Humboldt-Universität zu Berlin
Institut für Informatik
Prof. Dr. Jens Redlich
M. Kabierski, Dr. W. Müller, L. Pfahlsberger, M. Sänger

Berlin, den 16.01.2023

Übungen zur Vorlesung
Grundlagen der Programmierung

Übungsblatt 6

Abgabe: Die Abgabe Ihrer Lösungen erfolgt über den Moodle-Kurs der Vorlesung (https://moodle.hu-berlin.de/enrol/index.php?id=114664). Nutzen Sie die im Abschnitt „Übung“ angegebene **Aufgabe 6.** Die Abgabe ist bis zum **30.01.2023** um **09:15 Uhr** möglich. **Die Abgabe erfolgt in Gruppen.** Bitte bilden Sie im Moodle-Gruppen! Bitte verwenden Sie **keine** (noch nicht in der Vorlesung eingeführten) Java-Bibliotheken.

Hinweis: Testen Sie Ihre Lösung bitte auf einem Rechner aus dem Informatik Computer-Pool z.B. `gruenau6.informatik.hu-berlin.de` (siehe auch Praktikumsordnung: Referenzplattform).

Aufgabe 1 (Zahlensysteme) — 10 Punkte

Wandeln Sie die folgenden Zahlendarstellungen um:

1) Basis 10 (Dezimal) ➔ Basis 15

2023_{10}

2) Basis 5 ➔ Basis 16

42021_{5}

3) Basis 9 ➔ Basis 3

2023_{9}

4) Basis 10 (Dezimal) mit Vorzeichen ➔ **16-Bit Binärzahl im Zweierkomplement**

-2023_{10}

5) 3-Byte hexadezimalen Zweierkomplement ➔ **Dezimalzahl mit Vorzeichen**

$FFF819_{16}$

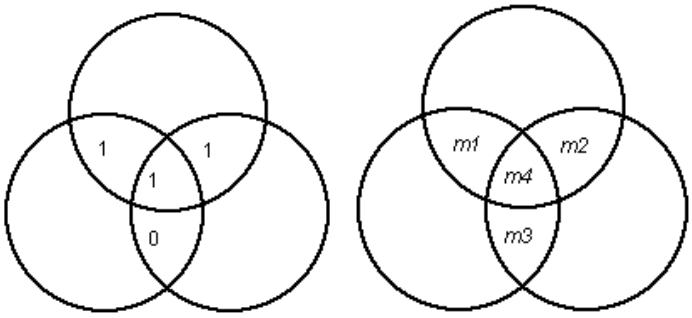
Geben Sie Ihre Lösungen als **UTF8-Textdatei A1.txt** ab. Ihre Antworten sollen hierbei, wie im folgenden Beispiel formatiert sein:

```
1: F41_15
2: AF1_16
3: 120012_3
4: 0101010101010101_2
5: +42_10
```

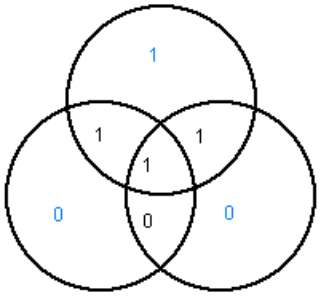
Aufgabe 2 (Hamming Codes in TOY) 10 Punkte

Fehlerkorrekturcodes ermöglichen es Daten, die über einen gestörten Kommunikationskanal übertragen werden, fehlerfrei zu empfangen. Um dies zu erreichen, fügt der Absender redundante Informationen zu der Nachricht hinzu, sodass, selbst wenn ein Teil der Ausgangsdaten bei der Übertragung beschädigt wird, der Empfänger immer noch die ursprüngliche Nachricht empfangen kann. Übertragungsfehler sind häufig und können z. B. bei der drahtlosen Kommunikation durch atmosphärische Störung auftreten. In einer gestörten Umgebung können Fehlerkorrekturcodes den Durchsatz einer Kommunikationsverbindung erhöhen, da keine Notwendigkeit besteht, die Nachricht erneut zu übertragen, wenn sie während der Übertragung teilweise beschädigt wird.

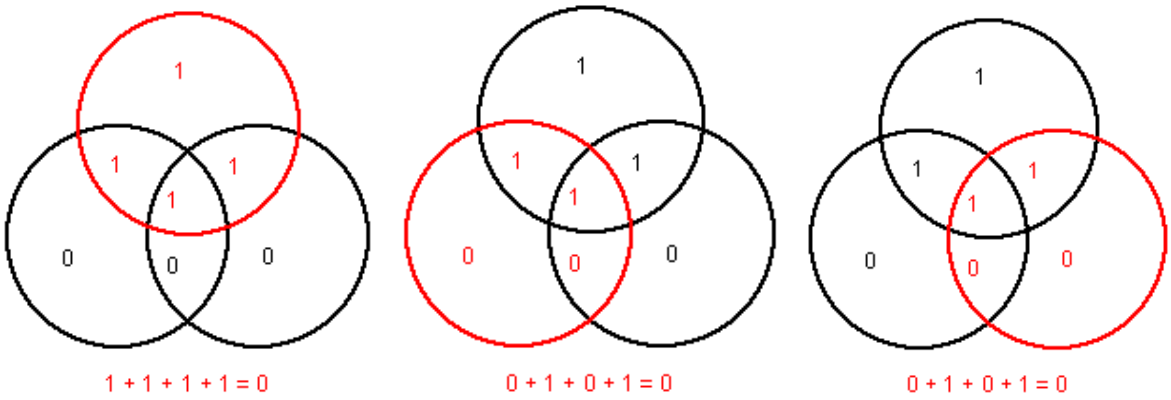
Ein Hamming-Code ist eine bestimmte Art von Fehlerkorrekturcode, der die Erkennung und Korrektur von Einzel-Bit-Übertragungsfehlern ermöglicht. Hamming-Codes arbeiten, indem sie immer vier Nachrichten-Bits lesen (`m1`, `m2`, `m3` und `m4`) und dann drei Parity-Bits (`p1`, `p2` und `p3`) einfügen. Wenn eines dieser sieben Bits während der Übertragung beschädigt wird, kann der Empfänger den Fehler erkennen und die ursprünglichen vier Nachrichtenbits wieder rekonstruieren. Als Beispiel nehmen wir die Nachricht `1101`. Jedes der vier Nachrichten-Bits schreiben wir in einen bestimmten der vier Kreuzungsbereiche dreier paarweise überlappender Kreise, wie unten dargestellt:



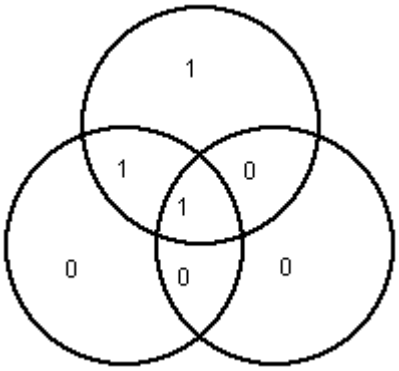
Der Hamming-Code fügt nun drei Parity-Bits hinzu, sodass jeder Kreis gerade Parität besitzt, d.h. die Summe der vier Bits in jedem Kreis ist gerade.



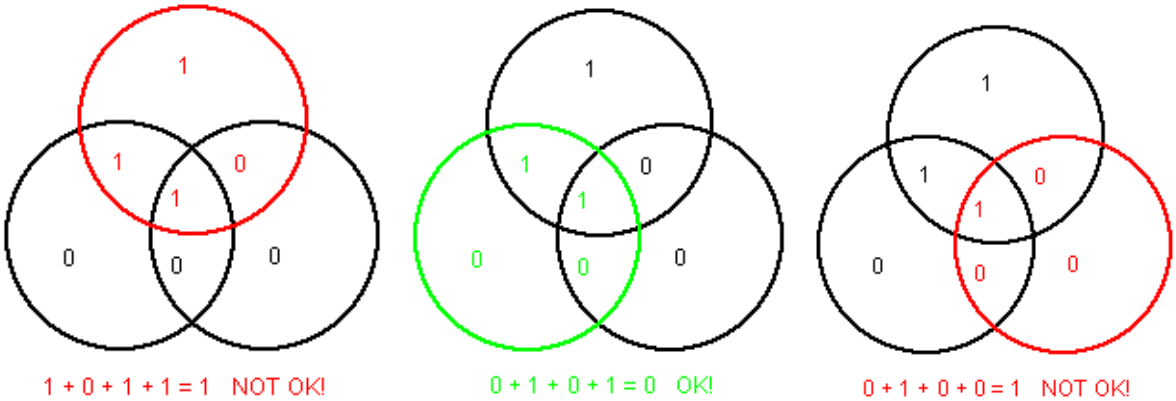
In diesem Fall senden wir `1101100` da die drei Paritätsbits `1` (oben), `0` (links), und `0` (rechts) sind.



Nun stellen Sie sich vor, dieses Bild wird per WLAN über einen verrauschten Kommunikationskanal übertragen und dass dabei ein Bit beschädigt wird, sodass folgendes Bild bei der Empfangsstation ankommt (entsprechend `1001100`):



Der Empfänger erkennt durch Prüfen der Parität, dass ein Fehler in einem der drei Kreisen aufgetreten ist. Darüber hinaus kann der Empfänger bestimmen, wo der Fehler aufgetreten ist (das zweite Bit), den Fehler korrigieren und so die vier ursprünglichen Nachricht-Bits empfangen!



Da die Paritätsprüfung für den ersten Kreis und letzten Kreis gescheitert ist, aber der mittlere Kreis in Ordnung war, gibt es nur ein Bit, das verantwortlich sein kann und das ist `m2`. Wenn das mittlere Bit `m4` beschädigt wird, dann schlagen alle drei Paritätsprüfungen fehl. Wenn ein Paritäts-Bit selbst beschädigt ist, dann wird nur eine Paritätsprüfung fehlschlagen. Wenn die Datenverbindung so gestört ist, dass zwei oder mehr Bits gleichzeitig beschädigt werden, wird unsere Codierung nicht funktionieren.

Teil 1 (5 Punkte): Schreiben Sie ein TOY-Programm `encode.toy` das eine binäre Nachricht mit dem oben beschriebenen Schema verschlüsselt. Ihr Programm sollte wiederholend vier Bits `m1`, `m2`, `m3` und `m4` von der TOY-Standardeingabe lesen und die sieben Bits `m1`, `m2`, `m3`, `m4`, `p1`, `p2`, `p3` auf der TOY-Standardausgabe ausgeben, wobei:

- `p1 = m1 ^ m2 ^ m4`
- `p2 = m1 ^ m3 ^ m4`
- `p3 = m2 ^ m3 ^ m4`

Hinweis: `^` ist der XOR-Operator in Java und TOY. Dies entspricht dem oben beschriebenen Paritätsbegriff.

Teil 2 (5 Punkte): Schreiben Sie ein TOY-Programm `decode.toy`, welches eine Hamming-codierte Nachricht korrigiert. Ihr Programm sollte wiederholt sieben Bits `m1`, `m2`, `m3`, `m4`, `p1`, `p2`, `p3` von der TOY-Standard-Eingabe lesen und die vier Nachrichten-Bits auf der TOY-Standardausgabe ausgeben.

Um zu bestimmen, ob und welche, der Nachrichten-Bits überhaupt beschädigt sind, führen Sie die folgenden drei Paritätsprüfungen durch:

(1) `p1 = m1 ^ m2 ^ m4`
(2) `p2 = m1 ^ m3 ^ m4`
(3) `p3 = m2 ^ m3 ^ m4`

Das Ergebnis ist wie folgt zu interpretieren:

- Wenn genau eine oder keine der Paritätsprüfungen versagen, dann sind alle vier Nachrichten-Bits korrekt.
- Wenn Kontrollen 1 und 2 fehlschlagen (aber nicht 3), so ist das Bit `m1` falsch.
- Wenn Kontrollen 1 und 3 fehlschlagen (aber nicht 2), so ist das Bit `m2` falsch.
- Wenn Prüfungen 2 und 3 fehlschlagen (aber nicht 1), so ist das Bit `m3` falsch.
- Wenn alle drei Prüfungen fehlschlagen, so ist das Bit `m4` falsch.

Wenn nötig, korrigieren (d.h. invertieren) Sie die beschädigten Nachrichten-Bits `m1`, `m2`, `m3` und `m4`.

Der Einfachheit halber werden wir jedes Bit einzeln (`0000` oder `0001`) übertragen. Ihr Programm sollte das Einlesen wiederholen, bis `FFFF` auf TOY-Standardeingabe gelesen wird. Die Eingabedateien für `encode.toy` und `decode.toy` sehen bspw. wie folgt aus:

```
encode.toy input file          decode.toy input file
---------------------          ---------------------------------
0001 0001 0000 0001            0001 0000 0000 0001 0001 0000 0000
0001 0001 0001 0000            0000 0001 0001 0000 0000 0000 0000
0001 0001 0001 0001            0001 0001 0001 0001 0001 0001 0000
FFFF                           FFFF
```

Sie können davon ausgehen, dass die Anzahl der übertragenen Bits (ohne das abschließende Wort) ein Vielfaches von vier bei der Codierung, und ein Vielfaches von sieben bei der Korrektur ist.

Laden Sie die Dateien: `encode.toy` und `decode.toy` über Moodle hoch.

Hinweise zur Bearbeitung:

- Zur Prüfung Ihrer Lösung können Sie das in Moodle bereitgestellte zip-Archiv *hamming.zip* und die darin enthaltenen Programme `HammingEncoder.java` und `HammingDecoder.java` nutzen. Diese implementieren das En- und Dekodieren von Nachrichten nach dem Hamming-Code-Verfahren in Java. Ferner finden Sie Eingabedateien mit Beispielen für das En- und Dekodieren von Nachrichten. Zur Verwendung dieser Materialien können Sie bspw. folgende Aufrufe nutzen:

  ```
  javac HammingEncoder.java
  java HammingEncoder < input_encoding_1.txt

  javac HammingDecoder.java
  java HammingDecoder < input_decoding_1.txt
  ```

  Die Programme erzeugen das gleiche Ausgabeformat wie die Toy-Maschine, d.h. jedes Ausgabebit wird als vierstellige Binärzahl in einer separaten Zeile ausgeben. Sie können die Java-Programmausgabe mit der Ausgabe Ihres Toy-Programms abgleichen:

  ```
  javac Toy.java
  java Toy encode.toy < input_encoding_1.txt
  java Toy decode.toy < input_decoding_1.txt
  ```

  Für eine bessere Lesbarkeit der Ausgabe der Java-Implementierung finden Sie in den Quelltextdateien auch die Möglichkeit die Eingabebeispiele in einer Zeile auszugeben. Kommentieren Sie diese ein (und die vorgegebene Ausgabevariante aus) und kompilieren Sie das Programm neu, um diese zu erhalten.

- Hinweise zur Verwendung der grafischen Toy-Maschine und der Kommandozeilenversion finden Sie im Moodle-Kurs unter den Punkten FAQ - Toy Machine.

Viel Erfolg beim Lösen der Aufgaben!