Exercise walk-through

Programming Basics ▪ 11.1

eXercise

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`

Method — 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
```

Method — 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.

Method — Operators: DIV and MOD

quotient (whole groups)

```
17 DIV 5 = 3
```

remainder

```
17 MOD 5 = 2
```

three full groups of 5 = 15

2 left

Method — 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.

Practice 2.1 ▪ 2 marks

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

Solution 2.1

Method: Constants and variables

Worked steps

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

Solution 2.1

B1 B1 *'CONSTANT' with value, 'DECLARE ... : REAL'*

Practice 2.2 ▪ 2 marks

Evaluate `23 DIV 4` and `23 MOD 4`.

Solution 2.2

Worked steps

`23 DIV 4 = 5; 23 MOD 4 = 3` B1 B1.

Practice 2.3 ▪ 1 mark

Evaluate `3 + 4 * 2` (use precedence).

Solution 2.3

Worked steps

11 — multiply first (`4 * 2 = 8`), then add 3 B1.

Practice 2.4 ▪ 2 marks

State the result of `LENGTH("Computer")` and `LEFT("Computer", 3)`.

Solution 2.4

Method: Built-in functions

Worked steps

`LENGTH("Computer") = 8`; `LEFT("Computer", 3) = "Com"` B1 B1.

Exam-style 3.1 ▪ 4 marks

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

Solution 3.1

Worked steps

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

Solution 3.1

M1 M1 M1 A1 *constant for tax, 'INPUT Price', correct calculation, 'OUTPUT' the total*

Exam-style 3.2 ▪ 5 marks

Evaluate each expression.

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

Solution 3.2

Worked steps

`7 DIV 2 = 3`; `7 MOD 2 = 1`; `(5 > 3) AND (2 > 4) = FALSE`; `NOT (1 = 1) = FALSE`; `MID("PROGRAM", 2, 3) = "ROG"` **B1** *each*.

Exam-style 3.3 ▪ 3 marks

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

Solution 3.3

Method: Built-in functions

Worked steps

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

Solution 3.3

M1 M1 A1 *'INPUT', 'UCASE(Word)', 'LENGTH(Word)' output*

Exam-style 3.5 ▪ 4 marks

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.

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?	_____	_____

Exam-style 3.5 ▪ 4 marks

(b) **Write** pseudocode that counts the passes and calculates `PassRate`.
(c) A programmer initialises the counter with `Count ← 0` before the loop. **Explain** what would happen if this line were left out.
(d) **Describe** how an **IDE** helps a programmer locate a fault during **alpha testing**.
(e) **State** the difference between **alpha** and **beta** testing.

Solution 3.5

Method: Built-in functions

Worked steps

(a) `Mark[1]` — `INTEGER`, e.g. `67` B1; `PassRate` — `REAL`, e.g. `73.3` B1; `TopName` — `STRING`, e.g. `"Wei"` B1; `AllPassed` — `BOOLEAN`, e.g. `FALSE` B1.

(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
```

Solution 3.5

B1 B1 B1 B1 B1 *initialise the counter; loop over all 30; correct condition; increment inside the 'IF'; percentage calculated after the loop*

(c) The counter would hold whatever value happened to be in that memory location, or the program would fail with an "unassigned variable" error B1; either way the pass count and the rate would be wrong, and the error might not show every time it runs B1.

(d) An IDE provides a **debugger**: breakpoints stop the program at a chosen line B1; single-stepping runs one statement at a time B1; a watch window shows the changing value of each variable, so the programmer sees exactly where the value first goes wrong B1.

(e) **Alpha** testing is done **inside** the development organisation, on incomplete software, by staff B1; **beta** testing releases the near-finished software to selected **external users**, who report faults found in real use B1.