Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH-232-A
Date: 2021-10-08
Due: 2021-10-15

ICS 2021 Problem Sheet #5

Problem 5.1: *base b numbers closed form* (1+2 = 3 points)

Consider a base b number system ($b > 1$) with n digits and the number $1 \ldots 1_b$ (n times the digit 1).

a) Define a sum formula that provides the value of the n-digit number $1 \ldots 1_b$.

b) Proof via induction that the value of the n-digit number $1 \ldots 1_b$ is given by the following closed form:

$$\frac{1 - b^n}{1 - b}$$

For example, the 4-digit base 5 number 1111_5 has the value $\frac{1-5^4}{1-5} = \frac{-654}{-4} = 156$.

Problem 5.2: *unicode and utf-8 encoding* (1+2+1 = 4 points)

The content of a file containing UTF-8 Unicode encoded text is given by the following sequence of bytes in hexadecimal notation:

```
48 65 6c 6c 6f 20 f0 9f 8c 8d 21 0a
```

a) Write each byte in binary notation.

b) Identify the unicode code points of the characters. What is the text stored in the file?

c) Which line end convention is used? What are other popular line end conventions?

Problem 5.3: *long years (haskell)* (1 point)

According to the ISO8601 calendar, most years have 52 weeks, but some have 53 weeks. These are so called long years. There is a relatively simple way to calculate whether a given year y is a long year. The function $w(y)$ determines with the helper function $p(y)$ the number of weeks in the year y. (Note that the functions use integer division.)

$$p(y) = (y + \frac{y}{4} - \frac{y}{100} + \frac{y}{400}) \bmod 7$$

$$w(y) = 52 + \begin{cases} 1 & p(y) == 4 \wedge p(y-1) == 3 \\ 0 & \text{otherwise} \end{cases}$$

Implement a Haskell function `isLongYear :: Int -> Bool` to determine whether a year is a long year. Use the `isLongYear` function to calculate all long years in the range 2000..2100.

Submit your Haskell code as a plain text file.

Problem 5.4: *decimal to binary and binary to decimal (haskell)* (1+1 = 2 points)

Implement a function to convert a decimal number into a binary notation and one function to convert from a binary notation back.

a) Implement a function `dtob :: Int -> String` that converts a non-negative integer number into a `String` (consisting of the characters `'0'` and `'1'`) representing the integer number as a binary number. It is not necessary to handle negative integers in a meaningful way.

b) Implement a function `dtob :: String -> Int` that converts 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 Test.HUnit
4
5  -- |The 'dtob' function converts a non-negative integer number into a
6  -- String providing a binary representation of the number.
7  dtob :: Int -> String
8  dtob _ = undefined
9
10 -- |The 'btod' function converts a String representing a non-negative
11 -- integer number as a binary number into an integer number.
12 btod :: String -> Int
13 btod _ = undefined
14
15 {-
16    Below are some test cases.
17 -}
18
19 dtobTests = TestList [ dtob 0 ~?= "0"
20                       , dtob 1 ~?= "1"
21                       , dtob 2 ~?= "10"
22                       , dtob 127 ~?= "1111111"
23                       , dtob 12345 ~?= "11000000111001"
24                       ]
25
26 btodTests = TestList [ btod "0" ~?= 0
27                       , btod "1" ~?= 1
28                       , btod "10" ~?= 2
29                       , btod "1111111" ~?= 127
30                       , btod "11000000111001" ~?= 12345
31                       ]
32
33 main = runTestTT $ TestList [ dtobTests, btodTests ]
```