

6.2 Data Integrity

Name: _____ Class: _____ Date: _____ Total: 37 marks

KEY WORDS verification • validation • check digit • data integrity • checksum

Objective

Build the skills to answer exam questions on **data integrity** – how **validation** and **verification** protect it, and the methods used for each.

You must be able to:

- explain how **validation** and **verification** protect data integrity
- describe and use methods of **validation**
- describe and use methods of **verification** during data entry and data transfer

1 Key knowledge and worked examples

Study these first. Each one shows the method for a question type used later – follow the steps and you can do the Practice and Exam-style questions yourself.

■ Key knowledge: use these earlier methods

- A field stores one attribute; a record stores the fields for one item. A primary key uniquely identifies a record; a foreign key refers to a primary/candidate key in another table.
- **Referential integrity** prevents a non-null foreign-key value referring to a missing parent record. Validation alone does not prove that a field is factually correct.
- A client–server system keeps shared data and access control on the server. Describe the integrity measure, not merely the network device.

■ Validation vs verification

- **Validation** asks “*is this data sensible?*” – automatic checks before storing.
- **Verification** asks “*was this data copied/entered correctly?*” – confirms it was not changed.

Use both: validation catches nonsense; verification catches copying errors. Neither proves the data is *factually* correct (a sensible, correctly-typed value can still be wrong).

■ Validation methods

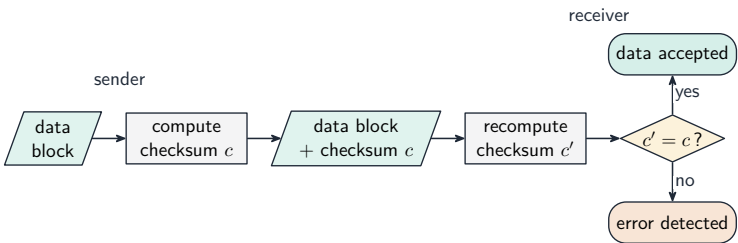
Check	Rejects...	Example
range	values outside limits	a month not in 1–12
length	wrong number of characters	a 5-digit phone number
type / character	wrong kind of data	a letter in an age field
format	wrong pattern	an email with no @
presence	empty required fields	a blank surname
check digit	mistyped long numbers	a wrong ISBN / card number

■ Verification methods

- **On entry: double entry** (type it twice and compare, like a new password) or a visual check.
- **On transfer** (bits can flip): a **parity check** adds a bit so the number of 1s is even (or odd); the receiver re-counts.

7 data bits							parity
0	1	1	0	1	0	0	1

data has three 1s; for **even parity** the parity bit is **1** (four 1s in total)



worked example – checksum = (sum of bytes) mod 256

1	sum of all bytes: $X = 1185$
2	$X \div 256 = 4.629$, round down: $Y = 4$
3	$Z = Y \times 256 = 1024$
4	checksum = $X - Z = 1185 - 1024 = 161$

A *checksum* is the other main data-integrity check

A **checksum** sends a summary value the receiver recomputes and compares.

■ Check digit

A **check digit** is an extra digit added to the end of a long number (ISBN, barcode, card number) so that a mistyped digit can be spotted.

Worked example (modulo 11). For the digits 4 2 1 3, multiply each by a weight (here 5, 4, 3, 2) and add: $4 \times 5 + 2 \times 4 + 1 \times 3 + 3 \times 2 = 37$. The check digit is chosen so the grand total – with the check digit itself weighted 1 – is a **multiple of 11**: $37 + c \equiv 0 \pmod{11}$, so $c = 7$. The full number is **42137**.

To **verify** a number, repeat the weighted sum including the check digit; if the total is a multiple of 11, the number is probably correct.

■ Key knowledge: the question chooses the check-digit rule

- **Unweighted sum modulo 10:** add the original digits, take the remainder on division by 10, and append it. For 5271, $c = (5 + 2 + 7 + 1) \bmod 10 = 5$, giving 52715.
- **Limit:** swapping digits leaves their sum unchanged. Different digit errors can also cancel. A matching check digit does not prove that the data is correct.
- **Weighted schemes:** positions have different weights. They detect some errors that the unweighted sum misses; detection depends on the weights and modulus, so never promise that every transposition is detected.
- Verification applies the **same specified rule** again and compares the expected digit with the supplied digit. Follow the rule in the question, not a memorised ISBN rule.

2 Practice

Now apply the methods above.

2.1 State the difference between **validation** and **verification**. [2]

2.2 Name the validation check for each: (a) a month must be 1–12 (b) an email must contain @. [2]

Month

Email

2.3 Describe how **double-entry** verification works. [1]

2.4 A byte uses **even parity**. The seven data bits are 1011001. State the parity bit. [1]

2.5 Using the sum-of-digits modulo-10 rule, find the stored number for 6382. Would entering 36829 pass? Explain. [3]

number check digit

entered

3 Exam-style questions

3.1 Describe **three** validation checks, giving an example of each. [3]

Name

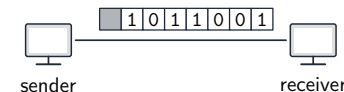
Date of birth

Phone

3.2 Explain why validation **cannot guarantee** that stored data is correct. [2]

Date of birth

3.3 (a) Describe how a **parity check** detects an error during transmission. [2]



(b) State one limitation of a parity check. [1]

3.4 A new user sets a password by typing it into two boxes. State which technique this is and what it checks. [2]

Password

Again

3.5 A 4-digit code 5 1 3 2 uses a modulo-11 check digit. Each digit is multiplied by a weight (5, 4, 3, 2) and the check digit (weight 1) makes the total a multiple of 11. Calculate the **check digit**. Show your working. [3]

digit

5

1

3

2

weight

5

4

3

2

1

3.6 A membership number is a five-digit integer. When a number is typed in, the system checks it before use.

(a) **Draw one line** from each method to say whether it is used during **data entry** or during **data transfer**. [2]

method	used during
double entry	data entry
parity check	
visual check	data transfer
checksum	

(b) **State** the difference between **validation** and **verification**. [2]

member

(c) A check digit is the sum of the original digits modulo 10. The stored number **41623** is typed as **14623**. **Explain** why this check digit does not detect the transposition, and **state** a verification method that could detect it. [3]

(d) A **check digit** is added to make a six-digit number. **Describe** how a check digit is calculated and used. [3]

(e) **Write** pseudocode for a function **Generate(Original)** that takes a five-digit integer and returns it with a check digit appended, where the check digit is the sum of the five digits modulo 10. [5]