Introduction to Computer Science
Constructor University
Dr. Jürgen Schönwälder

Module: CH-232
Date: 2023-10-06
Due: 2023-10-13

Problem Sheet #5

Problem 5.1: *b-complement* (1+1+1 = 3 points)

We use a fixed size b-complement number system with the base $b = 3$ and $n = 6$ digits (b3n6).

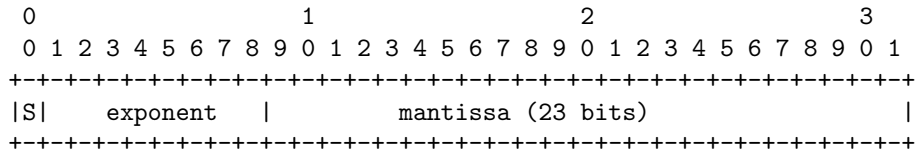
a) What are the smallest and the largest numbers that can be represented and why?

b) What is the representation of -1 and -99 in b3n6?

c) Add the numbers -1 and -99 in b3n6. What is the result in b-complement representation? Convert the result from b-complement representation back into the decimal number system.

Problem 5.2: *IEEE 754 floating point numbers* (4+1 = 5 points)

IEEE 754 floating point numbers (single precision) use the following format (the numbers on the top of the box indicate bit positions, the fields in the box indicate what the various bits mean).



The encoding starts with a sign bit, followed by the biased exponent (8 bits), followed by the mantissa (23 bits). For single-precision floating-point numbers, the exponents in the range of -126 to +127 are biased by adding 127 to get a value in the range 1 to 254 (0 and 255 have special meanings).

a) Explain step by step how the decimal fraction 321.123 is converted into a single precision floating point number.

b) What is the decimal fraction that is actually stored in the single precision floating point number and what is the absolute error?

Problem 5.3: *decimal to binary and binary to decimal (haskell)* (1+1 = 2 points)

Implement a functions to convert non-negative integer numbers into binary notation and back.

a) Implement a function `dtob :: Word -> String` converting a non-negative integer number into a `String` (consisting of the characters `'0'` and `'1'`) representing the decimal number as a binary number.

b) Implement a function `dtob :: String -> Word` converting a `String` (consisting of the characters `'0'` and `'1'`) representing a binary number into the corresponding non-negative integer number. It is not necessary to handle unexpected strings in a meaningful way.

Submit your Haskell code as a plain text file. Below is a template file with a few unit test cases.

```
1  module Main (main) where
2
3  import Data.Word
```

```haskell
import Test.HUnit

-- | Convert a non-negative integer number into a String providing a
--   binary representation of the number.
dtob :: Word -> String
dtob _ = undefined

-- | Convert a String representing a non-negative integer number as a
--   binary number into a non-negative integer number.
btod :: String -> Word
btod _ = undefined

-- Below are some test cases...

dtobTests = TestList [ dtob 0 ~?= "0"
                     , dtob 1 ~?= "1"
                     , dtob 2 ~?= "10"
                     , dtob 127 ~?= "1111111"
                     , dtob 12345 ~?= "11000000111001"
                     ]

btodTests = TestList [ btod "0" ~?= 0
                     , btod "1" ~?= 1
                     , btod "10" ~?= 2
                     , btod "1111111" ~?= 127
                     , btod "11000000111001" ~?= 12345
                     ]

main = runTestTT $ TestList [ dtobTests, btodTests ]
```