A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

11.1 Programming Basics

Name: _____ Class: _____ Date: _____ **Total: 35 marks**

KEY WORDS pseudocode 伪代码 • variables 变量 • constant 常量 • function 函数 • operators 运算符

Objective

Build the skills to answer exam questions on **programming basics** 编程基础 — constants, variables, operators and built-in functions.

You must be able to:

- declare **constants** and **variables**, and use **assignment**
- evaluate expressions with arithmetic, comparison and logic **operators** (including `DIV` and `MOD`)
- use **built-in functions** such as `LENGTH`, `LEFT`, `MID`, `UCASE`

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.

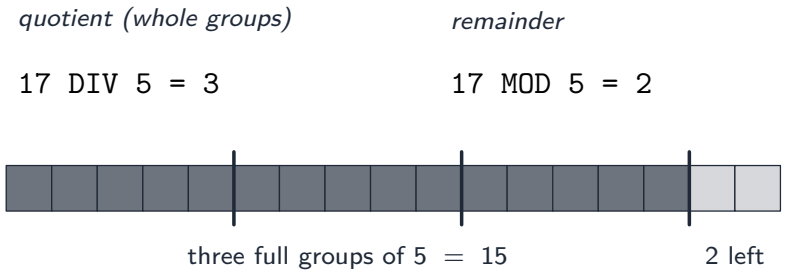
■ Constants and variables

A **constant** holds a value that never changes; a **variable** holds one that can. Declare with a type, assign with ←:

```
CONSTANT Pi ← 3.14159
DECLARE Radius : REAL
Radius ← 5
Area ← Pi * Radius * Radius
```

■ Operators: DIV and MOD

`DIV` is the **whole-number quotient**; `MOD` is the **remainder**:



Precedence (do first → last): `NOT` → `* / DIV MOD` → `+ -` → comparisons → `AND` → `OR`. So `3 + 4 * 2` = `3 + 8` = `11`. Use brackets when unsure.

Built-in functions

Function	Example	Result
`LENGTH(s)`	`LENGTH("Computer")`	`8`
`LEFT(s, n)`	`LEFT("Computer", 3)`	`"Com"`
`MID(s, start, len)`	`MID("Computer", 4, 3)`	`"put"`
`UCASE(s)`	`UCASE("hi")`	`"HI"`
`INT(x)`	`INT(3.9)`	`3`

`MID` counts characters from **1**. Use the exact names from the paper's reference list.

2 Practice

Now apply the methods above.

2.1 Write pseudocode to declare a **constant** `Pi` of `3.14159` and a **real** variable `Radius`. [2]

2.2 Evaluate `23 DIV 4` and `23 MOD 4`. [2]

2.3 Evaluate `3 + 4 * 2` (use precedence). [1]

2.4 State the result of `LENGTH("Computer")` and `LEFT("Computer", 3)`. [2]

3 Exam-style questions

3.1 Write pseudocode that inputs a `Price`, adds **20% tax**, and outputs the total. Use a **constant** for the tax rate. [4]

3.2 Evaluate each expression. [5]

Expression	Value
`7 DIV 2`	
`7 MOD 2`	
`(5 > 3) AND (2 > 4)`	
`NOT (1 = 1)`	
`MID("PROGRAM", 2, 3)`	

3.3 Write pseudocode that reads a word and outputs it in **capitals** followed by its **length**, using built-in functions. [3]

3.5 A program stores the marks of 30 students in a 1D array `Mark`, and reports how many passed (a mark of 50 or more).

(a) **Complete** the table by giving an appropriate data type and an example value for each variable. [4]

variable	purpose	data type	example value
`Mark[1]`	one student's mark	_____	_____
`PassRate`	percentage that passed	_____	_____
`TopName`	name of the best student	_____	_____
`AllPassed`	did everyone pass?	_____	_____

(b) **Write** pseudocode that counts the passes and calculates `PassRate`. [5]

(c) A programmer initialises the counter with `Count ← 0` before the loop. **Explain** what would happen if this line were left out. [2]

(d) **Describe** how an **IDE** helps a programmer locate a fault during **alpha testing**. [3]

(e) **State** the difference between **alpha** and **beta** testing. [2]

Solutions

2.1

```
CONSTANT Pi ← 3.14159
DECLARE Radius : REAL
```

2.2 `23 DIV 4 = 5`; `23 MOD 4 = 3`.

2.3 `11` — multiply first (`4 * 2 = 8`), then add 3.

2.4 `LENGTH("Computer") = 8`; `LEFT("Computer", 3) = "Com"`.

3.1

```
CONSTANT TaxRate ← 0.20
DECLARE Price : REAL
DECLARE Total : REAL
INPUT Price
Total ← Price * (1 + TaxRate)
OUTPUT "Total: ", Total
```

3.2 `7 DIV 2 = 3`; `7 MOD 2 = 1`; `(5 > 3) AND (2 > 4) = FALSE`; `NOT (1 = 1) = FALSE`; `MID("PROGRAM", 2, 3) = "ROG"`.

3.3

```
OUTPUT "Enter a word:"
INPUT Word
OUTPUT UCASE(Word)
OUTPUT "Length: ", LENGTH(Word)
```

3.5 (a) `Mark[1]` — `INTEGER`, e.g. `67`; `PassRate` — `REAL`, e.g. `73.3`; `TopName` — `STRING`, e.g. `"Wei"`; `AllPassed` — `BOOLEAN`, e.g. `FALSE`.
(b)

```
Count ← 0
FOR Index ← 1 TO 30
    IF Mark[Index] >= 50
      THEN
        Count ← Count + 1
    ENDIF
NEXT Index
PassRate ← Count / 30 * 100
OUTPUT PassRate
```

(c) The counter would hold whatever value happened to be in that memory location, or the program would fail with an "unassigned variable" error; either way the pass count and the rate would be wrong, and the error might not show every time it runs.
(d) An IDE provides a **debugger**: breakpoints stop the program at a chosen line; single-stepping runs one statement at a time; a watch window shows the changing value of each variable, so the programmer sees exactly where the value first goes wrong.
(e) **Alpha** testing is done **inside** the development organisation, on incomplete software,

by staff; **beta** testing releases the near-finished software to selected **external users**, who report faults found in real use.