By Maha Rana

CS636 Finals Subjective Solved

✅Short Questions

1. Purpose of `inRange`:

inRange Function

- The purpose of this function is to check whether an integer value, val, is within the range MIN and MAX as defined earlier.
- You can see that the use of the equivalence connective ensures that the output is true if the input is in range, but is otherwise false.



2. Truth Table Banana Tha:

Truth tables show all possible combinations of **truth values (T/F)** for propositions and their **result** under logical operations (AND, OR, NOT, etc.).

Example for AND:

p	q	p ∧ q
T	T	T
T	F	F
F	T	F
F	F	F

➡ *[Slide Ref: Propositional Logic – AND/OR Explained]*

By Maha Rana

3. Two Ways of Specify Function in VDM-SL:

Specifying a Function in VDM-SL

There are two ways in which we can specify a function in VDM-SL:

- Explicitly
- Implicitly

➡ *[Slide Ref: Topic#063 – Specifying Functions]*

4. Sequence Definition and Three Kinds:

A **sequence** is an **ordered collection** where **repetitions matter**.
Types/Kinds of sequences:

- Normal sequence: `[a, b, c]`
- Empty sequence: `[]`
- Repeating sequence: `[a, b, a]`

➡ *[Slide Ref: Sequences – Intro and Examples]*

5. Domain Operators:

□ Domain and Range Operator

The *domain* operator, **dom**, returns the set of all the domain elements of the maplets.

The *range* operator, **rng**, returns the set of all the range elements.

$m1 = \{a \mapsto 1, b \mapsto 2, c \mapsto 2, d \mapsto 3, e \mapsto 4\}$

$m2 = \{a \mapsto 2, f \mapsto 1, c \mapsto 7\}$

$m3 = \{f \mapsto 2, g \mapsto 6\}$

- **dom** $m1 = \{a, b, c, d, e\}$
- **rng** $m1 = \{1, 2, 3, 4\}$
- **dom** $m2 = \{a, f, c\}$
- **rng** $m2 = \{1, 2, 7\}$

By Maha Rana

➡ *[Slide Ref: Maps – Domain Operator]*

✅Long Questions

1. 5 Sequence Operators Table Matching Tha:

Here are **five sequence operators** and their actions:

Operator	Description	Example
`len`	Length of sequence	`len [a,b,c] = 3`
`hd`	Head (first element)	`hd [a,b,c] = a`
`tl`	Tail (all except first)	`tl [a,b,c] = [b,c]`
`elems`	Converts sequence to set	`elems [a,b,a] = {a,b}`
`inds`	Returns set of valid indices	`inds [a,b,c] = {1,2,3}`

2. Negation, AND, OR, Exclusive OR, Implication – Match Symbols:

Operation	Symbol	Description
Negation	¬	Opposite value
AND	∧	True if both are true
OR	∨	True if at least one is true
Exclusive OR	⊕	True if values are **different**
Implication	→	If p then q (false only if p = T, q = F)

➡ *[Slide Ref: Propositional Logic – Operators Explained]*

3. Robust ke bary mein tha – 5 Robust Evaluation Batany Thy:

- **Completeness:** The method should cover all essential behaviors and requirements of the system.

- **Scalability:** It should work well for both small and large systems without becoming slow or impractical

- **Expressiveness:** It should provide rich language features to accurately represent different system behaviors
- **Automated:** Automated tools should help with model checking and verification, reducing human errors.
- **Soundness and Completeness of Analysis**: Any conclusion drawn from the method should be theoretically correct and explore all relevant aspects of the system.

✓✓Short Questions

1. Universal Quantifier:

Quantification

- The universal quantifier ∀:
- This quantifier enables a predicate to make a statement about all the elements in a particular set. For example, if M(x) is the predicate x chases mice, we could write:
- ∀x ∈ Cats ● M(x)
- This reads *For all the x which are members of the set Cats, x chases mice*, or, more simply, *All cats chase mice*.

➡ *[Slides – Topic: Predicate Logic – Quantification]*

By Maha Rana

2. `inRange`:

inRange Function

- The purpose of this function is to check whether an integer value, val, is within the range MIN and MAX as defined earlier.
- You can see that the use of the equivalence connective ensures that the output is true if the input is in range, but is otherwise false.

$$
\begin{array}{l}
inRange(val: \mathbb{Z})\ result: \mathbb{B} \\
\textbf{pre} \quad \text{TRUE} \\
\textbf{post} \quad result \Leftrightarrow MIN \leq val \leq MAX
\end{array}
$$

➡ *[Slides – Topic#069: inRange Function]*

3. Explicitly ke 2 Functions:
Explicit function directly describes **how output is computed**.
Examples from slides:

- `add(x, y) == x + y`
- `square(n) == n * n` (implied style in slides)

  ➡ *[Slides – Topic#063: Explicit Function]*

4. Which Operator Joins Two Sequences?
Answer: **Concatenation Operator (^)**

Sequence Operators

- **The concatenation operator (^) operates on two sequences, and returns a sequence that consists of the two sequences joined together**

if first = [w, e, r, w]
and second = [t, w, q]
then
first^second = [w, e, r, w, t, w, q]

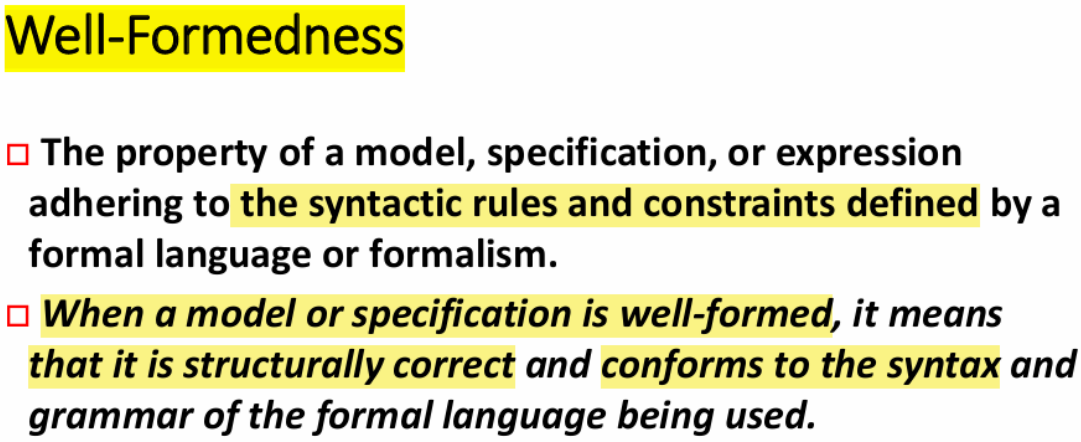
By Maha Rana

It joins two sequences into one.
Example:
`[a,b] ^ [c,d] = [a,b,c,d]`

➡ *[Slides – Sequence Operators: Concatenation]*

✓Long Questions

1. Well-Formedness:



➡ *[Slides – Topics #064, #065 – Invariant & Initialization]*

2. Predicate Logic:

Predicate logic is an extension of propositional logic that allows reasoning about **sets and variables**.

- **Predicates** are truth-valued functions with variables, like `C(x): x is a cat`.
- You can bind variables using:
  - **Substitution** (e.g., `C(Simba)`)
  - **Quantification** (e.g., `∀x ∈ Cats ● M(x)`)

It is useful to **express general properties** of elements in a set.

➡ *[Slides – Topic: Predicate Logic]*

By Maha Rana

3. Difference Find Karna Tha j and k Ka (Set Difference):

Set Operations

- *Three operations that take two sets as input and return a new set as output*
  - *Set union*
  - *Set intersection*
  - *Set difference*
- *In each case, the types of elements in each set are assumed to be the same.*

Set Operations: Union

- **The union of two sets, j and k returns a set that contains all the elements of the set j and all the elements of the set k.**

  $J \cup K$ (J Union K)

- **Example:**

  if j = {<MON>, <TUE>, <WED>, <SUN>}
  and k = {<MON>, <FRI>, <TUE>}
  then $j \cup k$ = {<MON>, <TUE>, <WED>, <SUN>, <FRI>}

➡ *[Slides – Set Operations: Difference]*

✅Short Questions

1. **Fallacy Argument:**

By Maha Rana

ARGUMENTS

- An argument which is not valid is called fallacy.
- Eg: fallacy argument
- $p \rightarrow q, q \vdash p$

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

➡ *[Slides – Propositional Logic → Arguments → Fallacy]*

2. (Universal Quantifier):

Symbol: **∀**
Used to say something is true for **all elements** in a set.

Example:

```
∀x ∈ Cats ● M(x)
⇒ All cats chase mice
```

➡ *[Slides – Predicate Logic → Quantification]*

3. Ways of Binding a Variable:

Two ways to bind variables in **Predicate Logic**:

1. **Substitution** – Replace variable with a specific value.
    - e.g., `C(Simba)`

2. **Quantification** – Use quantifiers like ∀ or ∃.
    - e.g., `∀x ∈ Cats ● M(x)`

      ➡ *[Slides – Binding Variables]*

4. Robustness:

Robustness in a formal specification is achieved through:

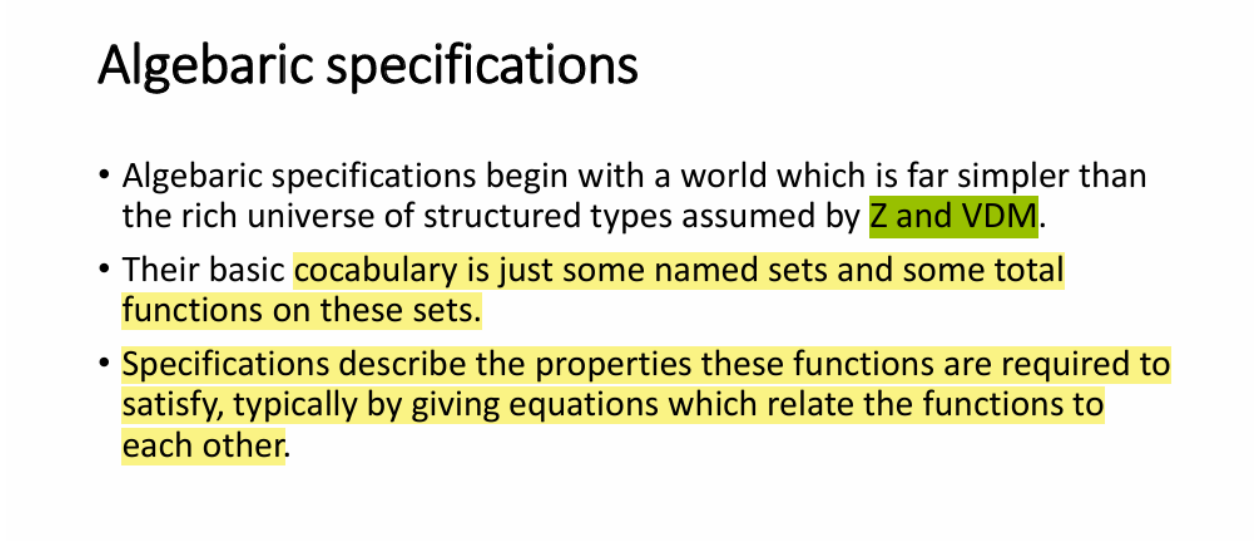
- **Completeness:** The method should cover all essential behaviors and requirements of the system.
- **Scalability:** It should work well for both small and large systems without becoming slow or impractical
- **Expressiveness:** It should provide rich language features to accurately represent different system behaviors
- **Automated:** Automated tools should help with model checking and verification, reducing human errors.
- **Soundness and Completeness of Analysis**: Any conclusion drawn from the method should be theoretically correct and explore all relevant aspects of the system.

➡ *[Slides – Topics#064–065: Invariant, Initialization]*

5. Algebraic Methods Name:

These are methods that specify systems **using equations** and **algebraic structures** like sets, functions, and tuples.

By Maha Rana



Examples include:

- **Z notation**
- **VDM**
- **B-method**
  (Though not deeply named in slides, Z and VDM are mentioned.)

➡ *[Slides – Referenced under Z and VDM Topics]*

6. Rules to Name Wala:

Rules for naming **user-defined types** in VDM-SL:

- **Type names** begin with an **uppercase** letter
- Quote types are **in angle brackets** like `<INCREASE>`, `<DECREASE>`
  ➡ *[Slides – Topic#066: User-defined Types]*

✅Long Questions

1. Z Language:

- Z is a **formal specification language** based on set theory and first-order predicate logic.
- It uses **schemas** to describe system states and operations.
- Not covered in depth in slides, but **VDM is similar in role and purpose.**

By Maha Rana

➡ *[Slides – Mentioned indirectly, VDM focus]*

2. Semantics of Z:

(Z semantics not directly covered in slides, but general understanding applies.)
Z uses:

- **Declarative semantics** – what the system should do
- Describes **preconditions, postconditions, and invariants** in **schemas**

  ➡ Comparable to how **VDM-SL** uses invariants and state.

3. Types of Specifications:

VDM-SL supports:

1. **State-based Specification** – using state definitions, invariants
2. **Functional Specification** – explicit/implicit functions
3. **Algebraic Specification** – using abstract equations (indirectly supported)
4. **Operational Specification** – operations that change state

➡ *[Slides – Topics#058–066]*

5. Algebraic Specifications:

- Algebaric specifications often describe a collection of a basic data-types in a property oriented way, then use these to build a model of the system being specified.

In simple terms, algebraic specifications describe systems by defining basic data types (like numbers, sets, or lists) and their properties (what the data can do or how it behaves). Then, these data types are combined to build a model that represents the whole system.

➡ *[Slides – Topics: Specifying Functions, Composite Types]*

✅Short Questions

1. Set of Operations (in VDM-SL):

By Maha Rana

VDM-SL supports the following **set operations**:

- **Union (∪)**
- **Intersection (∩)**
- **Difference (\ )**
- **Membership (∈) and Non-membership (∉)**
- **Subset (⊆)**
- **Equality (=)**
- **Cardinality (card)**

  ➡ *[Slides – Topic: Set Operations]*

2. Explicitly ke 2 Functions (VDM-SL):

Explicit functions define **how** output is calculated.
Examples:

1. `add(x, y) == x + y`
2. `square(n) == n * n`

   ➡ *[Slides – Topic#063: Explicit Function]*

3. Sequence Definition ke 3 Kinds:

1. **Normal sequence** – `[a, b, c]`
2. **Empty sequence** – `[]`
3. **Repeating/duplicate sequence** – `[a, b, a]`

   ➡ *[Slides – Sequence Definition Examples]*
4. **Model-Oriented Method ke Names:**

Model-oriented methods

- The closest method to Z is the "Vienna Development Method (VDM)", which originated at the IBM Vienna Laboratory, and has been developed in the work of Dines Bjorner and Cliff Jones.
- In their aims, VDM and Z are quite similar, but there are a number of differences of style which give each method its own advantages and disadvantages.

Z and other methods

- Styles are divided into model-oriented methods
- Aim of a specification is to construct an abstract model of the information system being specified.
- And property-oriented or algebaric methods, where the aim is to describe a system in terms of its desired properties, without constructing and explicit model.

- ➡ *[Slides mention VDM; Z is implied as another method]*

5. Truth Table for $p \wedge \neg p$:

By Maha Rana

p	¬p	p ∧ ¬p
T	F	F
F	T	F

➡️ This is always **False**, hence a **contradiction**.

➡️ *[Slides – Propositional Logic – Tautologies & Contradictions]*

6. Which Operator Joins Two Sequences?

Answer: **Concatenation Operator (^)**

```
[a, b] ^ [c, d] = [a, b, c, d]
```

➡ *[Slides – Sequence Operators → Concatenation]*

✅Long Questions

1. Components of VDM-SL:

Components of VDM-SL

As we studied, the components of a complete VDM-SL specification are:

- Declaration of constants
- State definition
- Functions
- Operations

2. ➡ *[Slides – Topic#066 & #071: Template + Specification Structure]*

2. Predicate Logic with Examples:

By Maha Rana

- A **predicate** is a truth-valued expression involving variables.
- Variables are **free** until **bound** by substitution or quantification.

Examples:

- `C(x)` → x is a cat
- `Prime(n)` → n is a prime number
- `Studies(x, y)` → x studies y

With Substitution:

- `C(Simba)` → Simba is a cat (TRUE or FALSE)

With Quantifiers:

- Universal: ∀`x ∈ Cats ● C(x)` → All x are cats
- Existential: ∃`n ∈ ℕ ● Prime(n)` → Some n is prime
- Unique Existential: ∃`! x ∈ Cats ● G(x)` → Exactly one cat is green

  ➡ *[Slides – Predicate Logic: Topics on Substitution, Quantification, Examples]*

3. Robustness ke Points (VDM-SL):

Robust specifications ensure the **system behaves safely and reliably**. Key points:

- **Completeness:** The method should cover all essential behaviors and requirements of the system.
- **Scalability:** It should work well for both small and large systems without becoming slow or impractical
- **Expressiveness:** It should provide rich language features to accurately represent different system behaviors
- **Automated:** Automated tools should help with model checking and verification, reducing human errors.
- **Soundness and Completeness of Analysis**: Any conclusion drawn from the method should be theoretically correct and explore all relevant aspects of the system.

➡ *[Slides – Topics#064, #065: Robustness via Invariant, Init, Pre/Postconditions]*

✅Short Questions

1. Union (Set Operator):

Definition: Union combines all unique elements from two sets.

 Symbol: ∪
 Example:

```
j = {<MON>, <TUE>}, k = {<TUE>, <FRI>}
j ∪ k = {<MON>, <TUE>, <FRI>}
```

➡ *[Slide Ref: Set Operations – Union]*

2. Model-Oriented Method:

Definition: Focuses on modeling the **state and operations** of a system explicitly.

 Examples:

- **VDM (Vienna Development Method)**
- **Z language**

➡ *[Slide Ref: VDM-SL topics]*

3. Algebraic Model:

Definition: Focuses on defining system behavior using **algebraic equations** and **data types**.

 Examples:

- Equational specifications
- Abstract Data Types

➡ Not deeply covered in slides, but related to Z/VDM implicitly.

By Maha Rana

4. Two Functions of VDM-SL:

VDM-SL allows defining:

- **Explicit Functions**: Show exact computation
  → `add(x, y) == x + y`
- **Implicit Functions**: Use pre/postconditions
  → `add(x,y) pre TRUE post result = x+y`

➡ *[Slide Ref: Topic#063]*

5. Purpose of `inRange`:

Checks whether a number is within the **valid range** (like temperature limits).

 Example:

```
inRange(val) == (val ≥ MIN and val ≤ MAX)
```

➡ *[Slide Ref: Topic#069]*

6. Completeness of Formal Method:

Ensures all system behavior is **completely and clearly defined** using math logic, leaving **no ambiguity**.

➡ *VDM-SL achieves this through: types, invariants, init, operations.*

7. Contradiction Proof (`p ∧ ¬p`):

By Maha Rana

p	¬p	p ∧ ¬p
T	F	F
F	T	F

📌 Always **false** → So it is a **contradiction**.

➡️ *[Slide Ref: Tautology & Contradiction]*

3. Robust Monitoring System:

- **Completeness:** The method should cover all essential behaviors and requirements of the system.
- **Scalability:** It should work well for both small and large systems without becoming slow or impractical
- **Expressiveness:** It should provide rich language features to accurately represent different system behaviors
- **Automated:** Automated tools should help with model checking and verification, reducing human errors.
- **Soundness and Completeness of Analysis**: Any conclusion drawn from the method should be theoretically correct and explore all relevant aspects of the system.

9. How to Identify a Block in a Disk (MCQ + Short):

By Maha Rana

DiskScanner System

- software designed to keep track of damaged blocks on the surface of a disk.
- *A disk is divided into a number of tracks*
- *each track into a number of sectors.*
- *block = track + sector number*
- *DiskScanner class*

```
types
Block : : track: ℕ
sector: ℕ

state DiskScanner of
        damagedBlocks: Block-set

init mk-DiskScanner (dB) △ dB = { }
```

Block
track: Integer *sector: Integer*

✓Long Questions

1. Components of VDM-SL:

Components of VDM-SL

As we studied, the components of a complete VDM-SL specification are:

- Declaration of constants
- State definition
- Functions
- Operations

By Maha Rana

➡ *[Slide Ref: Topic#066 & #071]*

2. Symbol Matching Table:

Logic Operator Symbol

Logic Operator	Symbol
Negation	¬
AND	∧
OR	∨
Exclusive OR (XOR)	⊕
Implication	→

➡ *[Slide Ref: Propositional Logic – Operators]*

3. Role of Well-Formedness:

Well-formedness ensures that a formal specification is:

- **Logically correct**
- **Complete**
- **Free of contradictions**
- **Safe and consistent**
  In VDM-SL, this is achieved through:
- State invariant
- Type constraints
- Initialization
- Pre/Postconditions

➡ *[Slide Ref: Topics on Invariant, Init, Pre/Postconditions]*

By Maha Rana

May Allah grant you success, ease your efforts, and bless you with wisdom and perseverance. Keep your faith strong, for with prayer, every challenge becomes an opportunity. And please remember me in your prayers.

I hope this file helps you a lot; I have tried my best to find the most relevant and authenticated answers for the questions. However, in some cases, the questions might slightly differ from the ones in your slides. I recommend that every student kindly verify the answers on their own before using them. I am not responsible for any discrepancies.

Best wishes for Finals!