Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH-232-A
Date: 2019-09-20
Due: 2019-09-27

ICS 2019 Problem Sheet #2

Even though some proofs may look simple or obvious, make sure you work out the proof details correctly and you write things down using correct mathematical notation. Construct a proof of the requested type.

Problem 2.1: *proof by contrapositive* (2 points)

Let n be a natural number. If n is not divisible by 3, then n is also not divisible by 15.

Problem 2.2: *proof by induction* (4 points)

Let n be a natural number with $n \geq 1$. Prove that the following holds:

$$1^2 + 3^2 + 5^2 + \ldots (2n-1)^2 = \sum_{k=1}^{n} (2k-1)^2 = \frac{2n(2n-1)(2n+1)}{6}$$

Problem 2.3: *leap year in the Gregorian calendar (haskell)* (1+1 = 2 points)

In the Gregorian calendar, a leap year occurs if (i) the year is a multiple of four and (ii) the year is not divisible by 100 or (iii) the year is divisible by 400. Note that (ii) and (iii) overlap but (iii) takes precedence. Write a Haskell function `isLeapYear` to determine whether a year is a leap year or not.

a) Write a `isLeapYear` function using a Boolean expression involving the Boolean operators `&&` (and), `||` (or), and the Boolean function `not`.

b) Write a `isLeapYear'` function using guards and without any usage of the Boolean operators `&&` (and), `||` (or), and the Boolean function `not`.

The Haskell function `div` returns how many times the first number can be divided by the second one and the function `mod` returns the remainder of an integer division.

Explain how you have tested your `isLeapYear` and `isLeapYear'` functions.

Problem 2.4: *rotate a list and produce all possible rotations of a list (haskell)* (1+1 = 2 points)

a) Using pattern matching, implement a recursive function `rotate :: Int -> [a] -> [a]`, which left rotates the list given as the second argument by the number of positions indicated by the first argument. Below are some example evaluations of the `rotate` function:

```
> rotate 0 "abcdef"
"abcdef"
> rotate 1 "abcdef"
"bcdefa"
> rotate 7 "abcdef"
"bcdefa"
> rotate 7 ""
""
```

b) Using your `rotate` function, implement a function `circle :: [a] -> [[a]]`, which takes a list and returns a list of all possible rotations of the list. Below are some example evaluations of the `circle` function:

```
> circle ""
[]
> circle "a"
["a"]
> circle "ab"
["ab","ba"]
> circle "abc"
["abc","bca","cab"]
```

Hint: Consider producing a list of the approriate size and then apply the `rotate` function to the elements of this list in order to produce the result. This can result in a very short functional solution. Another approach is to implement a helper function the produces the nth result list element and to call this helper function successively to produce the result list.

Submit your Haskell source code as a plain text file.