Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH-232-A
Date: 2020-09-25
Due: 2020-10-02

ICS 2020 Problem Sheet #3

Problem 3.1: *cartesian products* (1+1 = 2 points)

Prove or disprove the following two propositions:

a) $(A \cap B) \times (C \cap D) = (A \times C) \cap (B \times D)$

b) $(A \cup B) \times (C \cup D) = (A \times C) \cup (B \times D)$

Problem 3.2: *reflexive, symmetric, transitive* (3 points)

For each of the following relations, determine whether they are reflexive, symmetric, or transitive. Provide a reasoning.

a) $R = \{(a,b) | a,b \in \mathbb{Z} \wedge |a-b| \leq 3\}$
(The absolute difference of the numbers a and b is less than or equal to 3.)

b) $R = \{(a,b) | a,b \in \mathbb{Z} \wedge (a \bmod 10) = (b \bmod 10)\}$
(The last digit of the decimal representation of the numbers a and b is the same.)

Problem 3.3: *proof by induction* (1+2 = 3 points)

Consider the two Haskell functions `cnt` and `con` defined below.

```
cnt :: Eq a => a -> [a] -> Int
cnt x [] = 0
cnt x (y:ys)
    | x == y    = 1 + (cnt x ys)
    | otherwise = cnt x ys

con :: [a] -> [a] -> [a]
con [] ys     = ys
con (x:xs) ys = x : (con xs ys)
```

Proof by induction over s that `cnt x (con s t) == (cnt x s) + (cnt x t)` holds.

Problem 3.4: *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 possible number of rotations 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 n-th 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.