1. Wir rechnen in $(\mathbb{Z}_{35}, +, \cdot)$.

   a) Welche Elemente von $(\mathbb{Z}_{35}, +, \cdot)$ haben ein Invers?

   b) Man berechne das multiplicative Inverse von 8.

2. Man löse im Körper das lineare Gleichungssystem:

   a) $x + 3y = 1$

   b) $3x + y = 4$

3. Wir rechnen in $(\mathbb{Z}_p, +, \cdot)$, wobei p eine Primzahl größer 2 ist. Wir haben also einen Zahlenkörper. Man zeige:

   a) $x \neq 0 \Rightarrow x \neq -x$

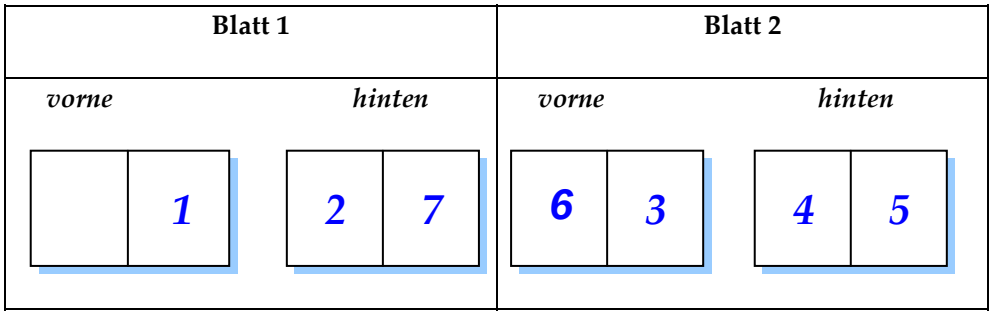
   b) 1 hat genau 2 Wurzeln, nämlich 1 und –1

   c) Wenn eine Zahl a 0 eine Wurzel x hat, dann haz sie genau 2 Wurzeln, nämlich x und $-x$

   d) Man finde in $(\mathbb{Z}_7, +, \cdot)$ alle Zahlen, die eine Wurzel haben.

4. **Druck einer Broschüre**

Wir wollen eine Broschüre aus Blättern erstellen, die wir in der Mitte falten. Ein Blatt wird vorne und hinten mit je zwei Seiten Text bedruckt. Die Reihenfolge beim Ausdruck kann deshalb nicht Seite 1, 2, 3, … sein. Zum Beispiel druckt man eine 7-seitige Broschüre so:



Unsere Aufgabe ist es, ein Programm zu entwickeln, das als Eingabe die Anzahl der Seiten einer Broschüre erwartet und die richtige Verteilung der Seiten auf die Blätter ausgibt. *Eingabe:* In der Datei *broschuere.in* befinden sich mehrere natürliche Zahlen *n* aus dem Intervall [1, 220], die die Anzahl der Seiten darstellen. *Ausgabe:* In *broschuere.out* werden die Informationen für den Ausdruck geschrieben, wie im Beispiel:

broschuere.in	**broschuere.out**
7 18 4 1	Anordnung fuer 7 Seiten: Seite 1 vorne: leer, 1 Seite 1 hinten: 2, 7 Seite 2 vorne: 6, 3 Seite 2 hinten: 4, 5 Anordnung fuer 18 Seiten: Seite 1 vorne: leer, 1 Seite 1 hinten: 2, leer Seite 2 vorne: 18, 3 Seite 2 hinten: 4, 17 Seite 3 vorne: 16, 5 Seite 3 hinten: 6, 15 Seite 4 vorne: 14, 7 Seite 4 hinten: 8, 13

	Seite 5 vorne: 12, 9 Seite 5 hinten: 10, 11 Anordnung fuer 4 Seiten: Seite 1 vorne: 4, 1 Seite 1 hinten: 2, 3 Anordnung fuer 1 Seiten: Seite 1 vorne: leer, 1 Seite 1 hinten: leer, leer

(ACM, South Central USA Regional, 1998)

5.

Korrekte Nachrichten

Um während einer Informationsübetragung in der EDV Fehler zu erkennen, wird meist das *CRC*-Verfahren (*Cyclic Redundancy Check*) eingesetzt. Die Daten werden in kleinen Paketen gesendet, und am Ende jedes Pakets fügt man zusätzliche Informationen (Prüfsummen) ein, die helfen sollen, Übertragungsfehler aufzuspüren.

Die Nachricht (das Paket) ist als eine lange binäre Zahl gegeben. Das erste Byte der Nachricht ist das Byte mit dem höchsten Stellenwert der binären Zahl. Das zweite Nachrichtenbyte hat den zweithöchsten Stellenwert usw. Wenn wir die Nachricht mit *m* bezeichnen, dann müssen wir nun *m2* anstatt *m* übertragen, wobei *m2* die Nachricht *m*

und zwei Bytes für die Prüfsumme beinhaltet.

Der *CRC*-Wert ist so gewählt, dass die Division von *m2* durch einen bestimmten 16-Bit-Wert *g* (Generatorwert) den Rest 0 ergibt. Dadurch kann der Empfangsprozess prüfen, ob die Nachricht fehlerfrei übertragen wurde. Er dividiert einfach die empfangenen Daten durch *g*. Wenn der Rest 0 ist, nimmt er an, dass die Übetragung korrekt erfolgt ist.

Unsere Aufgabe ist es ein Programm zu implementieren, das den *CRC*-Wert einer zu sendenden Nachricht berechnet. Wir nehmen als Generatorwert *g*=34943. Das Programm wird die Zeilen aus der Eingabedatei lesen und für jede von ihnen den zwei Byte langen *CRC*-Wert berechnen und ihn danach, repräsentiert durch zwei Hexadezimalzahlen, in die Ausgabedatei schreiben. Jede Eingabezeile beinhaltet maximal 1024 *ASCII*-Zeichen. Bemerkung*:* Jeder ausgegebene *CRC*-Wert liegt zwischen 0 und 34942 (dezimal). *Eingabe:* In der Eingabedatei *crc.in* finden sich mehrere String-Nachrichten, eine auf jeder Zeile. *Ausgabe:* Schreiben Sie in die Datei *crc.out* für jede Eingabezeile eine Zeile mit den hexadezimal dargestellten Prüfsummen-Bytes, wie im Beispiel:

crc.in	**crc.out**
Alle meine Froeschlein	5E 4A
Huepfen auf und ab	78 D6
Schrein dabei recht lustig	F BC
Quack, quack, quack, quack, quack.	56 26

`--------------------------------`	`27 43`
`Es ist Unsinn`	`2C 86`
`sagt die Vernuft`	`15 AB`
`Es ist was es ist`	`3E  E`
`Sagt die... .`	`35 C5`
`--------------------------------`	`27 43`

(ACM, New Zealand Contest, 1989, modifiziert)

Literatur

1. Albrecht Beutelspacher, Marc-Alexander Zschiegner, *Diskrete Mathematik für Einsteiger. Mit Anwendungen in Technik und Informatik*, 3. Auflage, Vieweg Verlag, 2007.

2. Doina Logofătu, *Algorithmen und Problemlösungen mit C++*, Vieweg+Tebuner Verlag, 2010.