A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

6.2 Data Integrity

Name: _____ Class: _____ Date: _____ **Total: 23 marks**

Objective

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

You must be able to:

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

1 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.

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).

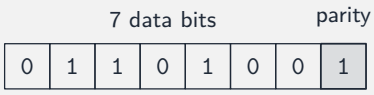
Validation methods

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

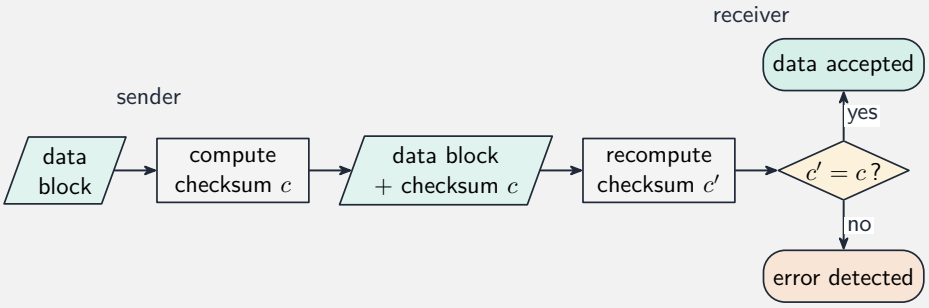
Verification methods

- **On entry: double entry** (type it twice and compare, like a new password) or a visual check.

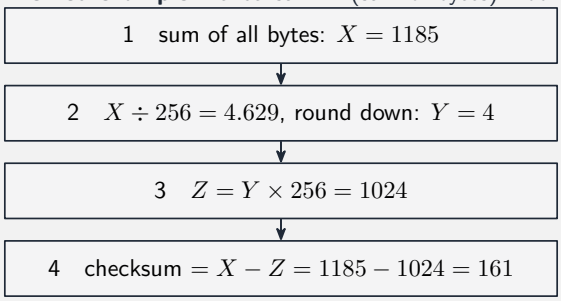
- **On transfer** (bits can flip): a **parity check** adds a bit so the number of 1s is even (or odd); the receiver re-counts.



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



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



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.

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]

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]

3 Exam-style questions

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

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

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]

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]

4 Go further

You are now ready for the real exam questions on this subtopic. Open the **6.2 Data Integrity** past-paper sheet in the Library, or try these in **Practice mode**:

- 9618/13 N24 —Q4 (check-digit calculation and verification)
- 9618/12 N25 —Q6 (parity and error checking)
- 9618/11 J25 —Q7 (data integrity and security)
- 9618/13 N25 —Q7 (validation and verification)

Solutions

2.1 Validation checks the data is **sensible** (against rules) before storing; verification checks the data was **entered/transferred correctly** (not changed).

2.2 (a) **range** check. (b) **format** check.

2.3 The data (e.g. password) is entered **twice** and the two copies are compared; if they differ, there was a typing error.

2.4 `1011001` has **four** 1s (already even), so the parity bit is **0**.

3.1 Any three with examples, e.g.: range (month 1–12); length (a 6-character code); type (digits only in a phone field); format (email contains `@`); presence (required field not blank).

3.2 Validation only checks the data is the right **format/range**; a value can be sensible and correctly formatted yet **factually wrong** (e.g. "Bob" instead of "Bib"), which validation cannot detect.

3.3 (a) A parity bit makes the number of 1s even (or odd) when sent; the receiver recounts the 1s; if the count no longer matches, a bit was flipped —an error is detected.
(b) It misses an **even number** of flipped bits (e.g. two), which leave the parity unchanged.

3.4 Double-entry verification; it checks the two typed copies match, catching a typing error before the password is saved.

3.5 Weighted sum $= 5\times5 + 1\times4 + 3\times3 + 2\times2 = 25 + 4 + 9 + 4 = 42$; $42 \bmod 11 = 9$, so the check digit $= 11 - 9 = 2$. Check: $42 + 2 = 44 = 4 \times 11$.