

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 1 June – Wednesday 3 June 2026**

Time 3 hours

Paper reference **4CP0/2B**

**Computer Science (C#)**

**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 C# 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 C#.
- 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 ►

P83971A

©2026 Pearson Education Ltd.  
P: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 C# throughout the examination.

1 Programmers write code to solve problems.

(a) A program asks the user to enter a decimal value.

(i) Identify the **most appropriate** data type for storing a decimal value.

(1)

- ☐ **A** Boolean
- ☐ **B** Integer
- ☐ **C** Real
- ☐ **D** String

(ii) Input data values must be between 1.5 and 2.5

Complete the table to show the type of test for **each** value.

The first row has been completed for you.

(2)

| Type of test | Value |
|--------------|-------|
| Boundary     | 1.5   |
|              | 2.0   |
|              | 0.5   |



DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

(b) A program calculates body mass index (BMI).

Open **Q01b** in the code editor.

Use the code to answer these questions.

(i) Give the line number where a logical operator is used.

(1)

(ii) Give the name of an integer variable.

(1)

(iii) Give the text of a comment.

(1)

(iv) Give the line number where iteration begins.

(1)

(v) Give the line number where arithmetic operators are used.

(1)



(c) A program helps athletes monitor their weekly exercise.

Athletes enter their level and the total minutes of exercise for each day.

Elite athletes should exercise for at least 300 minutes per week and club athletes for at least 200 minutes.

The program calculates total minutes of exercise per week and outputs an appropriate message.

**Figure 1** shows the output from a working program.

```
Please enter e (elite) or c (club) >e
Please enter total minutes exercise on Monday >50
Please enter total minutes exercise on Tuesday >30
Please enter total minutes exercise on Wednesday >0
Please enter total minutes exercise on Thursday >60
Please enter total minutes exercise on Friday >0
Please enter total minutes exercise on Saturday >60
Please enter total minutes exercise on Sunday >25
Weekly exercise: 225 minutes
You have not met your exercise target of 300 minutes.
```

**Figure 1**

Open **Q01c** in the code editor.

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

Amend the code to correct the errors.

Save your code as **Q01cFINISHED** with the correct file extension for the programming language.

(4)

**(Total for Question 1 = 12 marks)**



DO NOT WRITE IN THIS AREA

BLANK PAGE



## 2 Programs can process data input by users.

- (a) **Figure 2** shows the pseudocode for an algorithm used to process numbers entered by the user.

|    |                                        |
|----|----------------------------------------|
| 1  | RECEIVE number FROM (INTEGER) KEYBOARD |
| 2  |                                        |
| 3  | WHILE number > 0 DO                    |
| 4  | IF number >= 90 THEN                   |
| 5  | SET outMessage TO 'Too Big'            |
| 6  | ELSE                                   |
| 7  | IF number >= 30 THEN                   |
| 8  | SET outMessage TO 'Just Right'         |
| 9  | ELSE                                   |
| 10 | IF number MOD 2 = 0 THEN               |
| 11 | SET outMessage TO 'Low Evens'          |
| 12 | ELSE                                   |
| 13 | SET outMessage TO 'Low Odds'           |
| 14 | END IF                                 |
| 15 | END IF                                 |
| 16 | END IF                                 |
| 17 | SEND outMessage TO DISPLAY             |
| 18 | RECEIVE number FROM (INTEGER) KEYBOARD |
| 19 | END WHILE                              |
| 20 |                                        |
| 21 | SET outMessage TO 'End'                |
| 22 | SEND outMessage TO DISPLAY             |

**Figure 2**

Complete the table to show the output for **each** of the given inputs.

(3)

| Input | Output |
|-------|--------|
| 80    |        |
| -2    |        |
| 16    |        |



(b) Identify the programming construct used on **line 4** of the pseudocode.

(1)

- ☐ **A** Encryption
- ☐ **B** Function
- ☐ **C** Repetition
- ☐ **D** Selection

(c) The algorithm is coded in a high-level language.

Explain what happens when a user enters **TEN** instead of **10**

(2)

---

---

---

---

---

(d) Describe the purpose of the modulus operator.

(2)

---

---

---

---

---



(e) A program calculates the volume of a cylinder, as shown in **Figure 3**.

The user enters the radius and height.

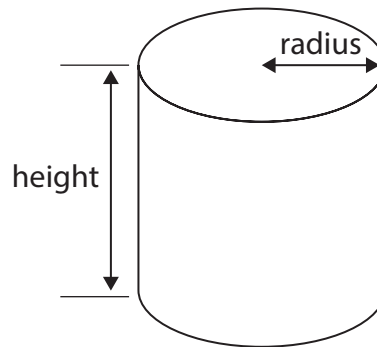


**Figure 3**

The formula to calculate the volume of a cylinder is:

$$\text{volume} = \pi \times \text{radius}^2 \times \text{height}$$

**Figure 4** shows the output from a working program.

```
Enter cylinder radius in cm: 12
Enter cylinder height in cm: 100
Cylinder volume: 45239cm3
```

**Figure 4**

Open **Q02e** in the code editor.

Amend the code to produce the intended output.

You must use the variables given in the code file.

Do not add any further functionality.

Save your code as **Q02eFINISHED** with the correct file extension for the programming language.

(4)

**(Total for Question 2 = 12 marks)**



DO NOT WRITE IN THIS AREA

BLANK PAGE



### 3 Algorithms are written to solve problems.

#### (a) A bank pays 4.5% interest on savings.

The bank wants a program to show customers the predicted value of their savings each year.

Customers enter how much money they will deposit and the period (in years) they will save for.

The program calculates and displays:

- the value of the savings account at the end of each year
- the total amount of money earned at the end of the period.

**Figure 5** shows an algorithm for the program.

|    |                                                                        |
|----|------------------------------------------------------------------------|
| 11 | RECEIVE deposit FROM (REAL) KEYBOARD                                   |
| 12 | RECEIVE period FROM (INTEGER) KEYBOARD                                 |
| 13 |                                                                        |
| 14 | SET value TO deposit                                                   |
| 15 |                                                                        |
| 16 | WHILE period > 0 DO                                                    |
| 17 | SET value TO value + value * INTEREST                                  |
| 18 | SET year TO year + 1                                                   |
| 19 | SET period TO period - 1                                               |
| 20 | SEND ("Year", year, "Savings value: £",<br>(INTEGER) value) TO DISPLAY |
| 21 | END WHILE                                                              |
| 22 |                                                                        |
| 23 | SET earned TO value - deposit                                          |
| 24 | SEND ("Total earned: £", (INTEGER) earned)<br>TO DISPLAY               |

**Figure 5**

**Figure 6** shows test data for a fully functional program.

| Deposit | Period | Output                                                                                                                                                                         |
|---------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| £ 500   | 5      | Year 1 Savings value: £ 522<br>Year 2 Savings value: £ 546<br>Year 3 Savings value: £ 570<br>Year 4 Savings value: £ 596<br>Year 5 Savings value: £ 623<br>Total earned: £ 123 |

**Figure 6**



Open **Q03a** in the code editor.

Write the code to implement the algorithm given in **Figure 5**.

You must use the variables given in the code file.

Do not add any further functionality.

Save your code as **Q03aFINISHED** with the correct file extension for the programming language.

(7)

(b) Programs use constants and variables to store data.

(i) Give **two** characteristics of a constant.

(2)

1 .....

2 .....

(ii) Describe **one** difference between local and global variables.

(2)

(Total for Question 3 = 11 marks)

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA



DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

BLANK PAGE



4 Programs can store values that have different data types.

(a) Identify the structure that can store multiple values with different data types.

(1)

- ☐ **A** Library
- ☐ **B** Parameter
- ☐ **C** Record
- ☐ **D** String

(b) A program logs hours of sunshine each day for a month.

State the **most appropriate** data structure for storing the sunshine data.

Justify your answer.

(2)

Data structure

Justification

(c) Programmers should write code that is easy to read.

Explain **one** way that subprograms improve the readability of code.

(2)



(d) A program is needed to add numbers to the Fibonacci sequence.

The Fibonacci sequence starts with 0 followed by 1

Each new number is the sum of the previous two numbers, e.g.  $0 + 1 = 1$ ,  $1 + 1 = 2$ ,  $1 + 2 = 3$ ,  $2 + 3 = 5$  and so on.

**Figure 7** shows part of the Fibonacci sequence.

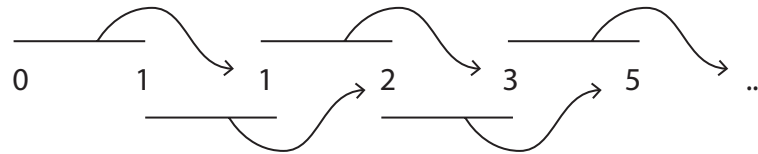


**Figure 7**

The program must:

- display the last two numbers in the given Fibonacci sequence
- calculate the next number in the sequence
- ask the user to input the next number
- if the input is correct:
  - display a suitable message
  - update the last two numbers
  - update the number of correct user inputs
- if the input is incorrect:
  - display a suitable message with the number of correct user inputs
  - finish the program.



**Figure 8** shows part of the output from a working program.

```
Last two numbers in the sequence are 0 and 1
Enter the next number in the sequence: 1
Correct

Last two numbers in the sequence are 1 and 1
Enter the next number in the sequence: 2
Correct

Last two numbers in the sequence are 1 and 2
Enter the next number in the sequence: 4
Incorrect. Numbers you added to sequence: 2
```

**Figure 8**

Open **Q04d** in the code editor.

Write the program.

You must use the variables given in the code file.

Do not add any further functionality.

Save your code as **Q04dFINISHED** with the correct file extension for the programming language.

(6)

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



5 Programs can sort and search sets of data.

(a) **Figure 9** shows a list of words.

|        |        |        |       |        |         |         |
|--------|--------|--------|-------|--------|---------|---------|
| Autumn | Sunset | Forest | Ocean | Marble | Jasmine | Crystal |
|--------|--------|--------|-------|--------|---------|---------|

**Figure 9**

State the maximum number of comparisons necessary for a linear search algorithm to find a target word in this list.

(1)

(b) Complete the table to show the order in which numbers will be compared, using a binary search algorithm, to find **5017**

Write the number 1 by the first number compared, 2 by the second to be compared, and so on.

(4)

| Number | Order compared |
|--------|----------------|
| 2328   |                |
| 3369   |                |
| 3928   |                |
| 4091   |                |
| 5017   |                |
| 5909   |                |
| 6266   |                |
| 6499   |                |
| 7148   |                |
| 7789   |                |
| 8027   |                |
| 8412   |                |
| 9027   |                |
| 9031   |                |
| 9256   |                |



DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

(c) A robot can move in four directions; north (N), south (S), east (E) and west (W).

A string holds the instructions for controlling the robot's movement.

A program is needed to compress the instruction string.

The compressed string will store direction characters, followed by the number of consecutive occurrences.

For example, the instruction string **NNNENN** will be compressed to **N3E1N2**

The program will write the compressed string to a file called **instructions.txt**

The message **Instruction file has been created** will be displayed.

**Figure 10** shows the **instructions.txt** file from a working program.

**N3W4S5E3N1W9S4W5S4W1N7E3**

**Figure 10**

Open **Q05c** in the code editor.

Write the program.

You must use the variables and structure given in the code file.

Do not add any further functionality.

Save your code as **Q05cFINISHED** with the correct file extension for the programming language.

(9)

**(Total for Question 5 = 14 marks)**

**6** A card game, based on computer specifications, is being programmed.

Each card has three categories: Storage, Cores and RAM.

**Figure 11** shows an example of the cards.

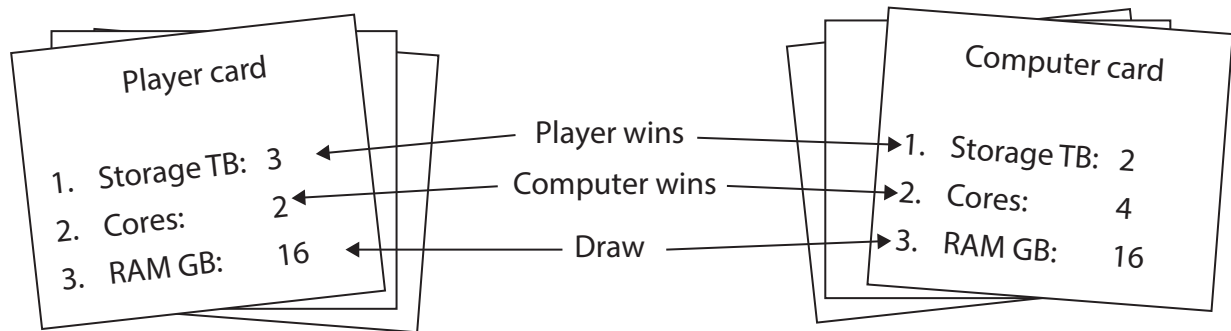


**Figure 11**

The user gets half the pack and the computer gets the other half.

All cards in the pack must be played.

To play, the player is shown the next card in their hand and chooses a category.

The chosen category on the player card is compared to the value on the computer's card. The highest value wins 1 point. There are no points for a draw.

The program outputs the final scores when all cards have been played.

**Figure 12** shows the output from the last round in a functioning program.

```

Your card:
Storage TB: 4
Cores: 6
RAM GB: 16
Please enter 1 for Storage, 2 for Cores or 3 for RAM
>3

Computer card:
Storage TB: 3
Cores: 4
RAM GB: 8

You win!

Final score:
Player 3 - 1 Computer
  
```

**Figure 12**

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA



The program code file contains a two-dimensional array representing a pack of cards.

The categories in the pack array are in the order: Storage, Cores and RAM.

The program has these requirements:

- Dealing:
  - half the cards are dealt to the user, the other half to the computer
  - each card is used once only
- Playing:
  - the player's next card is displayed
  - the player inputs their choice of category
  - the program must validate player input to only accept values 1, 2 or 3
  - the computer's next card is then displayed
  - the program compares the selected category to decide the winner of each round
  - 1 point is added to the winner's score. No points are awarded for a draw
- Outputs:
  - the result of each round is displayed (win, lose or draw)
  - the final scores are displayed when all cards have been played.

Open **Q06** in the code editor.

Write the program.

The program should work for any size of pack. Packs have an even number of cards.

You should use the data structure given in the code file.

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



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

**Monday 1 June – Wednesday 3 June 2026**

Paper  
reference

**4CP0/2B**

**Computer Science (C#)**

**Component 2**

**Pseudocode command set**

**Resource Booklet**

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

*Turn over* ►

**P83971A**

©2026 Pearson Education Ltd.  
P:1/1/1/



P 8 3 9 7 1 A

  
**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 WordsArray 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**

