \pmod{5}$, $c \equiv y_1 \pmod{11}$ und $d \equiv x_2 \pmod{5}$, $d \equiv y_2 \pmod{11}$.

Aufgabe 5:

```
Count(p)={
    local(counter);

    if(!isprime(p), error(p," ist keine Primzahl, deswegen wird die Berechnung abgebrochen."));

    for(x=0, p-1,
        for(y=0, p-1,
            if(Mod(x,p)^2 + Mod(x*y,p) + Mod(y,p)^2 == Mod(0,p),
                counter++;
            );
        );
    );

    return(counter);
}

printTabRow(p)=
{
   print("p: ",p,"\t Count(p) = ", Count(p), "\t  ", lift(Mod(p,3)), "    ",lift(Mod(p,6)) );
}

{
    print("alle p: ");
    forprime(p=1,100,
        printTabRow(p);
    );
}

/*  Mal die einen, mal die anderen p ausgeben, um die p etwas klarer zu sehen: */
{   print();
    print("p=3*n+1:");
    forprime(p=1,100,
        n = Count(p);
        if( n== 2*p-1, printTabRow(p));
        \\ ------------------------------>   p = 3*n +1
    );

    print();
    print("p=3*n-1:");
    forprime(p=1,100,
        n = Count(p);
        if( n== 1, printTabRow(p););
```

```
            \\ ------------------------------>    p= 3*n - 1
        );
}

/*
     Vermutung:
     ----------

          Count(3) = 3
          Count(p) = 1       , falls p kongruent -1 mod 3, deckt auch p=2 ab
          Count(p) = 2*p - 1, falls p kongruent  1 mod 3
*/


/*
alle p:
p: 2      Count(p) = 1      2    2
p: 3      Count(p) = 3      0    3
p: 5      Count(p) = 1      2    5
p: 7      Count(p) = 13     1    1
p: 11     Count(p) = 1      2    5
p: 13     Count(p) = 25     1    1
p: 17     Count(p) = 1      2    5
p: 19     Count(p) = 37     1    1
p: 23     Count(p) = 1      2    5
p: 29     Count(p) = 1      2    5
p: 31     Count(p) = 61     1    1
p: 37     Count(p) = 73     1    1
p: 41     Count(p) = 1      2    5
p: 43     Count(p) = 85     1    1
p: 47     Count(p) = 1      2    5
p: 53     Count(p) = 1      2    5
p: 59     Count(p) = 1      2    5
p: 61     Count(p) = 121    1    1
p: 67     Count(p) = 133    1    1
p: 71     Count(p) = 1      2    5
p: 73     Count(p) = 145    1    1
p: 79     Count(p) = 157    1    1
p: 83     Count(p) = 1      2    5
p: 89     Count(p) = 1      2    5
p: 97     Count(p) = 193    1    1

p=3*n+1:
p: 7      Count(p) = 13     1    1
p: 13     Count(p) = 25     1    1
p: 19     Count(p) = 37     1    1
p: 31     Count(p) = 61     1    1
p: 37     Count(p) = 73     1    1
p: 43     Count(p) = 85     1    1
p: 61     Count(p) = 121    1    1
p: 67     Count(p) = 133    1    1
p: 73     Count(p) = 145    1    1
p: 79     Count(p) = 157    1    1
p: 97     Count(p) = 193    1    1

p=3*n-1:
p: 2      Count(p) = 1      2    2
p: 5      Count(p) = 1      2    5
p: 11     Count(p) = 1      2    5
p: 17     Count(p) = 1      2    5
p: 23     Count(p) = 1      2    5
p: 29     Count(p) = 1      2    5
p: 41     Count(p) = 1      2    5
```

```
105 p: 47     Count(p) = 1     2   5
    p: 53     Count(p) = 1     2   5
    p: 59     Count(p) = 1     2   5
    p: 71     Count(p) = 1     2   5
    p: 83     Count(p) = 1     2   5
110 p: 89     Count(p) = 1     2   5
    */
```

Dateianhänge[1]:

- Aufgabe2.gp
- Aufgabe3.gp
- Aufgabe4.gp
- Aufgabe5.gp

[1] Im Acrobat Reader anklickbar oder unter Linux mittels „pdftk Musterloesung.pdf unpackfiles“ extrahieren.