

Please check the examination details below before entering your candidate information

|                      |                      |                      |                      |
|----------------------|----------------------|----------------------|----------------------|
| Candidate surname    |                      | Other names          |                      |
| Centre Number        |                      | Candidate Number     |                      |
| <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> |

**Pearson Edexcel International GCSE (9–1)**

**Monday 2 – Wednesday 4 June 2025**

Time 3 hours

Paper reference **4CP0/2AW**

**Computer Science (Python)**

**PAPER 2: Application of Computational Thinking**

**You must have:** A computer workstation with appropriate programming language code editing software and tools, including a code interpreter/compiler, CODES folder containing code and data files, and pseudocode command set (enclosed)

Total Marks

### Instructions

- Use **black** ink or ball-point pen.
- **Fill in the boxes** at the top of this page with your name, centre number and candidate number.
- Answer **all** questions.
- Answer the questions **requiring a written answer** in the spaces provided – *there may be more space than you need.*
- Only Python must be used throughout the examination.
- Carry out practical tasks on the computer system and save new or amended code using the name given in the question with the appropriate file extension.
- Do **not** overwrite the original code and data files provided to you.
- You must **not** use the internet during the examination.

### Information

- The total mark for this paper is 80.
- The marks for **each** question are shown in brackets – *use this as a guide as to how much time to spend on each question.*
- This paper covers Python.
- The CODES folder in your user area includes all the code and data files you need.
- The invigilator will tell you where to store your work.

### Advice

- Read each question carefully before you start to answer it.
- Save your work regularly.
- Check your answers if you have time at the end.

Turn over ►

P78639A

©2025 Pearson Education Ltd.  
Y:1/1/1/1/1/1/



  
Pearson

**Answer all questions.**

**Answer the questions requiring a written answer in the spaces provided.**

**Some questions must be answered with a cross in a box ☒. If you change your mind about an answer, put a line through the box ☒ and then mark your new answer with a cross ☒.**

**Carry out practical tasks on the computer system and save new or amended code using the name given with the appropriate file extension.**

**Use only Python throughout the examination.**

**1** Programmers use different programming constructs to create working code.

(a) Identify the keyword used to create a selection statement.

(1)

- ☐ **A** for
- ☐ **B** if
- ☐ **C** new
- ☐ **D** while

(b) Computer code can contain errors.

(i) Complete the table by adding a tick (✓) to match each error description to a type of error.

(3)

| Description                                       | Syntax error | Logic error | Runtime error |
|---------------------------------------------------|--------------|-------------|---------------|
| The program does not produce the expected output. |              |             |               |
| The program does not translate.                   |              |             |               |
| The program crashes during execution.             |              |             |               |



- (ii) A program counts the number of green and red balls stored in an array.

The program outputs the number of green balls, the number of red balls and the total number of balls.

**Figure 1** shows the expected output from the program.

|                 |
|-----------------|
| Green: 25 balls |
| Red: 15 balls   |
| Total: 40 balls |

**Figure 1**

Open **Q01bii** in the code editor.

There are **three** errors in the code.

Amend the code to correct the errors.

Save your amended code as **Q01biiFINISHED** with the correct file extension for the programming language.

(3)

- (c) Programs use constants, variables and data structures.

- (i) Give **one** reason for using a constant.

(1)

.....

.....

- (ii) Describe **one** difference between a variable and an array.

(2)

.....

.....

.....

.....

- (d) A program is used to calculate wages for sales staff. Staff can earn 5% commission based on their age and sales.

Open **Q01d** in the code editor.

Use the code to answer these questions.

- (i) Identify the name of a user-written subprogram.

(1)

- (ii) Identify a logical operator used in this program.

(1)

- (iii) Identify the name of a variable that holds a real number.

(1)

- (iv) Identify the name of a parameter.

(1)

**(Total for Question 1 = 14 marks)**



## 2 Programs can process numerical data.

(a) Identify the method used to check input data matches set rules.

(1)

- ☐ **A** Abstraction
- ☐ **B** Encryption
- ☐ **C** Iteration
- ☐ **D** Validation

(b) **Figure 2** shows a section from an employee payslip produced by a computer program.

| Pay Rate |       | Hours  | Total Pay |         |
|----------|-------|--------|-----------|---------|
| £        | 18.75 | 19     | £         | 356.25  |
|          |       | Tax    | 20%       | £ 71.25 |
|          |       | Income | £         | 285.00  |

**Figure 2**

(i) Complete the table to give **one** data type shown in **Figure 2** and give an example value.

(2)

|                  |  |
|------------------|--|
| <b>Data type</b> |  |
| <b>Example</b>   |  |

(ii) Complete the table to give an input, a process and an output used in the program to produce the payslip.

(3)

|                |  |
|----------------|--|
| <b>Input</b>   |  |
| <b>Process</b> |  |
| <b>Output</b>  |  |

(c) Numerical values can be represented in denary, binary and hexadecimal.

A program converts denary values in the range 0 to 255 into hexadecimal.

(i) The program is tested using appropriate integer values.

Complete the table to show an example of normal, boundary and erroneous test data for this program.

(2)

|           | Test data |
|-----------|-----------|
| Normal    |           |
| Boundary  |           |
| Erroneous |           |

(ii) The program uses subprograms to do the conversion.

Open **Q02cii** in the code editor.

Amend the code to complete the program to:

- convert the user input to integer
- check the input is between 0 and 255
- call the **getHex** subprogram, with the user input as an argument.

Do not add any further functionality.

Save your code as **Q02ciiFINISHED** with the correct file extension for the programming language.

(4)

(Total for Question 2 = 12 marks)



3 A youth club organises age-specific events for its members.

A program is used to find members of the correct age.

(a) The program first sorts the list of members by age.

(i) A bubble sort algorithm is used to sort the list into ascending order.

The table shows a list of ages that must be sorted.

Complete the table to show how the algorithm sorts the list.

You may not need to use all the rows.

(2)

| 9 | 11 | 12 | 19 | 14 | 18 | 15 | 17 |
|---|----|----|----|----|----|----|----|
|   |    |    |    |    |    |    |    |
|   |    |    |    |    |    |    |    |
|   |    |    |    |    |    |    |    |
|   |    |    |    |    |    |    |    |
|   |    |    |    |    |    |    |    |

(ii) State the number of swaps made in the first pass.

(1)

(iii) State the **least** number of passes needed to complete the sort algorithm for this list.

(1)

(b) The youth club stores a list of sports equipment.

The list contains over 1000 items, sorted in ascending order.

State and justify the **most** appropriate search algorithm to find an item in the list.

(2)

Search algorithm

Justification



- (c) A program is required to calculate and display the average age of members taking part in an activity.

**Figure 3** shows a flowchart for the algorithm.

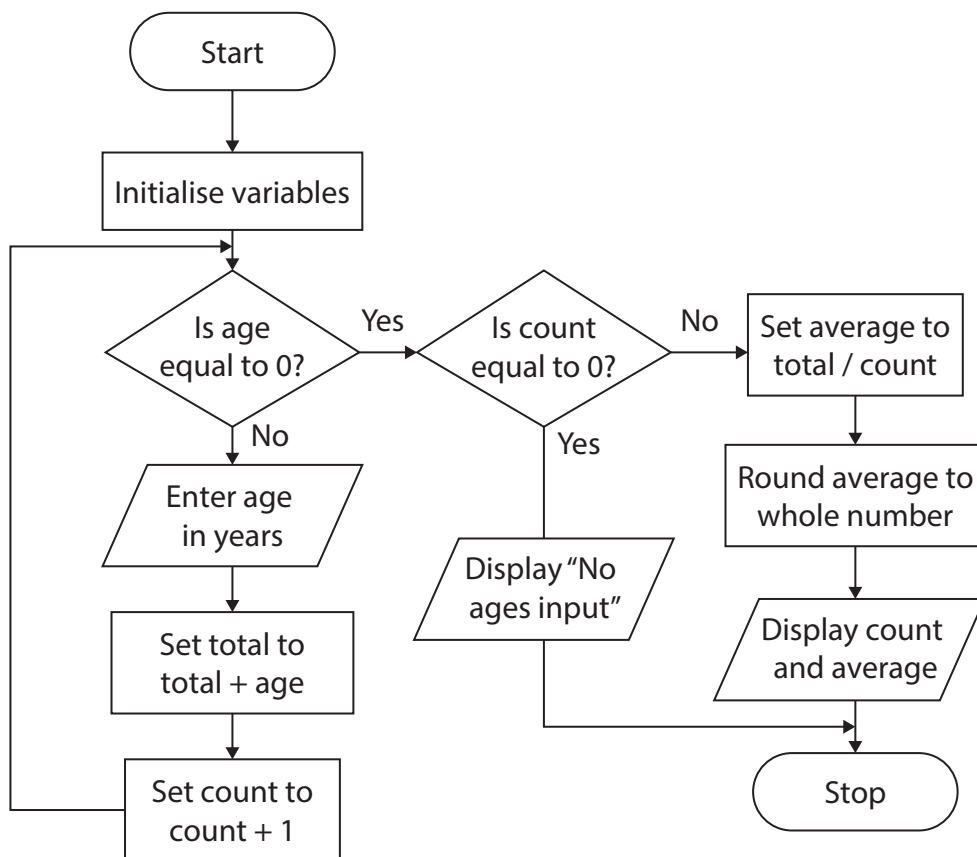


**Figure 3**

Open **Q03c** in the code editor.

Write a program to implement the logic of the flowchart.

Use the data given in **Figure 4** to test your program.

| Test data     | Expected output                            |
|---------------|--------------------------------------------|
| 7 8 9 10 11 0 | You entered 5 ages; the average age is 9.  |
| 16 16 14 0    | You entered 3 ages; the average age is 15. |
| 0             | No ages input                              |

**Figure 4**

Do not add any further functionality.

Save your code as **Q03cFINISHED** with the correct file extension for the programming language.

(5)

**(Total for Question 3 = 11 marks)**

4 Eli is designing a three-throw dice game to play on a computer.

He uses abstraction and decomposition to help him.

(a) **Figure 5** shows how Eli uses comments to plan code before writing it.

Eli's preferred language uses a double-slash // to write comments.

|    |                                                   |
|----|---------------------------------------------------|
| 1  | // roll dice 3 times                              |
| 2  |                                                   |
| 3  | // display number rolled                          |
| 4  |                                                   |
| 5  | // update subtotal                                |
| 6  |                                                   |
| 7  | // check for roll of 3 or 6 and increment booster |
| 8  |                                                   |
| 9  | // calculate and output final score               |
| 10 |                                                   |

**Figure 5**

State why this is a type of decomposition.

(1)

.....

.....

(b) Eli decides to use a library subprogram to generate random numbers that model a dice throw.

Explain why this is an example of abstraction.

(2)

.....

.....

.....

.....



(c) Eli's program has these requirements.

- A roll of the dice is a random number between 1 and 6.
- The dice is rolled three times.
- The numbers rolled are added to a subtotal.
- The final score is calculated by multiplying the subtotal by a booster value.
- The booster value starts at 1.
- Each time a 3 or 6 is rolled, the booster value is incremented by 1.
- The game outputs the value of each roll, and the final score.

**Figure 6** shows a trace table for one execution of the program.

| Output    | Subtotal                         | Booster                       |
|-----------|----------------------------------|-------------------------------|
| Rolled: 3 | $0 + 3 = 3$                      | 3 rolled, so incremented to 2 |
| Rolled: 2 | $3 + 2 = 5$                      | Neither 3 nor 6 rolled        |
| Rolled: 6 | $5 + 6 = 11$                     | 6 rolled, so incremented to 3 |
| Score 33  | Subtotal (11) * Booster (3) = 33 |                               |

**Figure 6**

Open **Q04c** in the code editor.

Write the code to create this program.

You must use the layout, variables and constants given in **Q04c**.

Do not add any further functionality.

Save your code as **Q04cFINISHED** with the correct file extension for the programming language.

(8)

**(Total for Question 4 = 11 marks)**



5 Martha is a computer science student and is learning to code.

(a) **Figure 7** shows an algorithm in pseudocode written by Martha.

|    |                                             |
|----|---------------------------------------------|
| 1  | SET first TO 0                              |
| 2  | SET second TO 0                             |
| 3  | SET answer TO 0                             |
| 4  | SET user_answer TO 0                        |
| 5  | SET score TO 0                              |
| 6  |                                             |
| 7  | REPEAT 10 TIMES                             |
| 8  | SET score TO 0                              |
| 9  | SET first TO RANDOM(99)                     |
| 10 | SET second TO RANDOM(99)                    |
| 11 | SET answer TO first + second                |
| 12 |                                             |
| 13 | SEND first "+" second "=" TO DISPLAY        |
| 14 | RECEIVE user_answer FROM (INTEGER) KEYBOARD |
| 15 |                                             |
| 16 | IF user_answer = answer THEN                |
| 17 | SET score TO score + 1                      |
| 18 | END IF                                      |
| 19 | END REPEAT                                  |
| 20 | SEND score TO DISPLAY                       |

**Figure 7**

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA



There is a logic error on line 8 of the pseudocode.

- (i) Explain the problem this error will cause.

(2)

- (ii) Describe the purpose of the algorithm shown in **Figure 7**, assuming the error has been corrected.

(2)

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA



(b) Martha is writing a program to calculate grades.

Each student takes three tests. Each test has a maximum score of 50.

Scores are stored in a two-dimensional array called `tbl_scores`.

The lowest of the three scores is ignored and the other two are added to get a total out of 100.

Grades are based on the table in **Figure 8**.

| Total      | Grade       |
|------------|-------------|
| 80 or more | Distinction |
| 65 to 79   | Merit       |
| 50 to 64   | Pass        |
| 49 or less | Fail        |

**Figure 8**

Each student's total and grade are saved to a text file.

For example, the first student scores 45, 42 and 43. The lowest score 42 is ignored. The total is 88 and the grade is Distinction.



**Figure 9** shows part of the text file.

|    |             |
|----|-------------|
| 88 | Distinction |
| 77 | Merit       |
| 96 | Distinction |
| 67 | Merit       |
| 48 | Fail        |
| 76 | Merit       |

**Figure 9**

The program must:

- calculate the total and grade for each student
- write total and grade to a new line in the file **grades.txt**
- display a message to say the grades file has been created.

Open **Q05b** in the code editor.

Write the program.

You must use the array given in **Q05b**.

Do not add any further functionality.

Save your code as **Q05bFINISHED** with the correct file extension for the programming language.

(8)

**(Total for Question 5 = 12 marks)**

- 6 Environmental scientists monitor the annual average temperatures of major cities around the world.

City names and their annual average temperatures are stored in two arrays.

- `tbl_city` stores the city names.
- `tbl_temp` stores the annual average temperature for each city in either Celsius or Fahrenheit. A temperature of 25° Celsius is stored as **C25** and a temperature of 80° Fahrenheit is stored as **F80**.

Scientists want a program that will enable them to:

- update temperature data already stored
- calculate the overall average annual temperature of all the cities
- view the data in a table as shown in **Figure 10**.

| City            | Celsius | Fahrenheit |
|-----------------|---------|------------|
| Reggane         | 39      | 103        |
| Cairo           | 28      | 82         |
| New Delhi       | 34      | 93         |
| Muscat          | 35      | 95         |
| Athens          | 29      | 84         |
| Barcelona       | 26      | 79         |
| Havana          | 29      | 84         |
| Phoenix         | 35      | 95         |
| Brisbane        | 25      | 77         |
| Darwin          | 27      | 81         |
| Overall Average | 30      | 87         |

**Figure 10**

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA



The program must:

- allow a scientist to input the name of a city and its temperature in either Celsius or Fahrenheit
  - if the city is in the array, its temperature is updated
  - if the city is not in the array, an appropriate error message is given
- display a table with:
  - a header row and dividing line
  - each city and its annual average temperature in both Celsius and Fahrenheit shown as whole numbers
  - the overall average annual temperature for all cities in both Celsius and Fahrenheit shown as whole numbers

The formula to convert from Celsius to Fahrenheit is:  $F = C \times 1.8 + 32$

The formula to convert from Fahrenheit to Celsius is:  $C = (F - 32) \div 1.8$

Open the file **Q06** in the code editor.

Write the code to produce this program with the intended output.

Include at least one subprogram.

**Your program should function correctly even if the number of cities represented in the data is changed.**

Use techniques to make your code easy to read.

Save your amended code as **Q06FINISHED** with the correct file extension for the programming language.

**(Total for Question 6 = 20 marks)**

**TOTAL FOR PAPER = 80 MARKS**



DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

**BLANK PAGE**

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

**BLANK PAGE**

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

**BLANK PAGE**



**Pearson Edexcel International GCSE (9–1)**

**Monday 2 – Wednesday 4 June 2025**

Paper  
reference

**4CP0/2AW**

**Computer Science (Python)**

**Component 2**

**Pseudocode command set**

**Resource Booklet**

**Do not return this Booklet with the question paper.**

*Turn over* ►

**P78639A**

©2025 Pearson Education Ltd.  
Y:1/1/1/1/1/1/



  
**Pearson**

## Pseudocode command set

Questions in the written examination that involve code will use this pseudocode for clarity and consistency. However, students may answer questions using any valid method.

### Data types

INTEGER

REAL

BOOLEAN

CHARACTER

### Type coercion

Type coercion is automatic if indicated by context. For example  $3 + 8.25 = 11.25$  (integer + real = real)

Mixed mode arithmetic is coerced like this:

|         | INTEGER | REAL |
|---------|---------|------|
| INTEGER | INTEGER | REAL |
| REAL    | REAL    | REAL |

Coercion can be made explicit. For example, RECEIVE age FROM (INTEGER) KEYBOARD assumes that the input from the keyboard is interpreted as an INTEGER, not a STRING.

### Constants

The value of constants can only ever be set once. They are identified by the keyword CONST. Two examples of using a constant are shown.

CONST REAL PI

SET PI TO 3.14159

SET circumference TO radius \* PI \* 2

### Data structures

ARRAY

STRING

Indices start at zero (0) for all data structures.

All data structures have an append operator, indicated by &.

Using & with a STRING and a non-STRING will coerce to STRING. For example, SEND 'Fred' & age TO DISPLAY, will display a single STRING of 'Fred18'.

**Identifiers**

Identifiers are sequences of letters, digits and '\_', starting with a letter, for example: MyValue, myValue, My\_Value, Counter2

**Functions**

LENGTH()

For data structures consisting of an array or string.

RANDOM(n)

This generates a random number from 0 to n.

**Comments**

Comments are indicated by the # symbol, followed by any text.

A comment can be on a line by itself or at the end of a line.

**Devices**

Use of KEYBOARD and DISPLAY are suitable for input and output.

Additional devices may be required, but their function will be obvious from the context. For example, CARD\_READER and MOTOR are two such devices.

**Notes**

In the following pseudocode, the < > indicates where expressions or values need to be supplied. The < > symbols are not part of the pseudocode.

## Variables and arrays

| Syntax                                       | Explanation of syntax                                          | Example                                               |
|----------------------------------------------|----------------------------------------------------------------|-------------------------------------------------------|
| SET Variable TO <value>                      | Assigns a value to a variable.                                 | SET Counter TO 0<br>SET MyString TO 'Hello world'     |
| SET Variable TO <expression>                 | Computes the value of an expression and assigns to a variable. | SET Sum TO Score + 10<br>SET Size to LENGTH(Word)     |
| SET Array[index] TO <value>                  | Assigns a value to an element of a one-dimensional array.      | SET ArrayClass[1] TO 'Ann'<br>SET ArrayMarks[3] TO 56 |
| SET Array TO [<value>, ...]                  | Initialises a one-dimensional array with a set of values.      | SET ArrayValues TO [1, 2, 3, 4, 5]                    |
| SET Array [RowIndex, ColumnIndex] TO <value> | Assigns a value to an element of a two dimensional array.      | SET ArrayClassMarks[2,4] TO 92                        |

## Selection

| Syntax                                                           | Explanation of syntax                                                                             | Example                                                                                                    |
|------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| IF <expression> THEN<br><command><br>END IF                      | If <expression> is true then command is executed.                                                 | IF Answer = 10 THEN<br>SET Score TO Score + 1<br>END IF                                                    |
| IF <expression> THEN<br><command><br>ELSE<br><command><br>END IF | If <expression> is true then first <command> is executed, otherwise second <command> is executed. | IF Answer = 'correct' THEN<br>SEND 'Well done' TO DISPLAY<br>ELSE<br>SEND 'Try again' TO DISPLAY<br>END IF |

| Repetition                                                                                    |                                                                                                           |                                                                                                                                                                |
|-----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Syntax                                                                                        | Explanation of syntax                                                                                     | Example                                                                                                                                                        |
| WHILE <condition> DO<br><command><br>END WHILE                                                | Pre-conditioned loop. Executes <command> whilst <condition> is true.                                      | WHILE Flag = 0 DO<br>SEND 'All well' TO DISPLAY<br>END WHILE                                                                                                   |
| REPEAT<br><command><br>UNTIL <expression>                                                     | Post-conditioned loop. Executes <command> until <condition> is true. The loop must execute at least once. | REPEAT<br>SET Go TO Go + 1<br>UNTIL Go = 10                                                                                                                    |
| REPEAT <expression> TIMES<br><command><br>END REPEAT                                          | Count controlled loop. The number of times <command> is executed is determined by the expression.         | REPEAT 100-Number TIMES<br>SEND '*' TO DISPLAY<br>END REPEAT                                                                                                   |
| FOR <id> FROM <expression> TO<br><expression> DO<br><command><br>END FOR                      | Count controlled loop. Executes <command> a fixed number of times.                                        | FOR Index FROM 1 TO 10 DO<br>SEND ArrayNumbers[Index] TO DISPLAY<br>END FOR                                                                                    |
| FOR <id> FROM <expression> TO<br><expression> STEP<br><expression> DO<br><command><br>END FOR | Count controlled loop using a step.                                                                       | FOR Index FROM 1 TO 500 STEP 25 DO<br>SEND Index TO DISPLAY<br>END FOR                                                                                         |
| FOR EACH <id> FROM<br><expression> DO<br><command><br>END FOREACH                             | Count controlled loop. Executes for each element of an array.                                             | SET WordsArray TO ['The', 'Sky', 'is', 'grey']<br>SET Sentence to "<br>FOR EACH Word FROM WordsUArray DO<br>SET Sentence TO Sentence & Word & "<br>END FOREACH |

## Input/output

| Syntax                                          | Explanation of syntax          | Example                                                                                                                                 |
|-------------------------------------------------|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| SEND <expression> TO DISPLAY                    | Sends output to the screen.    | SEND 'Have a good day.' TO DISPLAY                                                                                                      |
| RECEIVE <identifier> FROM<br>(type)<br><device> | Reads input of specified type. | RECEIVE Name FROM (STRING) KEYBOARD<br>RECEIVE LengthOfJourney FROM (INTEGER) CARD_READER<br>RECEIVE YesNo FROM (CHARACTER) CARD_READER |

## File handling

| Syntax                | Explanation of syntax                                                                                          | Example                                     |
|-----------------------|----------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| READ <File> <record>  | Reads in a record from a <file> and assigns to a <variable>. Each READ statement reads a record from the file. | READ MyFile.doc Record                      |
| WRITE <File> <record> | Writes a record to a file. Each WRITE statement writes a record to the file.                                   | WRITE MyFile.doc Answer1, Answer2, 'xyz 01' |

## Subprograms

| Syntax                                                                                                    | Explanation of syntax            | Example                                                                                                                             |
|-----------------------------------------------------------------------------------------------------------|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| PROCEDURE <id><br>(<parameter>, ...)<br>BEGIN PROCEDURE<br><command><br>END PROCEDURE                     | Defines a procedure.             | PROCEDURE CalculateAverage (Mark1, Mark2, Mark3)<br>BEGIN PROCEDURE<br>SET Avg to (Mark1 + Mark2 + Mark3)/3<br>END PROCEDURE        |
| FUNCTION <id><br>(<parameter>, ...)<br>BEGIN FUNCTION<br><command><br>RETURN <expression><br>END FUNCTION | Defines a function.              | FUNCTION AddMarks (Mark1, Mark2, Mark3)<br>BEGIN FUNCTION<br>SET Total to (Mark1 + Mark2 + Mark3)/3<br>RETURN Total<br>END FUNCTION |
| <id> (<parameter>, ...)                                                                                   | Calls a procedure or a function. | Add (FirstMark, SecondMark)                                                                                                         |

| Arithmetic operators |                  |
|----------------------|------------------|
| Symbol               | Description      |
| +                    | Add              |
| -                    | Subtract         |
| /                    | Divide           |
| *                    | Multiply         |
| ^                    | Exponent         |
| MOD                  | Modulo           |
| DIV                  | Integer division |

| Relational operators |                          |
|----------------------|--------------------------|
| Symbol               | Description              |
| =                    | equal to                 |
| <>                   | not equal to             |
| >                    | greater than             |
| >=                   | greater than or equal to |
| <                    | less than                |
| <=                   | less than or equal to    |

| Logical operators |                                                                                 |
|-------------------|---------------------------------------------------------------------------------|
| Symbol            | Description                                                                     |
| AND               | Returns true if both conditions are true.                                       |
| OR                | Returns true if any of the conditions are true.                                 |
| NOT               | Reverses the outcome of the expression; true becomes false, false becomes true. |

**BLANK PAGE**

