Exercise walk-through

6.2
Data Integrity

A-Level Computer Science

A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

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.

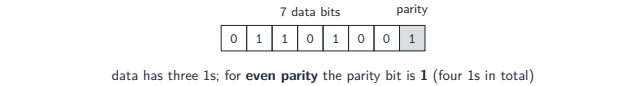
Use both: validation catches nonsense; verification catches copying errors. Neither proves the data is *factually* correct (a sensible, correctly-typed value can still be wrong).

Validation methods

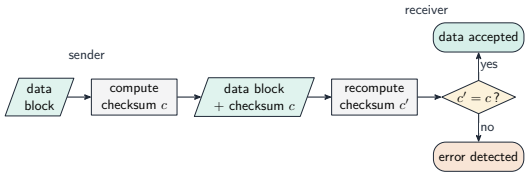
Check	Rejects...	Example
range	values outside limits	a month not in 1–12
length	wrong number of characters	a 5-digit phone number
type / character	wrong kind of data	a letter in an age field
format	wrong pattern	an email with no `@`
presence	empty required fields	a blank surname
check digit	mistyped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

Verification methods

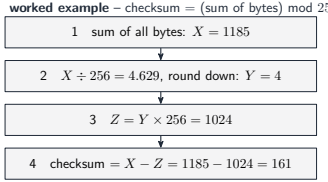
- **On entry: double entry** (type it twice and compare, like a new password) or a visual check.
- **On transfer** (bits can flip): a **parity check** adds a bit so the number of 1s is even (or odd); the receiver re-counts.



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.

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.

Exercises 2.1·2.2

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

validation verification

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

Month

Email

Exercises 2.1·2.2

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

validation verification

2.1

- validation checks the data is **sensible** (against rules) before storing
- verification checks the data was **entered/transferred correctly** (not changed)

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

Month

Email

2.2

(a) **range** check

(b) **format** check

Exercises 2.3·2.4

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]

	1	0	1	1	0	0	1

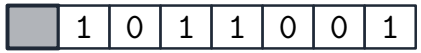
Exercises 2.3·2.4

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

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



2.4

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

Exercise 2.5

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

Exercise 2.5

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

2.5

- $c = (6 + 3 + 8 + 2) \bmod 10 = 9$, so the stored number is `63829`.
- `3682` has the same digit sum, 19.
- It also gives check digit 9, so `36829` passes even though the first two digits are swapped.

Exercises 3.1·3.2

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 12/03/2009

Exercises 3.1·3.2

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

Name
Date of birth
Phone

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

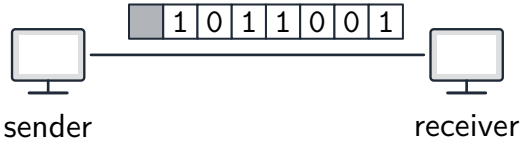
Date of birth 12/03/2009

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

Exercise 3.3

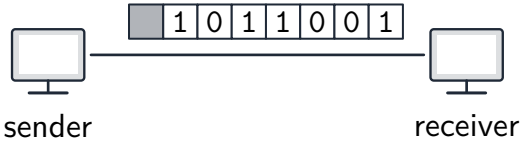
3.3 (a) Describe how a **parity check** detects an error during transmission. [2]



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

Exercise 3.3

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



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

3.3

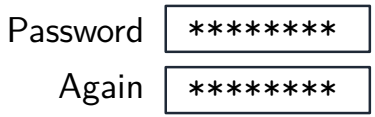
(a) a parity bit makes the number of 1s even (or odd) when sent; the receiver re-counts the 1s

- 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

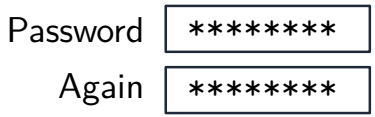
Exercise 3.4

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



Exercise 3.4

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



3.4

- **double-entry** verification
- it checks the two typed copies match, catching a typing error before the password is saved

Exercise 3.5

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

Exercise 3.5

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

3.5

- weighted sum of the digits

$$5 \times 5 + 1 \times 4 + 3 \times 3 + 2 \times 2 = 25 + 4 + 9 + 4 = 42$$

- remainder modulo 11, then 11 minus that remainder

$$42 \bmod 11 = 9 \Rightarrow \text{check digit} = 11 - 9 = \mathbf{2}$$

- check: $42 + 2 = 44 = 4 \times 11$

Exercise 3.6 (a)

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	

Exercise 3.6 (a)

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	

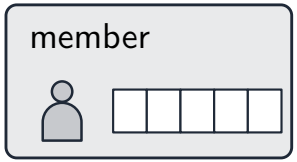
(a) **data entry**: double entry and visual check

- **data transfer**: parity check and checksum

Exercise 3.6 (b)

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

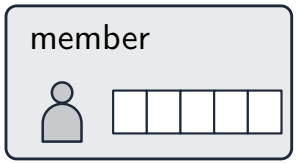
(b) **State** the difference between **validation** and **verification**. [2]



Exercise 3.6 (b)

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

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



(b) **validation** is an automatic check that data is **reasonable** – the right type, in range, the right format

- **verification** checks that data has been **copied or transmitted accurately**, that it matches the source

Exercise 3.6 (c)

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

(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]

Exercise 3.6 (c)

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

(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]

(c) the digit sum is unchanged: $4 + 1 + 6 + 2 = 1 + 4 + 6 + 2 = 13$

- both give remainder 3 when divided by 10, so both match the stored check digit
- **double entry**, comparing a correct second entry with the first, or a visual comparison with the original source could detect it

Exercise 3.6 (d)

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

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

Exercise 3.6 (d)

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

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

(d) the check digit is calculated from the other digits by a fixed rule – often a weighted sum reduced modulo a number

- it is appended to them
- whenever the number is entered or received, the rule is applied again to the leading digits and compared with the carried digit; a mismatch means an error

Exercise 3.6 (e)

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

(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]

Exercise 3.6 (e)

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

(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]

(e)

```
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
```

Full solution on next slide

Full solution 3.6 (e)

(e)

```
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
```

- header with parameter and return type
- digit-extraction loop using `MOD 10` and `DIV 10`
- running total

- `CheckDigit ← Total MOD 10`
- appending it with `* 10 +`