ICS 2022 Problem Sheet #4

Problem 4.1: *sets and relations in a coffee bar*

A coffee bar is offering different kinds of coffee to their customers. Customers order their coffee at the cashier, who then notifies the barrister about the order so that she can brew the coffee requested by a customer. A customer may have to wait until the coffee is ready. During this time the cashier may accept additional orders. Customers may place a gratuity, which are shared among all employees. The distribution of the tips is organized by the barrister since she owns the coffee shop.

- Identify at least five sets (entities) that play a role in the coffee bar scenario. Introduce suitable notation.
  - Entities:
    - C "the set of customers"
    - X "the set of coffees"
    - B "the set of barristers"
    - O "the set of orders"
    - E "the set of employees"
    - CS "the set of coffee shops"
    - P "the set of persons"
    - . . .
- Identify at least five relations between the sets (entities) that you have identified. Define the relations using suitable mathematical notation.
  - Relations:
    - $EWICS \subseteq (E \times CS)$ "employee works in a coffee shop"
    - $EWWE \subseteq (E \times E)$ "employee works with some other employee"
    - $CIP \subseteq (C \times P)$ "customer is a person"
    - $EIP \subseteq (E \times P)$ "employee is a person"
    - $BOCS \subseteq (B \times CS)$ "barrister owns the coffee shop"
    - $CAO \subseteq (C \times O)$ "cashier accepts orders"
    - . . .
- Identify at least five endorelations including at least one equivalence relation, one partial order relation, and one strict partial order relation. Define the relations using suitable mathematical notation. Try to cover different sets (entities)
  - Endorelations:
    - $EWW \subseteq (E \times E)$ "an employee works with some other employee"
      a ≡_{EWW} b for $(a, b) \in EWW \implies$ EWW is an equivalence relation
    - $COB \subseteq (X \times X)$ "a coffee is ordered before some other coffee"
      a ≺_{COB} b for $(a, b) \in COB \implies$ COB is a strict partial order relation
    - $COA \subseteq (C \times C)$ "a customer orders after some other customer"
      a ≺_{COA} b for $(a, b) \in COA \implies$ COA is a strict partial order relation
    - $CWAL \subseteq (E \times E)$ "a customer waits as long as some other customer"
      a ≼_{CWAL} b for $(a, b) \in CWAL \implies$ CWAL is a partial order relation
    - $EGAM \subseteq (E \times E)$ "an employee gets as many (tips) as some other employee"
      a ≼_{EGAM} b for $(a, b) \in EGAM \implies$ EGAM is a partial order relation
    - . . .

Problem 4.2: *function composition*

Given the functions f(x) = x + 1. g(x) = 2x, and h(x) = x^2 , determine an expression for the following function compositions:

a) f ∘ g = 2x +1

b) f ∘ h = x^2 +1

c) g ∘ f = 2(x + 1) = 2x + 2

d) g ∘ h = $2x^2$

e) h ∘ f = $(x+1)^2 = x^2 + 2x + 1$

f) h ∘ g = $(2x)^2 = 4x^2$

g) f ∘ (g ∘ h) = $2x^2 + 1$

h) h ∘ (g ∘ f) = (2x + 2)^2 = $4x^2 + 8x + 4$

Problem 4.3: *b-complement*

We plan to use a fixed size b-complement number system with the base $b = 9$ and $n = 4$ digits.

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

b = 9
b-1 = 8
n = 4 ⟹ b^n = 6561 ⟹ range : [-3280, 3280]

The smallest number that can be represented is -3280 and the largest is 3280.

b) What is the representation of -1 and -8 in b-complement notation?

1_10 = 0001_9 ⟹ -1_10 = 8887_9 + 0001_9 = 8888_9
8_10 = 0008_9 ⟹ -8_10 = 8880_9 + 0001_9 = 8881_9

c) Add the numbers 1 and 8 in b-complement notation. What is the result in b-complement representation? Convert the result from b-complement representation back into the decimal number system.

(-1_9) + (-8_9) = 8888_9 + 8881_9 = 8880_9 in b-complement notation
But flipping it gives us 0008_9 and adding 1 gives us 0010_9 = 9, so the result is -9.

Problem 4.4: *munged passwords (haskell)*

a) Using pattern matching, implement a function sub that takes a character and returns either the character or a substitution of it. Write down the type signature of your function followed by its definition.

```
sub :: Char -> Char          --type signature
sub 'a' = '@'                --the function's definition start,
sub 'b' = '8'                --it substitutes the character, if it is possible,
sub 'c' = '('                --and returns either the character or a substitution of it.
sub 'd' = '6'
sub 'e' = '3'
sub 'f' = '#'
sub 'g' = '9'
sub 'h' = '#'
sub 'i' = '1'
sub 'l' = '1'
sub 'o' = '0'
sub 'q' = '9'
sub 's' = 's'
sub 'x' = '%'
sub 'y' = '?'
sub  x  =  x
```

b) Using pattern matching, implement a function munge that takes a string and returns a string with all character substitutions applied. Write down the type signature of your function followed by its definition.

```
munge :: String -> String    --type signature
munge z = map sub z          --definition starts, we use 'map' in order
                             --to apply the 'sub' function to every character of the string
                             --it returns a string.
```