Exercise walk-through

Data Integrity ▪ 6.2

eXercise

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

Method — 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).

Method — Validation methods

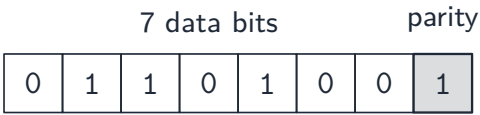
Check	Rejects...	Example
range	values outside limits	a month not in 1–12
length	wrong number of characters	a 5-digit phone number
type / character	wrong kind of data	a letter in an age field
format	wrong pattern	an email with no `@`
presence	empty required fields	a blank surname
check digit	mistyped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

Method — 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.

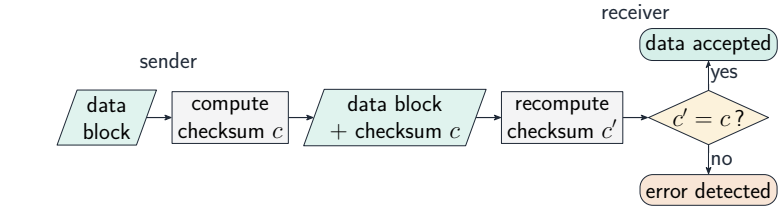
A **checksum** sends a summary value the receiver recomputes and compares.

Method — Verification methods



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

Method — Verification methods



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

Method — 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: $4\times5+2\times4+1\times3+3\times2=37$. 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.

Practice 2.1 ▪ 2 marks

State the difference between **validation** and **verification**.

Solution 2.1

Method: Validation vs verification

Worked steps

Validation checks the data is **sensible** (against rules) before storing; verification checks the data was **entered/transferred correctly** (not changed) B1 B1.

Practice 2.2 ▪ 2 marks

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

Solution 2.2

Method: Validation methods

Worked steps

(a) **range** check B1. (b) **format** check B1.

Practice 2.3 ▪ 1 mark

Describe how **double-entry** verification works.

Solution 2.3

Method: Verification methods

Worked steps

The data (e.g. password) is entered **twice** and the two copies are compared; if they differ, there was a typing error B1.

Practice 2.4 ▪ 1 mark

A byte uses **even parity**. The seven data bits are `1011001`. State the parity bit.

Solution 2.4

Method: Verification methods

Worked steps

1011001 has **four** 1s (already even), so the parity bit is **0** B1.

Exam-style 3.1 ▪ 3 marks

Describe **three** validation checks, giving an example of each.

Solution 3.1

Worked steps

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) B1 B1 B1.

Exam-style 3.2 ▪ 2 marks

Explain why validation **cannot guarantee** that stored data is correct.

Solution 3.2

Method: Validation vs verification

Worked steps

Validation only checks the data is the right **format/range** B1; a value can be sensible and correctly formatted yet **factually wrong** (e.g. “Bob” instead of “Bib”), which validation cannot detect B1.

Exam-style 3.3 ▪ 2 marks

(a) Describe how a **parity check** detects an error during transmission.
(b) State one limitation of a parity check.

Solution 3.3

Method: Verification methods

Worked steps

(a) A parity bit makes the number of 1s even (or odd) when sent; the receiver recounts the 1s B1; if the count no longer matches, a bit was flipped — an error is detected B1.

(b) It misses an **even number** of flipped bits (e.g. two), which leave the parity unchanged B1.

Exam-style 3.4 ▪ 2 marks

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

Solution 3.4

Worked steps

Double-entry verification B1; it checks the two typed copies match, catching a typing error before the password is saved B1.

Exam-style 3.5 ▪ 3 marks

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.

Solution 3.5

Method: Check digit

Worked steps

Weighted sum $= 5{\times}5{+}1{\times}4{+}3{\times}3{+}2{\times}2 = 25{+}4{+}9{+}4 = 42$ M1; $42 \bmod 11 = 9$, so the check digit $= 11 - 9 = 2$ M1 A1. Check: $42 + 2 = 44 = 4 \times 11$.

Exam-style 3.6 ▪ 2 marks

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**.

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

Exam-style 3.6 ▪ 2 marks

(b) **State** the difference between **validation** and **verification**.
(c) The number `41623` is typed as `14623` — the **first two digits** were transposed. **State** which single check would catch this, and **explain** why a **range check** would not.
(d) A **check digit** is added to make a six-digit number. **Describe** how a check digit is calculated and used.
(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.

Solution 3.6

Method: Verification methods

Worked steps

(a) **Data entry**: double entry and visual check B1. **Data transfer**: parity check and checksum B1.
(b) **Validation** is an automatic check that data is **reasonable** — the right type, in range, the right format B1. **Verification** checks that data has been **copied or transmitted accurately**, that it matches the source B1.
(c) A **check digit** B1, because transposing two digits changes the weighted sum and so changes the digit that should follow B1. A range check passes both values: 14623 and 41623 are each five-digit numbers within the allowed range, so nothing about the range is violated B1.
(d) The check digit is calculated from the other digits by a fixed rule — often a weighted sum reduced modulo a number B1 — and appended to them B1. Whenever the number is entered or received, the rule is applied again to the leading digits and the result compared with the digit that was carried; a

Solution 3.6

```
FUNCTION Generate(Original : INTEGER) RETURNS INTEGER
    DECLARE Total, Temp, CheckDigit : INTEGER
    Total ← 0
    Temp ← Original
    WHILE Temp > 0
        Total ← Total + (Temp MOD 10)
        Temp ← Temp DIV 10
    ENDWHILE
    CheckDigit ← Total MOD 10
    RETURN Original * 10 + CheckDigit
ENDFUNCTION
```

Solution 3.6

B1 B1 B1 B1 B1 *header with parameter and return type; the digit-extraction loop using 'MOD 10' and 'DIV 10'; the running total; 'CheckDigit ← Total MOD 10'; appending it with '* 10 +'*