Berner Fachhochschule

Introduction to Computer Science

Exercise 3

1. Binary to integer

Calculate the value of the binary number

0 0000000 01010010 → + 64 + 16 + 2 = 82
0 0000010 10000011 → + 512 + 128 + 2 + 1 = 643

2. Integers to binary

Calculate the value (decimal system) of the decimal numbers

316 → 256 + 32 + 16 + 8 + 4 = 0 0000001 00111100
1035 → 1024 + 8 + 2 + 1 = 0 0000100 00001011

3. Number Types

The input on the left produces the following output:

```
int a = 62352;
int b = 1142342;
int c = a * b;
double d = 1.0 * a * b;
print(a);
print(b);
print(c);
print(d);
double e = 1 - 0.9 - 0.1;
print(e);
```

```
a =     62352
b =     1142342

c =     -1787135648
d =     7.1227308384 E10
e =     2.2351741790771484 E-8
```

Explain the output.

- ➔ The product of the two numbers a and b overflows the int range, so the sign bit is set to 1, which is interpreted as negative number.
- ➔ Changing the result type to double (by multiplying by 1.0) results in a 64 bit double number, which outputs the approximated (more or less correct) result.
- ➔ As some numbers like 0.1 cannot be saved correctly as binary numbers, the subtraction of 1 – 0.9 – 0.1 doesn't result in the value 0, but a small error.

4. XML vs. JSON

```
<TeamMeeting>
        <date>2021-12-20</date>
        <time>09:15</time>
        <location>Room 214</location>
        <participant>Rolf</participant>
        <participant>Kurt</participant>
        <participant>Vera</participant>
 </TeamMeeting>
```

```
{
  "TeamMeeting": {
      "date": "2021.12.20"
      "time": "09:15"
      "location": "Room 214"
      "participants": [ "Rolf", "Kurt", "Vera"]
   }
}
```

5. Data Chaos

➔ If we don't know the original texts encoding (ASCII, UTF, ISO-Latin-xxx, …) or character set used, we are not able to decode the "source-code" correctly.

➔ We can either ask for the original character encoding and hope to be able to convert the source code to a well-known encoding, or we ask the college to save the file in an encoding we can read without error.