Introduction to Computer Science
Constructor University
Dr. Jürgen Schönwälder

Module: CH-232
Date: 2024-01-08
Due: 2024-01-15

Problem Sheet #13

This sheet is only for students who failed to obtain the module achievement.

Problem 13.1: *sum formula* (2 points)

Prove that $1 + 4 + \ldots + (3n - 2) = \frac{1}{2}n(3n - 1)$ for $n \in \mathbb{N}$ and $n > 0$.

Problem 13.2: *equivalence relation* (2 points)

Let $A = \mathbb{N}_+ \times \mathbb{N}_+$ be the set of pairs of positive natural numbers. Let $\sim \subseteq A \times A$ be a binary relation on A. The tuple $((a, b), (c, d))$ is an element of $\sim$ if and only if $ad = bc$ (the product of a and d is equal to the product of b and c).

Show that $\sim$ is an equivalence relation (i.e., $\sim$ is reflexive, symmetric and transitive). For each property, first state what you are trying to show before you provide the argument.

Problem 13.3: *not-or is a universal boolean function* (3 points)

Prove that not-or ($\overline{\vee}$) is a universal boolean function by showing how $\overline{\vee}$ functions can implement the classic universal Boolean functions $\wedge, \vee, \neg$.

Problem 13.4: *bnf grammar reduction* (2 points)

Let $\Sigma = \{0, 1, \ldots, 9, x, y, z, +, *, (,)\}$. Consider the following grammar in Backus Naur Format:

```
<expression> ::= <term> | <expression> "+" <term>
<term>       ::= <factor> | <term> "*" <factor>
<factor>     ::= <constant> | <variable> | "(" <expression> ")"
<variable>   ::= "x" | "y" | "z"
<constant>   ::= <digit> | <digit> <constant>
<digit>      ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

a) Use the grammar to reduce the expression `42 + 8 * x` to the start symbol. Show each step of your derivation.

b) Show four different examples of syntactically invalid expressions and describe which grammar rules are detecting the errors.

Problem 13.5: *divisors in haskell* (1 point)

Write a function `divisors :: Int -> [Int]` that returns the list of proper divisors of a given positive integer `x`. The result of `divisors x` includes 1, but not the number `x` itself. For example:

```
Prelude> divisors 6
[1,2,3]
Prelude> divisors 12
[1,2,3,4,6]
Prelude> divisors 15
[1,3,5]
Prelude> divisors 1
[]
Prelude> divisors 2
[1]
```

Recall that the Haskell function `div` gives you the result of an integer division (truncated toward negative infinity) and the function `mod` gives you the integer modulus (remainder of an integer division).

Problem 13.6: *folds in haskell* (1 point)

Consider the following function definition:

```haskell
m f xs = foldr g [] xs
  where g y ys = (f y) : ys
```

How is the expression `m (*2) [1..3]` evaluated? Explain step-by-step how the expression is expanded and how the result is produced. Describe what the function `m` is doing, i.e., to which standard Haskell function it relates.