Qualification Accredited

Oxford Cambridge and RSA

GCSE (9-1)

Examiners' report

COMPUTER SCIENCE

J277

For first teaching in 2020

J277/02 Summer 2023 series

Version 1

Contents

Introduction

Our examiners' reports are produced to offer constructive feedback on candidates' performance in the examinations. They provide useful guidance for future candidates.

The reports will include a general commentary on candidates' performance, identify technical aspects examined in the questions and highlight good performance and where performance could be improved. A selection of candidate answers is also provided. The reports will also explain aspects which caused difficulty and why the difficulties arose, whether through a lack of knowledge, poor examination technique, or any other identifiable and explainable reason.

Where overall performance on a question/question part was considered good, with no particular areas to highlight, these questions have not been included in the report.

A full copy of the question paper and the mark scheme can be downloaded from OCR.

Paper 2 series overview

This is the second examination series for the J277 specification.

J277/02 (Computational thinking, algorithms and programming) is one of two examined components for GCSE Computer Science. This component focuses on:

- algorithms
- programming fundamentals
- producing robust programs
- Boolean logic
- programming languages and Integrated Development Environments.

In Section A, candidates need to be comfortable with writing, completing, and using algorithms using pseudocode and/or flowcharts. This may involve applying their knowledge to unfamiliar contexts. In Section B, candidates are asked to provide answers using either OCR Exam Reference Language or a high-level language that they have studied.

It was pleasing to see candidates apply practical programming knowledge covered in the classroom in Section B. Many candidates provided succinct and creative responses to the questions.

It is a requirement that all centres offer candidates the opportunity to undertake a programming task or tasks during their course of study. These tasks should allow candidates to design, write, test, and refine programs in a high-level programming language. It was clear that candidates who had been given this opportunity answered questions on this paper more confidently.

Use of functions/procedures, text file access and 2D arrays are all covered in the J277 specification. Repeated experience of these in a practical setting helped candidates to confidently answer questions in this paper.

Centres are encouraged to be aware of the contents of the specification. Section 3c shows details of OCR Exam Reference Language (ERL). Candidates **do not** have to use this in their answers but should be aware of the ERL conventions. This will allow candidates to successfully understand and access examination questions.

Candidates who did well on this paper generally:	Candidates who did less well on this paper generally:
• were able to decompose a given problem and attempt each section, meaning that marks were able to be given for elements of each question even if the overall solution was not completely correct • understood how to create and use subroutines, including procedures and functions • demonstrated understanding of the steps needed to write data to a text file • understood how to access data from a 2D array • were able to describe the importance of IDEs, validation and testing when programming.	• missed out key steps from algorithms • repeated the question instead of breaking down how the problem could be solved algorithmically • ignored or missed key sections of answers, such as the NOT gate in the logic system or the requirement for doors in the SQL response • provided generic responses when discussing sorting algorithms instead of focusing scientifically on the steps taken • were confused about the concepts of input validation and testing, sometimes mixing these two up.

Section A overview

Section A consists of multiple questions and scenarios.

Candidates are free to write algorithms in any suitable way. This includes using:

- flowcharts
- structured English
- pseudocode
- a high-level language.

Most candidates who scored highly tended to use a high-level language consistently.

Section A begins with some simple, low challenge questions to help candidates ease into the examination; this is deliberate by design for the J277 specification.

Question 1 (a)

1 **(a)** The table contains four statements about programming languages.

Tick (✓) **one** box in each row to identify whether each statement describes a low-level programming language or a high-level programming language.

Statement	Low-level	High-level
The same language can be used on computers that use different hardware		
It allows the user to directly manipulate memory		
It allows the user to write English-like words		
It always needs to be translated into object code or machine code		

[4]

This question was answered well by many candidates. The strongest responses showed a good understanding of the difference between low-level and high-level languages. Incorrect responses tended to be on the first row related to portability. A high-level language such as Python is portable, with translators available for many different types of processors. A low-level language is specific to one type of processor.

Question 1 (b)

(b) The variables `num1` and `num2` store integers.

Write pseudocode to add the integers stored in `num1` and `num2`. Store the result in a variable with the identifier `total`

....

.... **[1]**

Candidates appear to be getting more confident at answering simple programming/pseudocode questions such as this. The majority of responses included code written to produce the required outcome. The use of multiple steps was allowed.

Question 1 (d)

(d) Read the following pseudocode algorithm:

```
01  start = 3
02  do
03    print(start)
04    start = start - 1
05  until start == -1
06  print("Finished")
```

Complete the following trace table for the given algorithm.

Line number	`start`	**Output**

[3]

This question assessed candidates' ability to trace through and understand the steps taken by an algorithm. This also tested their understanding of condition-controlled loops. Many responses were very successful with this and achieved full marks.

Mistakes tended to be with identifying the line number where each change occurred or outputting values that were not actually output (e.g. -1).

Examiners were instructed to only penalise a misunderstanding once. Where (for example) line numbers were incorrect, this would still have allowed 2 out of 3 marks to be achieved.

Question 2 (a)

2 This pseudocode algorithm totals all the numbers in the 0-indexed array `scores`

```
total = 0
for scoreCount = 1 to scores.length – 1
    scores[scoreCount] = total + total
next scoreCount
print(total)
```

The function `length` returns the number of elements in the array.

The algorithm contains several errors.

Two types of errors in a program are syntax and logic errors.

(a) State what is meant by a syntax error and a logic error.

Syntax error ....

....

Logic error ....

....

[2]

This question was answered extremely well by most candidates. These candidates correctly defined the terms given.

Candidates must understand that an example is not the same as a definition. For example, a misspelling of a command such as `print` would of course be a syntax error but this is not a definition; many other issues would cause a syntax error.

Other incorrect responses included generic responses that could apply to either term, for example: "a mistake in the program" or "where the computer doesn't understand the code".

Question 2 (b)

(b) Identify **two** logic errors in the pseudocode algorithm.

Write the refined line to correct each error.

Error 1 line number ....

Corrected line ....

....

....

Error 2 line number ....

Corrected line ....

....

....
[4]

This proved to be a relatively challenging question for many candidates. This question relied on an understanding of the term "logic error".

Two errors were present:

- line 02 (where the count-controlled loop ignored element 0 in the array)
- line 03 (where the total was incorrectly calculated and stored).

Many responses identified at least one of the line numbers containing the error. Far fewer were able to successfully fix the errors satisfactorily.

Examiners were instructed to be generous in interpreting potential fixes. The errors in either case could have been fixed using OCR Exam Reference Language (as presented) or using any other sensible form. Responses using programming syntax were credited, as were those who simply used English, e.g. "change the 1 to a 0 on line 02".

Instructions were also included in the mark scheme to allow full marks for candidates who fixed the errors in one step on line 03.

Question 3 (a)

3 An insertion sort is one type of sorting algorithm.

A student has written a pseudocode algorithm to perform an insertion sort on a 1D array `names`.

```
names = ["Kareem", "Sarah", "Zac", "Sundip", "Anika"]
for count = 1 to names.length - 1
    pos = count
    while (pos > 0 and names[pos] < names[pos - 1])
        temp = names[pos]
        names[pos] = names[pos - 1]
        names[pos - 1] = temp
        pos = pos - 1
    endwhile
next count
```

(a) Describe the purpose of the variable `temp` in the insertion sort pseudocode algorithm.

....

....

....

.... **[2]**

This question was an excellent discriminator in terms of candidate achievement. Many responses correctly described the `temp` variable as storing the name so that it could be overwritten during the process of swapping values. Partial credit was given for responses simply commenting on it storing data, as this is essentially the purpose of any variable.

However, credit was not given where candidates discussed storing a position/index, as this was not the case; the `pos` variable is used as the index of the array. The content of this array index (a name) is stored in the `temp` variable.

Where responses are open to interpretation, the mark scheme attempts to credit as many of these as possible. However, incorrect responses (such as reference to the position here instead of the name) are not credited.

Question 3 (b)

(b) An insertion sort contains a nested loop; a loop within a loop. In this pseudocode algorithm the outer loop is a count-controlled loop and the inner loop is a condition-controlled loop.

Explain why the inner loop needs to be a condition-controlled loop.

....

....

....

.... **[2]**

Previous questions such as Question 2(a) assessed knowledge (AO1). This question focused on the application (AO2) of a technique (condition-controlled loops) to the algorithm given (an insertion sort). This assessed candidates' understanding of the process an insertion sort follows to sort values.

The J277 specification is clear that candidates do not have to remember the code for this algorithm. But the specification does state that candidates must be able to understand the main steps of the algorithm and identify the algorithm if given the code for it. In this case, candidates were given all of the code for the algorithm.

Candidates generally found this question challenging. Many responses simply repeated the question and discussed sorting values. More successful responses understood that the inner loop is the part of the algorithm that moves the `name` repeatedly in the list and that we do not know how many times this move needs to be made. Further to this, it is the condition that the `name` is in the correct position (or even better, an explanation of how this is decided on) which ends the loop.

Assessment for learning

When asked to explain why a particular technique is used, it is often useful for candidates to think about why alternative options have **not** been used. In this question, thinking about what would happen if a count-controlled loop had been used as the inner loop may have given candidates the insight to discuss why a condition-controlled loop has been used.

Question 3 (c) (i) and (ii)

(c) A bubble sort is another type of sorting algorithm.

(i) Describe **one** difference between an insertion sort and a bubble sort.

....

....

....

.... **[2]**

(ii) Describe **two** similarities between an insertion sort and a bubble sort.

1 ....

....

2 ....

....

[2]

Question 3(c)(i) and Question 3(c)(ii) asked candidates to describe one difference and two similarities between an insertion sort and a bubble sort.

Examiners were instructed to be generous in their interpretation of the requirements of both algorithms. This included considering both their algorithmic implementation and the understanding of how these are typically described on a classroom whiteboard.

As such, many combinations of answers could gain the marks available.

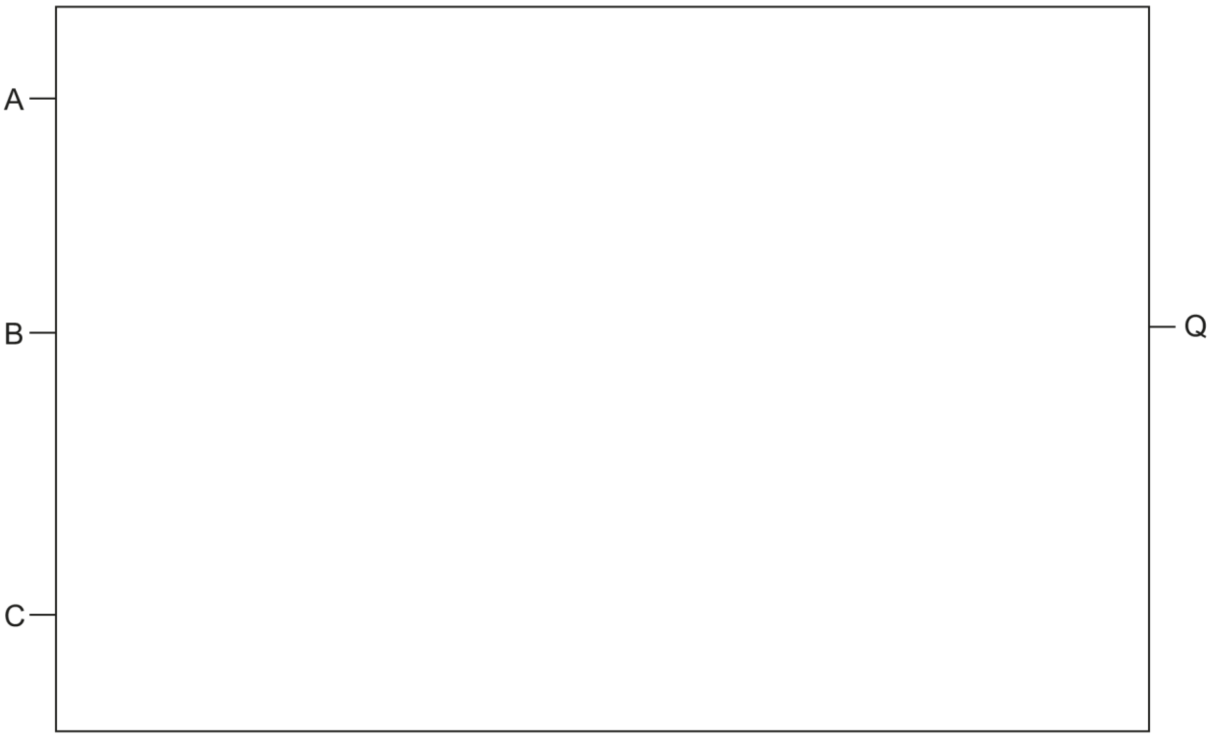
Question 4 (a)

4 A garden floodlight system uses inputs from sensors and switches to decide whether it should be turned on.
The table shows the inputs into the system and the meaning of each input value:

Letter	Input device	Input of 1	Input of 0
A	Motion sensor	Motion is detected	Motion is not detected
B	Light sensor	Light levels indicate it is daytime	Light levels indicate it is nighttime
C	Light switch	The switch is turned on	The switch is turned off

The floodlight (Q) is designed to be on (Q = 1) when the switch is turned on and the motion sensor detects motion at nighttime.

(a) Draw a logic diagram for the floodlight.



[3]

This logic question is based around a typical outdoor light system. The majority of candidates were able to show understanding that both the switch (input C) and the motion sensor (input A) need to be triggered for the light (output Q) to be on. However, some candidates missed the description that this only occurs at nighttime. Input B is the light sensor and so a NOT gate is required on this input to fulfil this requirement.

Given that AND gates generally have two inputs and this system uses three inputs, two AND gates were required. The mark scheme credits any and all arrangements of these to reach the correct output.

Examiners were also instructed to credit use of a 3 input AND gate.

Question 5 (a) (i)

5 Charlie is developing an adding game. The rules of the game are:

- the player is asked 3 addition questions
- each question asks the player to add together two random whole numbers between 1 and 10 inclusive
- if the player gets the correct answer, 1 is added to their score
- at the end of the game their score is displayed.

(a) Charlie has been told that the game will need to be tested before giving it to the players.

(i) Explain why programs should be tested before use.

....

....

....

.... **[2]**

Testing as a process could be done for many reasons. The mark scheme attempts to credit as many sensible explanations for this as possible. This includes testing to find errors, checking for robustness and checking against user requirements. Furthermore, fixing errors that are found is also credited as this could form part of the testing process.

The majority of responses demonstrated an understanding of this and achieved highly.

Question 5 (a) (ii)

(ii) Complete the table by naming and describing **one** type of test that should be used on Charlie's program before releasing it.

Test type	Description

[2]

The J277 specification lists iterative and final/terminal testing as test types. However, many candidates interpreted this question as asking about test data (such as normal or erroneous data). Where candidates described the use of test data and link it to expected outcomes, this was credited by examiners.

Other types of suitable testing that do not appear on the J277 specification (such as white box/black box testing, alpha/beta testing) were also accepted.

Assessment for learning

The J277 specification states the minimum content that candidates are expected to know and understand at GCSE level. However, it is possible for teachers to go beyond this.

For example, iterative and final/terminal testing are stated in the specification. However, there are other types of testing. Other technically correct responses will be accepted by examiners even if they do not appear in this specification.

Another example is sorting algorithms. Merge sort, bubble sort and insertion sort appear in the specification. However, when candidates have been asked to name sorting algorithms, previous mark schemes credit other valid responses (such as quick sort, selection sort or bogo sort).

Question 5 (a) (iii)

(iii) Complete the table by identifying **and** describing **two** features of an IDE that can be used when testing a program.

Feature	Description

[4]

This question was generally well answered with a variety of features given. The question specifically asks about features used when testing a program. Therefore, features such as debugging tools, stepping and variable watch windows were very common responses.

However, more general responses were also accepted, such as text editors, translators, and keyword completion. These could all potentially be used when editing programs after errors had been identified.

Less successful responses tended to be descriptions that simply repeated the name of the feature given. For example "debugging tools, to allow debugging" would gain 1 mark for the feature identification but not the description.

Question 5 (b)

(b) Validating inputs can reduce errors when a program is being run.

Identify **two** methods of validation **and** explain how they can be used on this game.

Validation method 1 ....

Use ....

....

....

Validation method 2 ....

Use ....

....

....

[6]

Responses which focused on the explicit link to the game described in this question tended to do well.

The stronger responses stated a validation method and linked the use of the validation method to the game. For example, a requirement of the game is that two random numbers between 1 and 10 are picked. It is sensible to suggest that validation ensuring the total is between 2 and 20 could be implemented. Further discussion relating to how this could be done, even as far as suggesting sensible high-level code that could be used would have developed the response.

Some candidates gave examples of validation which did not clearly link to the game. Generic examples were partially credited. Explicit links to the game were required in order to gain all marks available.

Misconception

Input validation applies to values input by the user. In this case, the requested input is the sum of two numbers, each of which are between 1 and 10. It is not necessary to validate the random number generation (as this has not been input) and it would be inappropriate to limit user inputs to between 1 and 10; the total could easily be (for example) 8 + 6 = 14.

Where candidates suggested validating inputs to allow between 1 and 10, not all marks available were given due to this misconception.

Question 5 (c)

(c) Write an algorithm to play this game. The rules are repeated from the start of the question here:

- the player is asked 3 addition questions
- each question asks the player to add together two random whole numbers between 1 and 10 inclusive
- if the player gets the correct answer, 1 is added to their score
- at the end of the game their score is displayed.

....

....

....

....

....

.... **[6]**

As this question appears in Section A, candidates are free to respond in any suitable way, including using flowcharts, structured English, pseudocode or a high-level language.

The majority of high scoring responses used a high-level language consistently.

Where flowcharts or structured English were used, responses needed to clearly show the steps to be taken and not simply repeat the question to achieve marks.

The given question is already decomposed for candidates and many were able to use these bullet points to build a solution that achieved the majority of marks available.

Many responses used random number generation and iteration to create an elegant response that met all mark points. This was pleasing to see and it is extremely encouraging that candidates can use techniques such as these where appropriate without being prompted.

Other responses manually repeated asking the required questions; on this occasion, these were also credited and could have achieved full marks.

Where a mistake was made in one section (such as with iteration), examiners were instructed to use FT (follow through) where possible. This allowed candidates to score marks in later sections if their responses were logically constructed. This is to be fair to candidates so that mistakes are only penalised once in any given question.

A significant number of responses did not access many marks in this question. This would suggest that more practical programming time in lessons would be beneficial.

Section B overview

Section B consists of multiple questions. Questions are based on a given scenario. This year the scenario was a security company.

Questions will always be set within a single scenario for Section B.

Some questions require candidates to respond using either a high-level language that has been studied or OCR Exam Reference Language. If a candidate responded to these questions using flowcharts or structured English, no marks were given.

It is important for centres and candidates to understand that examiners **do not** penalise answers if they do not work in a particular language. Candidates do not state which language they use. Therefore, answers are marked for their logical correctness and consistency.

This includes where answers combine features from multiple high-level languages.

Question 6 (b)

(b) The alarm has an algorithm that decides whether to sound the alarm by checking the data that is stored in the following three variables.

- `SystemArmed`
- `DoorSensorActive`
- `WindowSensorActive`

The alarm will only sound when the alarm has been activated **and** one or both of the door and window sensors are activated. When the system needs to sound the alarm it calls the pre-written procedure `SoundAlarm()`

Write a program that checks the data in the variables and calls `SoundAlarm()` when appropriate.

You must use **either**:
- OCR Exam Reference Language, **or**
- A high-level programming language that you have studied.

....

....

....

....

....

.... **[4]**

Many responses achieved highly on this question. The question asks for a simple program to be written that checks the given variables and calls the given procedure when necessary.

Examiners were instructed to be generous with the first mark, crediting any use of selection or condition-controlled iteration. Responses may therefore have been rewarded for an attempt at this question even if their solution was not fully functional.

Centres should encourage candidates to attempt each question for precisely this reason; it is typical that a small number of marks are allocated to attempting a solution on many programming questions for J277/02.

A significant number of responses were given 3 out of 4 marks as they misunderstood the role of operator precedence in their solution; this is detailed in the "misconception" box below.

Misconception

Where multiple conditions are used in selection, these have an order of precedence very much like BIDMAS does in mathematics; an `AND` operator will always take precedence over an `OR` operator. A `NOT` operator (not used in this question) would have even higher precedence.

This can cause problems in candidate responses. A common candidate response was:

```
if SystemArmed AND DoorSensorActive OR WindowSensorActive then
    SoundAlarm()
```

However, because the `AND` operator takes precedence, the first check done here is if the system is armed and the door sensor is active. The result of this is then evaluated with an OR operator to check if the window sensor is active.

This results in the alarm sounding if the window sensor is active, even if the system is not armed. This was clearly not the candidate's intention.

To fix this, candidates could have either:

- put brackets/parentheses around the `Door OR Window` section of their response
- written the response as separate checks. This could have been done in multiple ways, including nested `if` statements or repeated checks.

Exemplar 1

```
if SystemArmed:
    if DoorSensorActive or WindowSensorActive:
        SoundAlarm()
```

Exemplar 1 shows one way that full marks are achieved on this question. The candidate has used nested `if` statements to check if the system is armed, and if true, then checking if either sensor has been activated. The `SoundAlarm()` procedure is only called if both `if` statements evaluate to True.

Question 6 (c) (i)

(c) The alarm system can also have motion sensors. Each type of sensor has a code. The code for each sensor is given in the table:

Code	Sensor
MS	Motion sensor
DS	Door sensor
WS	Window sensor

A program is written to reset the sensors. The program:

- asks the user to enter the code for the sensor they want to reset
- calls the prewritten function `CheckSensorCode()` to check whether the code entered is a valid code
- the sensor number is read as input if the code is valid and the function `ResetSensor()` is called for the sensor

```
01  sensorType = input("Enter code of the type of sensor to reset")
02  if(CheckSensorCode(sensorType)) then
03      sensorNumber = input("Please input the number of the sensor
                                to reset")
04      sensorID = sensorType + sensorNumber
05      ResetSensor(sensorID)
06  endif
```

(i) Give the line number where there is concatenation.

.... **[1]**

Concatenation is the process of joining strings together. In OCR Exam Reference Language, this is done using the + symbol. Line 04 joins together `sensorType` and `sensorNumber`, assigning the concatenated result to the variable `sensorID`.

Question 6 (c) (ii)

(ii) Give the identifier of a variable used in the program.

.... **[1]**

The vast majority of candidates are confident with identifying variable identifiers, such as `sensorNumber`. Small errors in spelling or spacing were not penalised. Candidates should be encouraged to be accurate with their namings, particularly if these are already given to them in the question.

Question 6 (c) (iii)

(iii) Identify the data type of the data returned by the function `CheckSensorCode()`

.... **[1]**

Many candidates found this question challenging. The question relies on understanding of how selection statements operate. The answer of Boolean can be inferred from how the function return value is used.

This use of Boolean values in selection statements also caused confusion in the previous J276 specification. It would be beneficial for centres to cover this specifically. More detail is given in the misconception box below.

Misconception

Where Boolean values are used in selection, there is no need to compare this to a `True` or `False` value. Line 02 uses an IF statement based on the return value from a function `CheckSensorCode()`. Because the function returns a Boolean value, the IF statement is valid.

Looking at another example, if the variable `x` is Boolean, then both of the following would be valid:

- `if x == True then…`
- `if x then…`

In both cases, if the value of `x` is `True`, the code underneath would be executed.

The second version (without comparing a Boolean value to `True` or `False`) is more elegant. However, both would be accepted as responses in an examination.

Question 6 (c) (iv)

(iv) Give the line number that contains a function call.

.... **[1]**

The program code given contains two explicit calls to functions. There is one function call on line 02 for `CheckSensorCode()` and one function call on line 05 for `ResetSensor`(). However, many programming languages (including Python) implement input as a function and so examiners were instructed to also give credit where candidates identified lines 01 or 03.

Question 6 (c) (v)

(v) Identify **two** programming constructs that have been used in the program.

1 ....

2 ....
[2]

The three programming constructs given in the specification are selection, sequence and iteration.

Sequence and selection are used within this program.

There is no use of iteration in the given program.

Surprisingly for an AO1 question, this proved relatively challenging for candidates with many identifying other features of the code, such as inputs, function calls or variables.

Question 6 (d)

(d) The alarm system has a log that stores a record each time a sensor is triggered. This is called an event. The record format is given in the table:

Fieldname	Description
`Date`	The date the event happened
`SensorID`	The sensor that was activated
`SensorType`	The type of sensor that was activated – Door, Motion or Window
`Length`	The number of seconds the sensor was triggered (to the nearest second)

The log is stored in a database table called `events`. The current contents of `events` is shown:

Date	SensorID	SensorType	Length
05/02/2023	WS2	Window	38
05/02/2023	MS1	Motion	2
06/02/2023	DS3	Door	1
06/02/2023	MS2	Motion	3
06/02/2023	MS1	Motion	2
07/02/2023	WS1	Window	24
07/02/2023	DS1	Door	1

Write an SQL statement to display the sensor IDs of the door sensors that have been triggered for more than 20 seconds.

....

....

....

....

....

.... **[3]**

Structured Query Language is obviously well understood by many candidates. Many high-quality responses were produced.

Most responses were able to use SELECT and FROM appropriately to produce a logically correct response. However, the vast majority of responses missed off the requirement that only door sensors were required to be included, gaining 2 out of 3 marks in the process.

Although a suggested response is shown in the mark scheme, any logically correct SQL that produces the required output would be accepted by examiners.

Where a mistake was made consistently (such as using colons after the SQL keyword), this was penalised once and then FT (follow through) allowed for subsequent marks.

Question 6 (e)

(e) A program written in a high-level language is used to access the data from the database. This program has a procedure, `SaveLogs()`, that stores the data to an external text file.

The procedure `SaveLogs()`:
- takes the string of data to be stored to the text file as a parameter
- takes the filename of the text file as a parameter
- stores the string of data to the text file.

Write the procedure `SaveLogs()`

You must use **either**:
- OCR Exam Reference Language, **or**
- A high-level programming language that you have studied.

....

....

....

....

....

.... **[6]**

This question proved to be challenging for many candidates. The question combined defining a procedure with the use of text files.

The tasks required were partially decomposed in the bullet points. A candidate attempting these in order would have achieved a significant number of marks.

Candidates could also have achieved numerous marks for a partial solution (e.g. defining a procedure that didn't use text files or writing to a text file outside of a procedure) and the mark scheme was deliberately constructed to credit these responses.

Full marks were often given where candidates appear to have had practical experience of these two techniques.

Exemplar 2

```
procedure SaveLogs(data, fileName)
    file = open(fileName)
    file.writeLine(data)
    file.close()
endprocedure
```

Exemplar 2 shows a response that scored full marks. The procedure has been defined with multiple parameters which are then used to open the file and to write the data. The candidate has also achieved the bullet point 7 on the mark scheme (closing the file) but this wasn't necessary in this case.

Question 6 (f) (i)

(f) OCR Security Services need to identify the total number of seconds the sensors have been activated on a specific date.

The data from the database table `events` is imported into the program written in a high-level programming language.

The program stores the data in a two-dimensional (2D) string array with the identifier `arrayEvents`

The data to be stored is shown in the table.

Date	SensorID	SensorType	Length
05/02/2023	WS2	Window	38
05/02/2023	MS1	Motion	2
06/02/2023	DS3	Door	1
06/02/2023	MS2	Motion	3
06/02/2023	MS1	Motion	2
07/02/2023	WS1	Window	24
07/02/2023	DS1	Door	1

In this table, the value of `events[1, 1]` contains "`MS1`".

(i) An array can only store data of one data type. Any non-string data must be converted to a string before storing in the array.

Identify the process that converts integer data to string data.

.... **[1]**

The use of the term "casting" to convert one data type to another is now well known and understood by candidates. This is given and referred to in the J277 specification and is essential knowledge.

Question 6 (f) (ii)

(ii) Write a program that:

- asks the user to input a date
- totals the number of seconds sensors have been activated on the date input
- outputs the calculated total in an appropriate message including the date, for example:
  `Sensors were activated for 40 seconds on 05/02/2023`

You must use **either**:
- OCR Exam Reference Language, **or**
- A high-level programming language that you have studied.

....

....

....

....

....

.... **[6]**

The final question in Section B is expected to be a high demand question.

The techniques required (iteration through a 2D array, selection, keeping a running total of times) are within the specification but it is acknowledged that the level of challenge was high.

Examiners were instructed to give marks for an attempt at a solution (as with previous questions).

For this question marks were given for:

- any attempt at selection
- any solution that accessed each element in the given array, even if this was via a manual process.

Therefore, many candidates gained multiple marks for an attempt that only partially solved the problem.

A significant number of candidates were able to create a solution that fully met the requirements of the question. This was often done in an elegant and efficient manner.

This is extremely pleasing and shows excellent understanding and significant experience of practical programming.

Exemplar 3

total=0
date = input ("Enter a date")
~~toto~~
~~count = 0~~
for count = 0 to arrayEvents.length
    if arrayEvents [0, count]
        == date then
        total = total + arrayEvents
            [3, count]
    endif
~~endfor~~
print ("Sensors were activated for",
    total, "Seconds on", date)

Exemplar 3 shows a high scoring response. A date has been asked for as input which has then been used to compare to each element at position 0 in the array.

Where any of these match, the total variable is updated to keep a running total of the corresponding element at position 3 in the array.

After each element has been checked, the total and date are output in a suitable message.

This is not the only method by which a response could be given full marks but is perhaps the most common.

Supporting you

Teach Cambridge

Make sure you visit our secure website **Teach Cambridge** to find the full range of resources and support for the subjects you teach. This includes secure materials such as set assignments and exemplars, online and on-demand training.

Don't have access? If your school or college teaches any OCR qualifications, please contact your exams officer. You can forward them this link to help get you started.

Reviews of marking

If any of your students' results are not as expected, you may wish to consider one of our post-results services. For full information about the options available visit the OCR website.

Access to Scripts

For the June 2023 series, Exams Officers will be able to download copies of your candidates' completed papers or 'scripts' for all of our General Qualifications including Entry Level, GCSE and AS/A Level. Your centre can use these scripts to decide whether to request a review of marking and to support teaching and learning.

Our free, on-demand service, Access to Scripts is available via our single sign-on service, My Cambridge. Step-by-step instructions are on our website.

Keep up-to-date

We send a monthly bulletin to tell you about important updates. You can also sign up for your subject specific updates. If you haven't already, sign up here.

OCR Professional Development

Attend one of our popular CPD courses to hear directly from a senior assessor or drop in to a Q&A session. Most of our courses are delivered live via an online platform, so you can attend from any location.

Please find details for all our courses for your subject on **Teach Cambridge**. You'll also find links to our online courses on NEA marking and support.

Signed up for ExamBuilder?

ExamBuilder is the question builder platform for a range of our GCSE, A Level, Cambridge Nationals and Cambridge Technicals qualifications. Find out more.

ExamBuilder is **free for all OCR centres** with an Interchange account and gives you unlimited users per centre. We need an Interchange username to validate the identity of your centre's first user account for ExamBuilder.

If you do not have an Interchange account please contact your centre administrator (usually the Exams Officer) to request a username, or nominate an existing Interchange user in your department.

Active Results

Review students' exam performance with our free online results analysis tool. It is available for all GCSEs, AS and A Levels and Cambridge Nationals.

Find out more.

Need to get in touch?

If you ever have any questions about OCR qualifications or services (including administration, logistics and teaching) please feel free to get in touch with our customer support centre.

Call us on
01223 553998

Alternatively, you can email us on
support@ocr.org.uk

For more information visit
- **ocr.org.uk/qualifications/resource-finder**
- **ocr.org.uk**
- **facebook.com/ocrexams**
- **twitter.com/ocrexams**
- **instagram.com/ocrexaminations**
- **linkedin.com/company/ocr**
- **youtube.com/ocrexams**

We really value your feedback

Click to send us an autogenerated email about this resource. Add comments if you want to. Let us know how we can improve this resource or what else you need. Your email address will not be used or shared for any marketing purposes.

I like this **I dislike this**

Please note – web links are correct at date of publication but other websites may change over time. If you have any problems with a link you may want to navigate to that organisation's website for a direct search.

CAMBRIDGE
UNIVERSITY PRESS & ASSESSMENT