# SOEN 331: Introduction to Formal Methods for Software Engineering

# **Assignment 1**

## Propositional and Predicate Logic, Structures, Binary Relations, Functions and Relational Calculus


Duc Nguyen - 40064649
Vithura Muthiah - 40062305
Auvigoo Ahmed - 40128901
Ali Hanni - 40157164

*Gina Cody School of Computer Science and Software Engineering*
*Concordia University, Montreal, QC, Canada*


Winter 2021

# Contents

# 1 Problem 1 (8 pts)

## 1.1 Description:

You are shown a set of four cards placed on a table, each of which has a **number** on one side and a **symbol** on the other side. The visible faces of the cards show the numbers **2** and **7**, and the symbols $\square$, and $\bigcirc$.

Which card(s) must you turn over in order to test the truth of the proposition that *"If a card has an odd number on one side, then it has the symbol* $\square$ *on the other side"*? Explain your reasoning by deciding for <u>each</u> card whether it should be turned over and why.

## 1.2 Answer:

Let us assume that p represent the statement "A card has an odd number on one side" and the statement q represent "It has the symbol $\square$ on the other side".

The first card's visible face shows the number 2. The statement reads: *"If a card has an odd number on one side, then it has the symbol* $\square$ *on the other side"* (p $\mapsto$ q). Since 2 is not an odd number, whatever symbol is behind that card will not affect the truth of the proposition. Therefore, it does not need to be turned over. In other words, if p is false, then whatever truth value of q (let it be a square of a circle), the proposition of p $\mapsto$ q will always remain true.

The second card's visible face shows the number 7. Since this is an odd number, to test the truth value of the stated proposition, this card has to be turned over to check if it has a square symbol on the back. In other words, the statement p is true in this case, therefore the truth value of the proposition p $\mapsto$ q will only be true if the statement q is true. To verify if q is true, the card needs to be turned over to check if it is a square on the back side.

The third card's visible face shows the square symbol. In this case, the statement q is true (the card has a square symbol). Therefore, the proposition p $\mapsto$ q will be true regardless of the truth value of the statement p (it can be an odd number or an even number). Therefore, this card does not need to be turned over.

The fourth card's visible face shows the circle symbol. In this case, the statement q is false (the card has a circle symbol). Therefore, the proposition p $\mapsto$ q will only be true if the truth value of the statement p is true (there is an odd number on the back). Therefore, to verify if p is true, the card needs to be turned over to check if it is an odd number on the back side.

In conclusion, the card with visible sides 7 and $\bigcirc$ will be turned over.

# 2 Problem 2 (8 pts)

## 2.1 Description

Consider the predicate *asks(a, b)* that is interpreted as *"a has asked b out on a date."*

1. Translate the following into English: $\forall\, a \exists\, b\ asks(a, b)$ and $\exists\, b \forall\, a\ asks(a, b)$.
2. Can we claim that $\forall\, a \exists\, b\ asks(a, b) \rightarrow \exists\, b \forall\, a\ asks(a, b)$? Discuss in detail.
3. Can we claim that $\exists\, b \forall\, a\ asks(a, b) \rightarrow \forall\, a \exists\, b\ asks(a, b)$? Discuss in detail.

## 2.2 Answer

1. $\forall\, a \exists\, b\ asks(a, b) \rightarrow$ Every person has asked at least one person out on a date.

   $\exists\, b \forall\, a\ asks(a, b) \rightarrow$ There is one person that have been asked out on a date by everyone else.

2. We cannot claim that $\forall\, a \exists\, b\ asks(a, b) \rightarrow \exists\, b \forall\, a\ asks(a, b)$. The first predicate states that everyone has asked out at least one person (b) on a date. We have no information about the other person. Everyone could have asked someone different or they could have asked the same person. The predicate does not provide this information.
   The second predicate states that one person was asked out by everyone. Here it is specified that everyone asked out the same person.
   Therefore, $\forall\, a \exists\, b\ asks(a, b) \rightarrow \exists\, b \forall\, a\ asks(a, b)$ is **false**.

3. We can confirm that it is justifiable to claim that $\exists\, b \forall\, a\ asks(a, b) \rightarrow \forall\, a \exists\, b\ asks(a, b)$. Indeed, claiming that someone have been asked out by everyone ($\exists\, b \forall\, a\ asks(a, b)$) implies that everyone has asked out at least one person ($\forall\, a \exists\, b\ asks(a, b)$).
   Therefore, the claim $\exists\, b \forall\, a\ asks(a, b) \rightarrow \forall\, a \exists\, b\ asks(a, b)$ is true.

# 3 Problem 3 (12 pts)

## 3.1 Description:

Let *scientist*($x$) denote the statement "x is a scientist", and *honest*($x$) denote the statement "x is honest." Formalize the following sentences and indicate their corresponding formal type.

1. "No scientists are honest."
2. "All scientists are crooked."
3. "All scientists are honest."
4. "Some scientists are crooked."
5. "Some scientists are honest."
6. "No scientist is crooked."
7. "Some scientists are not crooked."
8. "Some scientists are not honest."

Identify pairs that are contradictories, contraries, subcontraries, and pairs that support subalteration (clearly indicating superaltern and subaltern).

## 3.2 Formalize sentences and indicate formal type

Let's denote

**P(x)**: scientist(x): "x is a scientist"

**Q(x)**: honest(x): "x is honest"

We can formalize the statements as following

1. "No scientists are honest." $= \forall\, x, (P(x) \rightarrow \neg Q(x)) =$ E form
2. "All scientists are crooked." $= \forall\, x, (P(x) \rightarrow \neg Q(x)) =$ E form
3. "All scientists are honest." $= \forall\, x, (P(x) \rightarrow Q(x)) =$ A form
4. "Some scientists are crooked." $= \exists\, x, (P(x) \wedge \neg Q(x)) =$ O form
5. "Some scientists are honest." $= \exists\, x, (P(w) \wedge Q(x)) =$ I form
6. "No scientist is crooked." $= \forall\, x, (P(x) \rightarrow Q(x)) =$ A form
7. "Some scientists are not crooked." $= \exists\, x, (P(x) \wedge Q(x)) =$ I form
8. "Some scientists are not honest." $= \exists\, x, (P(x) \wedge \neg Q(x)) =$ O form

## 3.3 Identify pairs that are contradictories, contraries, subcontraries, and pairs that support subalteration

### 3.3.1 Pairs of contradictories

- (3) and (4)
- (6) and (4)
- (3) and (8)
- (6) and (8)
- (5) and (1)
- (5) and (2)
- (7) and (1)
- (7) and (2)

### 3.3.2 Pairs of contraries

- (3) and (1)
- (6) and (1)
- (3) and (2)
- (6) and (2)

### 3.3.3 Pairs of subcontraries

- (5) and (8)
- (7) and (8)
- (5) and (4)
- (7) and (4)

### 3.3.4 Pairs that support subalteration

- Subaltern: (5) - Superaltern: (3)
- Subaltern: (5) - Superaltern: (6)
- Subaltern: (7) - Superaltern: (3)
- Subaltern: (7) - Superaltern: (6)
- Subaltern: (4) - Superaltern: (1)

- Subaltern: (4) - Superaltern: (2)
- Subaltern: (8) - Superaltern: (1)
- Subaltern: (8) - Superaltern: (2)

# 4 Problem 4 (12 pts)

## 4.1 Description:

Consider list $\Lambda = \langle w, x, y, z \rangle$, deployed to implement a Queue Abstract Data Type.

1. Let the head of $\Lambda$ correspond to the front position of the Queue. Implement operations `enqueue(el, Λ)` and `dequeue(Λ)` using list construction operations. In both cases we can refer to $\Lambda'$ as the state of the list upon successful termination of one of its operations.
2. Let us now reverse the way we manipulate our data structure and let the head of $\Lambda$ correspond to the rear of the Queue.
   (a) What would be the result of $cons(el, \Lambda)$, and would it be a correct implementation for operation `enqueue(el, Λ)`?
   (b) What would be the result of $list(el, \Lambda)$, and would it be a correct implementation for operation `enqueue(el, Λ)`?
   (c) What would be the result of $concat(list(el), \Lambda)$, and would it be a correct implementation for operation `enqueue(el, Λ)`?

## 4.2 Enqueue and dequeue operations with head of list as front of queue

### 4.2.1 enqueue(el, Λ)

Λ' = concat(Λ, list(el));

### 4.2.2 dequeue(Λ)

- element = head(Λ);
- Λ' = tail(Λ);

where 'element' is the return value of the operation.

## 4.3 Let head of A correspond to the rear of Queue

### 4.3.1 cons(el, Λ)

**Result:** $\langle el, w, x, y, z \rangle$

It's the correct implementation for operation `enqueue(el Λ)` since it adds the new element to the rear of the queue (head of Λ). Hence, it is acceptable since it implements the Queue protocol.

#### 4.3.2 list(el, $\Lambda$)

**Result:** $\langle el, \langle w, x, y, z\rangle\rangle$

It's **not** the correct implementation for operation `enqueue(el Λ)` since it creates list containing A as a list inside. Hence, it is not acceptable because it does not implement the Queue protocol.

#### 4.3.3 concat(list(el), $\Lambda$)

**Result:** $\langle el, w, x, y, z\rangle$

It's the correct implementation for operation `enqueue(el Λ)` since it adds the new element to the rear of the queue (head of $\Lambda$). Hence, it is acceptable since it implements the Queue protocol.

# 5 Problem 5 (12 pts)

## 5.1 Description:

Let $A = \{0, 1, 2, 3, 4\}$ and relations $R$, $S$, $T$, and $U$ on $A$ defined as follows:

$$R = \{(0,0), (0,1), (0,3), (1,0), (1,1), (2,2), (3,0), (3,1), (3,3), (4,0), (4,1), (4,3), (4,4)\}$$

$$S = \{(0,1), (1,1), (2,3), (2,4), (3,0), (3,4), (4,0), (4,1), (4,4)\}$$

$$T = \{(0,3), (0,4), (2,1), (3,2), (4,2), (4,3)\}$$

$$U = \{(0,0), (0,1), (0,3), (1,0), (1,1), (1,3), (2,2), (3,0), (3,1), (3,3), (4,4)\}$$

Fill in the table below, using ✓, or ×.

| | | $R$ | $S$ | $T$ | $U$ |
|---|---|---|---|---|---|
| **Reflexive** | | | | | |
| **Irreflexive** | | | | | |
| **Symmetric** | | | | | |
| **Asymmetric** | | | | | |
| **Antisymmetric** | | | | | |
| **Transitive** | | | | | |
| **Equivalence** | | | | | |
| **Partial order** | | | | | |

## 5.2 Answer:

| | | $R$ | $S$ | $T$ | $U$ |
|---|---|---|---|---|---|
| **Reflexive** | | ✓ | × | × | ✓ |
| **Irreflexive** | | × | × | ✓ | × |
| **Symmetric** | | × | × | × | ✓ |
| **Asymmetric** | | × | × | ✓ | × |
| **Antisymmetric** | | × | ✓ | ✓ | × |
| **Transitive** | | × | × | × | ✓ |
| **Equivalence** | | × | × | × | ✓ |
| **Partial order** | | × | × | × | × |

Relation R

- Reflexive: Since (0,0), (1,1), (2,2) (3,3), (4,4) $\in R$, it is reflexive.
- Irreflexive: Not irreflexive since an element that is related to itself exists (i.e (0,0)).
- Symmetric: Not symmetric since (3,1) $\in R$, but (1,3) $\notin R$.

- Asymmetric: Not asymmetric since (0,1), (1,0) $\in R$.
- Antisymmetric: Not antisymmetric since (0,1), (1,0) $\in R$, but $0 \neq 1$.
- Transitive: Not transitive since (1,0), (0,3) $\in R$, but (1,3) $\notin R$.
- Equivalence: Not an equivalence relation since it is not reflexive, symmetric and transitive.
- Partial order: Not a partial order relation since it is not reflexive, antisymmetric and transitive.

<u>Relation S</u>

- Reflexive: Not reflexive since (0,0) $\notin S$.
- Irreflexive: Not irreflexive since (1,1) $\in S$.
- Symmetric: Not symmetric since (0,1) $\in S$, but (1,0) $\notin S$.
- Asymmetric: Not asymmetric since (1,1) $\in S$, which is a self-loop.
- Antisymmetric: Since every element in $S$ is not bidirectional except for self-loops, it is antisymmetric.
- Transitive: Not transitive since (2,3), (3,0) $\in S$, but (2,0) $\notin S$.
- Equivalence: Not an equivalence relation since it is not reflexive, symmetric and transitive.
- Partial order: Not a partial order relation since it is not reflexive, antisymmetric and transitive.

<u>Relation T</u>

- Reflexive: Not reflexive since (0,0) $\notin S$.
- Irreflexive: Since no element that is related to itself belongs to $T$, it is irreflexive.
- Symmetric: Not symmetric since (0,3) $\in T$, but (3,0) $\notin T$.
- Asymmetric: Since no element in $T$ is bidirectional and there are no elements that are self-loops, it is asymmetric.
- Antisymmetric: Since the relation is asymmetric, it is antisymmetric.
- Transitive: Not transitive since (0,3), (3,2) $\in S$, but (0,2) $\notin S$.
- Equivalence: Not an equivalence relation since it is not reflexive, symmetric and transitive.

- Partial order: Not a partial order relation since it is not reflexive, antisymmetric and transitive.

<u>Relation U</u>

- Reflexive: Since (0,0), (1,1), (2,2) (3,3), (4,4) $\in U$, it is reflexive.
- Irreflexive: Not irreflexive since an element that is related to itself exists (i.e (0,0)).
- Symmetric: Since every pair in $U$ is bidirectional, it is symmetric.
- Asymmetric: Not asymmetric since (0,1), (1,0) $\in U$.
- Antisymmetric: Not antisymmetric since (0,1), (1,0) $\in U$, but $0 \neq 1$.
- Transitive: Since $\forall\, a, b, c \in A$: $(aUb \wedge bUc) \rightarrow aUc$, it is transitive.
- Equivalence: Since the relation is reflexive, symmetric and transitive, it is an equivalence relation.
- Partial order: Not a partial order relation since it is not reflexive, antisymmetric and transitive.

# 6 Problem 6 (8 pts)

## 6.1 Description:

Consider the relation *"is a subtype of"* over the set $\{rectangle, quadrilateral, square, parellelogram, rhombus\}$.

1. Is this an *equivalence relation*?
2. Is this relation a *partial order*? If so, create a *Hasse diagram*, and identify *minimal* and *maximal* elements.

## 6.2 Answer:

Consider the relation *"is a subtype of"* as $R$ over the set $A = \{rectangle, quadrilateral, square, parallelogram, rhombus\}$.

$R = \{(rectangle, quadrilateral), (rectangle, parallelogram), (square, rectangle),$
$(square, quadrilateral), (square, parallelogram), (square, rhombus),$
$(parallelogram, quadrilateral), (rhombus, quadrilateral), (rhombus, parallelogram),$
$(rectangle, rectangle), (quadrilateral, quadrilateral), (square, square),$
$(parallelogram, parallelogram), (rhombus, rhombus)\}$

1. In order for a relation to be an equivalence relation, it must be reflexive, symmetric and transitive.

   - $R$ is reflexive since every element in $A$ is related to itself. That is, $\forall\, a \in A : aRa$.
   - $R$ is not symmetric since $(rectangle, quadrilateral) \in R, but(quadrilateral, rectangle) \notin R$.
   - $R$ is transitive since for elements $a, b, c$ in the set $A$, if $a$ and $b$ are related by $R$, and $b$ and $c$ are related by $R$, then $a$ and $c$ are also related by $R$. That is, $\forall\, a, b, c \in A : (aRb \wedge bRc) \rightarrow aRc$.

   Since the relation is reflexive, transitive, but not symmetric, it is not an equivalence relation.

2. In order for a relation to be a partial order relation, it must be reflexive, antisymmetric and transitive.

   - R is antisymmetric since for elements $a, b$ in the set $A$, if $aRb$ and $bRa$, then $a = b$. In other words, there is no pair in R such that the reverse of this pair exists, unless the pair contains identical elements.

   From the previous point, it has been concluded that $R$ is reflexive and transitive. Since $R$ is also antisymmetric, it is a partial order.

   The Hasse diagram of $R$ is shown in the following figure:

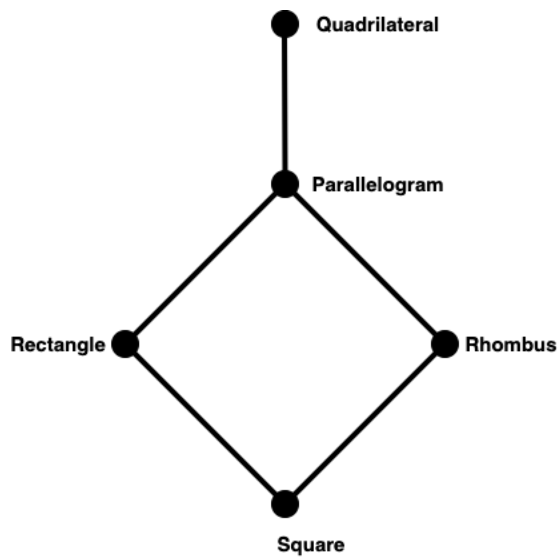


Figure 1: Hasse Diagram for Problem 6

From figure 1, it can be concluded that the maximal element is *Quadrilateral*, since it does not have any successors, and the minimal element is *Square*, since it does not have any predecessors.

# 7 Problem 7 (8 pts)

## 7.1 Description:

Consider the set $A = \{w, x, y, z\}$, and the relations

$$S = \{(w, x), (w, y), (x, w), (x, x), (z, x)\}$$

$$T = \{(w, w), (w, y), (x, w), (x, x), (x, z), (y, w), (y, y), (y, z)\}$$

Find the following compositions:

1. $S \circ T$
2. $T \circ S$
3. $T^{-1} \circ S^{-1}$

**NOTE**: Some authors (e.g. Rosen) adopt a different ordering of operands than the one we use in our lecture notes. Please follow the ordering (and the definition) of the lecture notes.

## 7.2 Answer:

1. $S \circ T = \{(w, w), (w, x), (w, z), (w, y), (x, w), (x, y), (x, x), (x, z), (z, w), (z, x), (z, z)\}$
2. $T \circ S = \{(w, x), (w, y), (x, x), (x, y), (x, w), (y, y), (y, x)\}$
3. $T^{-1} \circ S^{-1} = \{(w, x), (w, w), (w, z), (x, w), (x, x), (x, z), (y, x), (y, w), (z, w), (z, x), (z, z)\}$

# 8 Problem 8 (12 pts)

## 8.1 Description:

Consider sets $A = \{1, 2, 3, 4, 5, 6\}$ and $B = \{a, b, c, d, e, f\}$.

1. Determine the type of the correspondence in each of the following cases, or indicate if the correspondence is not a function.

   (a) $\{1 \mapsto b, 2 \mapsto c, 3 \mapsto e, 4 \mapsto d, 5 \mapsto f, 3 \mapsto a\}$

   (b) $\{1 \mapsto a, 2 \mapsto d, 3 \mapsto a, 4 \mapsto f, 5 \mapsto d, 6 \mapsto c\}$

   (c) $\{1 \mapsto c, 2 \mapsto b, 3 \mapsto d, 4 \mapsto e, 5 \mapsto e, 6 \mapsto f\}$

   (d) $\{1 \mapsto b, 2 \mapsto c, 3 \mapsto e, 4 \mapsto d, 5 \mapsto f, 6 \mapsto a\}$

   Fill in the table below, using ✓, or ×.

| | Injective | Surjective | Bijective | Neither injective nor surjective | Not a function |
|---|---|---|---|---|---|
| (a) | | | | | |
| (b) | | | | | |
| (c) | | | | | |
| (d) | | | | | |

2. Is it possible to construct a function $f\colon A \to B$ which is surjective and not injective? Discuss.

## 8.2 Answer:

| | Inective | Surjective | Bijective | Neither injective nor surjective | Not a function |
|---|---|---|---|---|---|
| (a) | | | | | ✓ |
| (b) | | | | ✓ | |
| (c) | | | | ✓ | |
| (d) | | | ✓ | | |

1. A function from $A$ to $B$ is an assisgnment of each element of the domain to exactly one element of the codomain. We know that a function is surjective if each of the elements of the codomain is mapped by at least one element of the domain.
   Therefore, if a function $f\colon A \to B$ is to be surjective, all elements of $A$ must point to one and only one element of $B$ (definition of a function) and all elements of $B$ must be pointed by at least one element of $A$ (definition of a surjective function).
   Fianlly, we know that $A$ and $B$ have the same cardinality, namely 6. This yields that for $f$ to be surjective, all elements of $B$ have to be pointed by exactly one element of $A$, making $f$ an injective function.
   In conclusion, given sets $A$ and $B$, it is impossible for a function $f : A \to B$ to be surjective without being injective.

# 9 Problem 9 (20 pts)

## 9.1 Description

Consider the following relation:

$$laptops : Model \leftrightarrow Brand$$

where

$$\begin{array}{l} laptops = \\ \quad \{ \\ \quad\quad legion5 \mapsto lenovo, \\ \quad\quad macbookair \mapsto apple, \\ \quad\quad xps15 \mapsto dell, \\ \quad\quad spectre \mapsto hp, \\ \quad\quad xps13 \mapsto dell, \\ \quad\quad swift3 \mapsto acer, \\ \quad\quad macbookpro \mapsto apple, \\ \quad\quad dragonfly \mapsto hp, \\ \quad\quad envyx360 \mapsto hp \\ \quad \} \end{array}$$

1. What is the domain and the range of the relation?

2. What is the result of the expression

   $$\{xps15, xps13, swift3, envyx360\} \lhd laptops$$

   What is the meaning of operator $\lhd$ and where would you deploy such operator in the context of a database management system?

3. What is the result of the expression

   $$laptops \rhd \{lenovo, hp\}$$

   What is the meaning of operator $\rhd$ and where would you deploy such operator in the context of a database management system?

4. What is the result of the expression

   $$\{legion5, xps15, xps13, dragonfly\} \ntriangleleft laptops$$

   What is the meaning of operator $\ntriangleleft$ and where would you deploy such operator in the context of a database management system?

5. What is the result of the expression

$$laptops \rhd \{apple, dell, hp\}$$

What is the meaning of operator $\rhd$ and where would you deploy such operator in the context of a database management system?

6. Consider the following expression

$$laptops \oplus \{ideapad \mapsto lenovo\}$$

(a) What is the result of the expression?

(b) What is the meaning of operator $\oplus$ and where would you deploy such operator in the context of a database management system?

(c) Does the result of the expression have a permanent effect on the database (relation)? If not, describe in detail how would you ensure a permanent effect.

## 9.2 Answer

1. dom laptops = {legion5, macbookair, xps15, spectre, xps13, swift3, macbookpro, dragonfly, envyx360 }

   ran laptops = {lenovo, apple, dell, hp, acer}

2. { xps15, xps13, swift3, envyx360 } $\lhd$ laptops =
   { xps15 $\mapsto$ dell,
   xps13 $\mapsto$ dell,
   swift3 $\mapsto$ acer,
   envyx360 $\mapsto$ hp }

   $\lhd$ is a domain restriction operator that selects pairs from the database table (in this case from laptops database table) based on the first element. Such operator is deployed in the database management system for query operation. It is specifically a select query (a data retrieval query).

3. laptops $\rhd$ { lenovo, hp } =
   {legion5 $\mapsto$ lenovo,
   spectre $\mapsto$ hp,
   dragonfly $\mapsto$ hp,
   envyx360 $\mapsto$ hp }

   $\rhd$ is a range restriction operator that selects pairs from the database table (in this case from laptops database table) based on the second element. As previously mentioned, this is also deployed in database management for query operation (specifically a select query used for data retrieval query).

4. $\{$ legion5, xps15, xps13, dragonfly $\} \lhd$ laptops $=$
$\{$macbookair $\mapsto$ apple,
spectre $\mapsto$ hp,
swift3 $\mapsto$ acer,
macbookpro $\mapsto$ apple,
envyx360 $\mapsto$ hp $\}$

$\lhd$ is a domain subtraction operator that removes all pairs based on the first element (the domain of the element). In other words, it removes all pairs where model is anything specified within the brackets. This is also a query operation (more specifically an action query) where additional operation (such as a deletion in this case) is applied on the data.

5. laptops $\rhd \{$ apple, dell, hp $\} =$
$\{$legion5 $\mapsto$ lenovo,
swift3 $\mapsto$ acer $\}$

$\rhd$ is a range subtraction operator that removes all pairs based on the second element (the range of the relation). In other words, it removes all pairs where the brand is anything specified within the brackets. This is an action query, where a deletion (in this case) is applied on the data.

6. (a) laptops $\oplus \{$ ideapad $\mapsto$ lenovo $\} =$
$\{$ legion5 $\mapsto$ lenovo,
macbookair $\mapsto$ apple,
xps15 $\mapsto$ dell,
spectre $\mapsto$ hp,
xps13 $\mapsto$ dell,
swift3 $\mapsto$ acer,
macbookpro $\mapsto$ apple,
dragonfly $\mapsto$ hp,
envyx360 $\mapsto$ hp,
ideapad $\mapsto$ lenovo$\}$

(b) $\oplus$ is an insertion operator used in model database updates for relational overriding. Therefore, the specified element within the bracket is added to the already existing laptop set. Such operator is deployed in the database management system for query operation, specifically an action query, where (in this case) the action is an insertion of data to the existing database.

(c) The result will not have a permanent effect on the database, because this is just an evaluation of the expression. To ensure a permanent effect, the result of this expression must be assigned to a variable (laptops') which will hold the state of the set upon successful evaluation of the right-hand side expression:
laptops' = laptops $\oplus$ { ideapad $\mapsto$ lenovo }