

Problem 4.1: Injective, surjective, bijective functions

Given definitions: - Injective function: A function $f: X \rightarrow Y$ is called injective if every element of the codomain Y is mapped to by at most one element of the domain X : $\forall x, y \in X, f(x) = f(y) \Rightarrow x = y$

- Surjective function: A function $f: X \rightarrow Y$ is called surjective if every element of the codomain Y is mapped to by at least 1 element of the domain X : $\forall y \in Y, \exists x \in X, f(x) = y$

- Bijective functions: A function $f: X \rightarrow Y$ is called bijective if every element of the codomain Y is mapped to by exactly one element of the domain X . (That is, the function is both injective and surjective.)

(A) $f: \mathbb{R} \rightarrow \mathbb{R}$ with $f(x) = x^3$

1: Injective (One-to-One):

$$f(x_1) = f(x_2)$$

$$f(x_1) = x_1^3$$

$$f(x_2) = x_2^3$$

Suppose $f(x_1) = f(x_2)$

$$x_1^3 = x_2^3 \quad \bigg/ \sqrt[3]{}$$

$$x_1 = x_2$$

BRUNNEN

So, $f(x) = x^3$ is injective by definition since distinct inputs map to distinct outputs.

2: ~~Surjective~~ Surjective: Since the previous expression explained that there is at least one element in Y for every element in X , we can state that by definition, the expression $f(x) = x^3$ is surjective. This is true for $f(x) = x^3$ which covers the entire real numbers R as every real number has a unique real number for cube root.

3: Bijective: By definition, since the expression $f(x) = x^3$ is both injective and surjective, then it is also bijective.

$f: R \rightarrow R$ with $f(x) = x^3$ is injective, surjective, and bijective.

Problem 4.1: ③ $g: N \rightarrow N$ with $f(x) = 2x + 1$

1: Suppose: $g(x_1) = g(x_2)$

$$2x_1 + 1 = 2x_2 + 1 \quad / -1$$

$$2x_1 = 2x_2 \quad / \div 2$$

$$x_1 = x_2$$

So, $g(x) = 2x + 1$ is injective by definition since distinct inputs map to distinct outputs.

2: Surjective: Function $g(x)$ maps natural numbers to natural numbers and since every natural number (for $x \rightarrow$ some non-negative integer) can be expressed in $2x + 1$, the expression is surjective.

3: Bijective: By definition, since the expression $g(x) = 2x + 1$ is both injective and surjective, then it is also bijective.

$g: N \rightarrow N$ with ~~Surjective~~ $f(x) = 2x + 1$ is injective, surjective and bijective.

Problem 4.1: (C) $h: \mathbb{R} \rightarrow \mathbb{R}$ with $f(x) = \sin(x)$

1: $f(x) = \sin(x)$ is not injective because of the sine function.

The sine function can give the same output for different inputs such as:

$$\sin(0) = 0 \quad ; \quad \sin(\pi) = 0 \quad \text{where } \pi \text{ is the mathematical constant } \pi.$$

2: Although, $f(x) = \sin(x)$ gives at least 1 output for every input, it does not cover the entire real number \mathbb{R} line, therefore it is not surjective. *It only gives ~~values~~ values from -1 to 1.

3: By definition, since the expression is neither injective nor surjective, then it is not bijective either.

$h: \mathbb{R} \rightarrow \mathbb{R}$ with $f(x) = \sin(x)$ is not injective, not surjective, and not bijective.

Problem 4.2: Properties of function compositions

Let $f: X \rightarrow Y$ and $g: Y \rightarrow Z$ be two functions. Show that the following propositions are true.

a) If f and g are injective, then $g \circ f$ is injective.

To prove this we need to show if $g \circ f(x_1)$, then $x_1 = x_2$ for all instances of x_1 and x_2 in the domain X . Suppose the following:

$$g \circ f(x_1) = g \circ f(x_2)$$

$$g(f(x_1)) = g(f(x_2))$$

Since we are free to assume g is injective: if $g(m) = g(n)$, then $m = n$ for all " m " and " n " in its domain. Thus:

$$f(x_1) = f(x_2)$$

Now since f is also injective the same applies for $f(m) = f(n)$, then $m = n$ for all " m " and " n " in its domain. Thus:

$$x_1 = x_2$$

So we have shown if $g \circ f(x_1) = g \circ f(x_2)$, then $x_1 = x_2$, which means that $g \circ f$ is injective.

Problem 4.2 : (B) If f and g are surjective, then $g \circ f$ is surjective.

To prove this we need to show that for every element " z " in Z codomain, there exists an element " x " in X domain so that $g \circ f(x) = z$.

Since " g " is surjective, for every element " y " in codomain Y , an element " x " exists in the codomain Y : $g(y) = z$.

~~Since~~ Since " f " is surjective, for every element " x " in codomain Y , an element " x " exists in the domain X : $f(x) = y$.

For " x " element in domain X : $g \circ f$

$$g \circ f(x) = g(f(x))$$

$$g \circ f(x) = g(y)$$

$$g \circ f(x) = z$$

This shows that for every " z " element in codomain Z , there exists an element " x " in the X domain so: $g \circ f(x) = z$, meaning $g \circ f$ is surjective.

© If f and g are bijective, then $g \circ f$ is bijective

To prove this we need to show that $g \circ f$ is both injective and surjective, since we have done that in the two previous proofs (A) and (B) we have proven that © is also true.

(A), (B), and © have all been proven true.

Problem 4.3: Sets and Relations in a cinema

(a) Sets (Entities) in the Coffee Bar Scenario:

1. Customers (C) {Set of all customers in the scenario}
2. Movies Offered (MO) {Set of all movies that are shown}
3. Movie Theaters (MT) {Set of Movie Theaters in the Cinema}
4. Tickets (T) {Set of all tickets for all movies}
5. Cinema Staff (CS) {Set of all cinema employees}

(b) Relations between sets:

1. Purchases: P (Relation between Customers and Tickets Sets which represents which customer would purchase what ticket).

$$P \subseteq C \times T$$

2. Schedule or Screening for each Theater (ST): (Relation between MO and MT representing the movies offered at each theatre)

$$ST \subseteq MO \times MT$$

3. ~~Drink~~ Drink Orders: DO (Relation between CS and C representing which employees take orders from which customers):

$$DO \subseteq CS \times C$$

4. Employee Placement: EP (Relation between CS and MT representing which employees work at which Movie Theaters).

$$EP \subseteq CS \times MT$$

5. Ticket Verification: TV (Relation between CS and T representing which employees in the cinema validate which tickets as verified for each movie).

$$TV \subseteq CS \times T$$

©

~~Endo~~ Endo relations (Relations within a Single Set):

1. Partial Order Relation: Movie Ratings

For PO_{mo} a partial order relation on MO, with some movies surpassing others in rating such that:

$$PO_{mo} \subseteq MO \times MO$$

2. Equivalence Relations: Employee Roles

For ER_{cs} an equivalence relation on CS, which explains some employees are related to each other with their given role (such as cashiers):

$$ER_{cs} \subseteq CS \times CS$$

3. Strict Partial Order Relation: Employee Shifts

For SPOR_{cs} a strict partial order relation on CS, indicating that some employees have strict shifts that are shorter than others:

$$SPOR_{CS} \subseteq CS \times CS$$

4. Partial Order Relation: Theater Capacities

Let PO_T be a partial order relation on T , showing some theaters have higher capacities than others:

$$PO_T \subseteq T \times T$$

5. Equivalence Relation: Customer's Age

Let Eq_C be an equivalence relation on C , where relation between two customers is made if they are of the same Age.

$$Eq_C \subseteq C \times C$$

Problem 4.4: characters and types (haskell)

a) The 'ord' function from Data.Char module is needed.

```
import Data.Char

main :: IO()
main = print (ord '=')
```

Output is : 61

b) To print the character with code 128119 (decimal) we need the 'chr' function from 'Data.Char' module:

```
import Data.Char

main :: IO()
main = print (chr 128119)
```

Output: '\128119'

I believe that my computer is able to produce the character with this point code and therefore is giving me this output. This happens no matter what compiler I use.

for clarification
when I say
 \rightarrow I mean
" \rightarrow " and
not the arrow.

(C) The type signature of 'zipWith' is:

$\text{zipWith} :: (a \rightarrow b \rightarrow c) \rightarrow [a] \rightarrow [b] \rightarrow [c]$

'zipWith' takes three arguments:

* First: argument is a function that takes two arguments "a" and "b" and produces "c" type.

* Second: A list of values type "a".

* Third: A list of values type "b".

'zipWith' applies the function element-wise to corresponding elements of the input lists, thus creating a new list containing the application of the function

(D) The type signature of the 'isPrefixOf' function is:

$\text{isPrefixOf} :: \text{Eq } a \Rightarrow [a] \rightarrow [a] \rightarrow \text{Bool}$

Takes 2 arguments:

* First: A list of values of type "a" ^{which} must belong to the 'Eq' type.

* Second: Another list of values type "a"

The function checks if the first list is a prefix of the second and returns "True" or "False".

THE END