Past-paper walk-through

Programming concepts ▪ 8.1

eXercise

Questions in this set

- 0478/21 N25 Q3 ▪ 1 mark
- 0478/22 N25 Q1 ▪ 1 mark
- 0478/23 N25 Q3 ▪ 1 mark
- 0478/21 J25 Q5 ▪ 4 marks
- 0478/22 N24 Q2 ▪ 1 mark
- 0478/21 J24 Q3 ▪ 4 marks
- 0478/22 J24 Q5 ▪ 6 marks
- 0478/23 J24 Q3 ▪ 4 marks
- 0478/23 J24 Q7 ▪ 7 marks
- 0478/22 M24 Q6 ▪ 4 marks

Questions in this set

- 0478/22 N23 Q5 ▪ 3 marks
- 0478/22 N23 Q8 ▪ 4 marks

Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

Question 3

0478/21 N25 ▪ Q3 ▪ 1 mark

Tick (✓) one box to identify the operator that returns the quotient of a calculation with the fractional part discarded. **[1]**

A /

B \texttt{DIV}

C ^{}

D \texttt{MOD}

Solution 3

A /

B \texttt{DIV} ✓

C ^{}

D \texttt{MOD}

Answer: **B**

Question 1

0478/22 N25 ▪ Q1 ▪ 1 mark

Tick (✓) **one** box to complete this sentence.

The result of the arithmetic operation $4 \wedge 2$ is **[1]**

- **A** 2
- **B** 8
- **C** 16
- **D** 42

Solution 1

A 2

B 8

C 16 ✅

D 42

Answer: **C**

Question 3

0478/23 N25 ▪ Q3 ▪ 1 mark

Tick (✓) **one** box to identify the most appropriate data type to store `'M'` **[1]**

- **A** integer
- **B** char
- **C** real
- **D** Boolean

Solution 3

A integer

B char ✓

C real

D Boolean

Answer: **B**

Question 5

0478/21 J25 ▪ Q5 ▪ 4 marks

Two library routines used in programming are `MOD` and `DIV`.
State the purpose of each of the library routines. Give **one** example pseudocode statement for each of the library routines. **[4]**

Solution 5

Worked steps

Both split a division into its two halves, and the marks are for **the purpose and an example** of each.

- `MOD` returns the **remainder** of a division. `X <- MOD(10, 3)` gives **1**.
- `DIV` returns the **quotient** — integer division, discarding the remainder. `X <- DIV(10, 3)` gives **3**.

Give an example with actual numbers. "MOD gives the remainder" is one mark; the worked example beside it is the other, and it also proves you have them the right way round.

Solution 5

Together they reconstruct the original: $10 = 3 \times 3 + 1$, that is DIV $\times$ divisor $+$ MOD. That identity is the check to apply if you ever doubt which is which.

Solution 5

The two uses that come up constantly and are worth knowing: `MOD 2` tests whether a number is odd or even, and `DIV` converts a total into whole units — seconds into minutes, pence into pounds — with `MOD` giving what is left over.

Mark scheme

- One mark for each bullet point
- ▪ MOD – returns the remainder of a division calculation.
- ▪ Example: X ← MOD(10, 3)
- ▪ DIV – Integer division/returns the quotient of a division.
- ▪ Example: Y ← DIV(10, 3)

Question 2

0478/22 N24 ▪ Q2 ▪ 1 mark

Tick (✓) **one** box to complete this sentence.

A pseudocode example of a selection statement is **[1]**

A \texttt{CALL Sorting(Value1, Value2)}

B \texttt{DECLARE Count : INTEGER}

C \texttt{IF X = 7 THEN Y <- 21 ENDIF}

D \texttt{WHILE X <> -1 DO}

Solution 2

A \texttt{CALL Sorting(Value1, Value2)}

B \texttt{DECLARE Count : INTEGER}

C \texttt{IF X = 7 THEN Y <- 21 ENDIF} ✓

D \texttt{WHILE X <> -1 DO}

Answer: **C**

Question 3

0478/21 J24 ▪ Q3 ▪ 4 marks

State what is meant by the data types integer and real. Give an example of each. **[4]**

Solution 3

- 3
- One mark for a correct statement about each data type ...
- ... one mark for a correct example of data for each data type.
- Example answer
- Integer A whole number [1]
- Example 27 [1]
- Real A number that contains a fractional part [1] ...
- Example 18.75 [1]

Question 5

0478/22 J24 ▪ Q5 ▪ 6 marks

A high-level programming language makes use of arithmetic, Boolean and logical operators.

State how each type of operator is used. Give an example statement, in pseudocode, for each one. **[6]**

Solution 5

Worked steps

Six marks: **one for each description and one for each example**, so every operator type needs both.

- **Arithmetic** — used in **calculations**. Example: `A <- B + C`. The operators are `+ - * / ^`, and in pseudocode also `MOD` and `DIV`.
- **Boolean** — used for operations that give **TRUE or FALSE**. Example: `IF Found = TRUE THEN`. The operators are `AND`, `OR`, `NOT`.
- **Logical** (comparison) — used to **compare two values**. Example: `IF Count > 10 THEN`. The operators are `> < >= <= = <>`.

Solution 5

Write each example as a **statement**, not as a bare operator. The mark is for showing the operator in use, and + on its own does not.

Solution 5

The distinction candidates blur is **Boolean** against **logical**. A logical operator *produces* TRUE or FALSE by comparing two values; a Boolean operator *combines* values that are already TRUE or FALSE. They chain naturally — `IF Count > 10 AND Found = TRUE THEN` uses a logical operator twice and a Boolean operator once — and quoting a line like that shows both at work.

Mark scheme

- One mark for each description, one mark for each example
- ▪ arithmetic – used in calculations (1) A <- B + C [1]
- ▪
- Boolean – used for operations with true or false values (1) IF B AND C [1]
- ▪ logical – used in comparisons/conditional statements/selection statements (1) IF B > C [1]

Question 3

0478/23 J24 ▪ Q3 ▪ 4 marks

Describe the characteristics of the string and char data types and give an example of each. **[4]**

Solution 3

Mark scheme

- One mark for a correct statement about each data type and one mark for a correct example of data for each data type.
- For example:
- String A group of characters consisting of letters, numbers and special characters [1], Cambridge2024 [1]
- Char A single character [1] X [1]

Question 7

0478/23 J24 ▪ Q7 ▪ 7 marks

The string operation `SUBSTRING(FullText, X, Y)` returns a string from `FullText` beginning at position `X` that is `Y` characters long. The first character in `FullText` is in position 1.

(a) Write the pseudocode statements to:

- store the string "IGCSE Computer Science at Cambridge" in `FullText`
- extract and display the words "Computer Science" from the string and store it in a suitable variable
- output the original string in upper case.

[4]

(b) Write the pseudocode statements to:

Question 7

- store the content of the variable you created in part **(a)** to a text file named `"Subjects.txt"`
- close the text file at the end of the algorithm.

[3]

Solution 7

(a) Worked steps

Two statements, and the second uses a string-handling function on the first.

```
FullText <- "IGCSE Computer Science at Cambridge"
PartText <- LEFT(FullText, 5)
```

Solution 7

The marks are per mark point: **assigning the given string** to the named variable, and **extracting the required part** into the second one.
Take the variable names **exactly from the question**; a correct statement using a name of your own does not match the mark point.
Cambridge's substring functions, and the trap in each:

- `LEFT(s, n)` — the first `n` characters.
- `RIGHT(s, n)` — the last `n`.
- `MID(s, start, length)` — `length` characters starting at position `start`, and **positions count from 1**, not 0.
- `LENGTH(s)` — the number of characters, which is what to use when the extraction runs to the end of a string of unknown size.

Solution 7

Mark scheme

- One mark per mark point
- MP1
- Assignment of given string to FullText
- MP2
- Correct use of SUBSTRING and assignment of reduced string to own variable
- MP3
- Correct use of UCASE
- MP4
- Output of both strings
- Example:
- FullText <- "IGCSE Computer Science at Cambridge"
- PartText <- SUBSTRING(FullText, 7, 16)
- OUTPUT PartText, UCASE(FullText)

Solution 7

(b) Worked steps

Three statements: open, write, close.

```
OPENFILE "MyFile.txt" FOR WRITE
WRITEFILE "MyFile.txt", PartText
CLOSEFILE "MyFile.txt"
```

Solution 7

One mark per point: **opening the correct file for writing**, **writing the variable's contents**, and **closing the file**.
Three details decide them:

- `FOR WRITE`, not `FOR READ` — and note that opening for write **empties** the file first. Use `FOR APPEND` when existing contents must survive.
- **The file name appears in every statement**, in quotation marks. Pseudocode has no notion of a "current" file.
- **Close it.** The close is a mark of its own, and in a real language data can sit unwritten in a buffer until the file is closed.

Solution 7

Write the **variable**, not the literal string: `WRITEFILE "MyFile.txt", "IGCSE"` puts the right characters in the file for the wrong reason and misses the mark for using what part (a) stored. **Mark scheme**

- One mark per mark point
- MP1
- Opening the correct text file for writing
- MP2
- Writing the variable from part (a) to the file
- MP3
- Closing the text file after writing
- Example:
- OPENFILE "Subjects.txt" FOR WRITE
- WRITEFILE "Subjects.txt", PartText

Question 6

0478/22 M24 ▪ Q6 ▪ 4 marks

Describe **two** types of iteration that a programmer can use whilst writing a program.**[4]**

Solution 6

Mark scheme

One mark for identifying a type of iteration, one mark for accompanying description max four

- count controlled number of iterations is pre-determined
- pre-condition checks condition at start of loop // loop may not iterate
- post-condition checks condition at end of loop // loop always iterates at least once

Question 5

0478/22 N23 ▪ Q5 ▪ 3 marks

5 Explain how variables and constants should be used when creating and running a program [3] **[3]**

Solution 5

- 5
- One mark per mark point, max three
- MP1
- variables and constants should have meaningful identifiers
- MP2
- ...so that programmers/future programmers are able to understand
- their purpose
- MP3

Solution 5

- they are both used for data storage
- MP4
- constants store values that never change during the execution of a
- program // by example
- MP5
- variables contain values that have been calculated within the
- program / can change during the execution of the program // by
- example
- 3

Solution 5

- 0478/22
- October/November
- 2023

Question 8

0478/22 N23 ▪ Q8 ▪ 4 marks

8 Explain why a programmer would use procedures and parameters when writing a program [4] **[4]**

Solution 8

- One mark per mark point, max four
- Procedures, max three
- MP1
- to enable the programmer to write a collection of programming
- statements under a single identifier
- MP2
- to allow modular programs to be created // to allow procedures to be
- re-used within the program or in other programs

Solution 8

- MP3
- to make program creation faster because procedures can be re-used
- // to enable different programmers to work on different procedures in
- the same project
- MP4
- to make programs shorter (than using the repeated code) / using less
- duplication of code // to make programs easier to maintain due to
- being shorter.
- Parameters, max three

Solution 8

- MP5
- to pass values from the main program to a procedure / function
- MP6
- ...so that they can be used in the procedure / function
- MP7
- allow the procedure / function to be re-used with different data.
- 4
- 0478/22
- October/November

Solution 8

- 2023