Qualification Accredited

OCR
Oxford Cambridge and RSA

Examiners' report

GCSE

COMPUTER SCIENCE

OCR Level 1/Level 2 GCSE (9-1) in Computer Science

J277

For first teaching in 2020

J277/02 Summer 2025 series

Version 1

ocr.org.uk/computerscience

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 responses 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 Teach Cambridge.

Paper 2 series overview

This is the fourth examination series for the J277 specification.

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

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

To do well on this paper, 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 programming language that they are familiar with.

In this series there was a noticeable increase in ability with respect to candidate responses to programming questions. Candidates also showed more developed understanding and resilience.

This year there were far fewer blank responses and the majority of candidates attempted most parts of even the tougher questions. This is extremely pleasing and centres and candidates should be applauded for this

It is hoped that previous Examiners' Reports highlighting the importance of practical programming experience within centres has had an impact on this. Programming experience in a practical setting definitely supports candidates to confidently answer questions in this paper.

Both creating and using functions, and SQL syntax remain areas for improvement which could be developed within centres' teaching.

Centres are encouraged to be aware of the contents of the specification, particularly Section 3c which shows details of OCR Exam Reference Language. Although candidates do not have to use this in their responses, questions will be presented in this way and candidates should be aware of these conventions in order for them to successfully understand and access examination questions.

Candidates who did well on this paper generally:	Candidates who did less well on this paper generally:
• attempted each question, particularly the programming questions in Section B. • understood the syntax required for SQL and were able to refine incorrect statements, applying these rules. • were able to define and create functions, using a parameters to perform calculations, before returning a value • were precise with definitions and instructions given.	• created selection (IF) statements that were logically incorrect when using multiple conditions joined by AND or OR. • were unclear in their understanding of a merge sort when showing the steps that a merge sort takes on a data set. • were imprecise when creating programs in a high-level programming language, including not checking boundary cases (such as when the exact number of tickets available was requested in Question 6 (e)).

Section A overview

Section A consists of multiple questions and scenarios. Candidates are free to respond in any suitable way when asked to design algorithm questions. They may use flowcharts, structured English, pseudocode or a high-level programming language. The majority of candidates who scored highly tended to use a high-level programming language consistently throughout the examination paper.

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

Question 1 (a)

1 A program asks a user to enter a number. Only integer values between 1 and 100 (inclusive) are accepted. The program is tested.

(a) Tick (✓) **one or more** boxes in each row to identify the type or types of test for each example of test data.

Example test data	Normal test data	Boundary test data	Invalid / erroneous test data
27			
"Hello"			
105			
100			

[4]

This question assessed candidates' understanding of different types of test data: normal, boundary, and invalid/erroneous.

The vast majority of candidates performed very well on this question, demonstrating a secure understanding of the definitions and application of each test data category. It was clear that this topic had been well taught across centres, with many candidates achieving full marks.

Use of Invalid/erroneous test data

Although the specification defines invalid data and erroneous data separately, questions in examinations will always group these together as data that a system would reject, as shown in this question.

Question 1 (b) (i)

(b) An algorithm is written for the program.

(i) Complete the algorithm for the program.

```
num = input("Enter a number between 1 and 100")

if num ..... AND ..... then

      print("accepted")

.....

      print("not accepted")

endif
```

[3]

Most candidates were able to correctly identify the purpose of the IF statement and therefore complete the third gap (**`else`**) with ease. Many candidates also correctly completed the first gap as a simple comparison with the variable **`num`** that was already given.

However, the second gap proved more challenging. A common mistake was to write **`<= 100`** without referencing the variable a second time, which is not syntactically valid.

Another frequent error involved misunderstanding the inclusivity of the boundaries. Some candidates used **`num > 1 AND num < 100`**, which would **exclude** valid inputs of 1 and 100. This suggests a need for further emphasis on the difference between inclusive and exclusive comparisons when validating input ranges.

Misconception

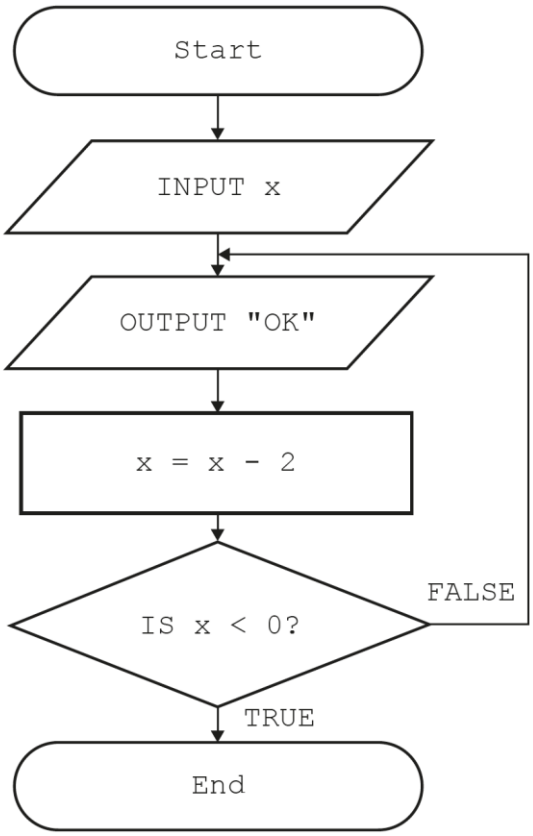
Selection statements using two conditions joined together with a Boolean condition such as AND must make sure that both conditions are logically valid comparisons. In many cases, this will mean referencing a variable twice, once in each condition.

For example, **`if num >= 1 AND <= 100`** is incorrect because the second condition is **not** a valid comparison - the value **`num`** must be included in the second part of the comparison.

The correct statement would be **`if num >= 1 AND num <= 100`**.

Question 2 (a)

2 This flowchart uses iteration.



(a) Complete the table by identifying how many times "OK" will be output for when each value is input.

Input value	Number of times "OK" is output
3	
10	

[2]

This question involved following a flowchart to determine the outcome produced. Most students did this successfully for the input value of 3, determining correctly that OK would be output twice.

However, fewer candidates were able to do this for the input value of 10. This may be due to confusion over the use of the 'less than' operator when the value held in variable **x** was set to zero.

Question 2 (b) (i)

(b) Iteration is a programming construct.

(i) Identify **and** describe **two** different examples of iteration that can be used in a high-level language.

1 .....

.....

.....

2 .....

.....

.....

[4]

Iteration is commonly referred to as loops, or repetition. Candidates were confidently able to name examples of keywords that are used in a high-level language (such as **`for`**, **`while`** or **`do until`**) and describe the use of these well. Many candidates used subject-specific terminology such as **condition controlled** loops or **count controlled** loops when doing this, as well as describing these in more depth.

Some candidates lacked precision with their description of a **`while`** loop, stating that this repeats **until** a condition is **`True`** which is incorrect.

Misconception

A **`while`** loop is condition controlled. It has a condition that is evaluated before each iteration begins; if the condition evaluates to be **`True`**, the loop repeats one more time before the condition is evaluated again.

For example, **`while x > 7`** : The condition here that is evaluated is whether the contents of the variable `x` is larger than 7. As long as this is the case, the loop will continue to repeat.

Many candidates suggested that the while loop should repeat **until** the condition is **`True`**, that is, **`while x > 7`** should repeat for values of **`x`** that are not larger than 7 and only stop repeating when **`x`** is larger than 7. This is clearly incorrect.

Centres and candidates must be aware that their answers should be precise and technically correct if they are to achieve highly.

Question 2 (b) (ii)

(ii) Give the names of **two** other programming constructs that are used in a high-level language.

1 .....

2 .....

[2]

Many candidates were able to list sequence and selection as two other programming constructs which are listed in the specification document. We accepted sensible alternatives (such as branching).

Question 2 (c)

(c) A programmer implements the flowchart as a program in a high-level language using an Integrated Development Environment (IDE).

Identify and describe **two** tools that the programmer could use in the IDE to create a program from the flowchart.

Tool 1 .....

Description .....

.....

Tool 2 .....

Description .....

.....

[4]

Candidates were mostly able to name and describe suitable tools contained within an Integrated Development Environment, such as an editor, error diagnostics, run-time environment or translator.

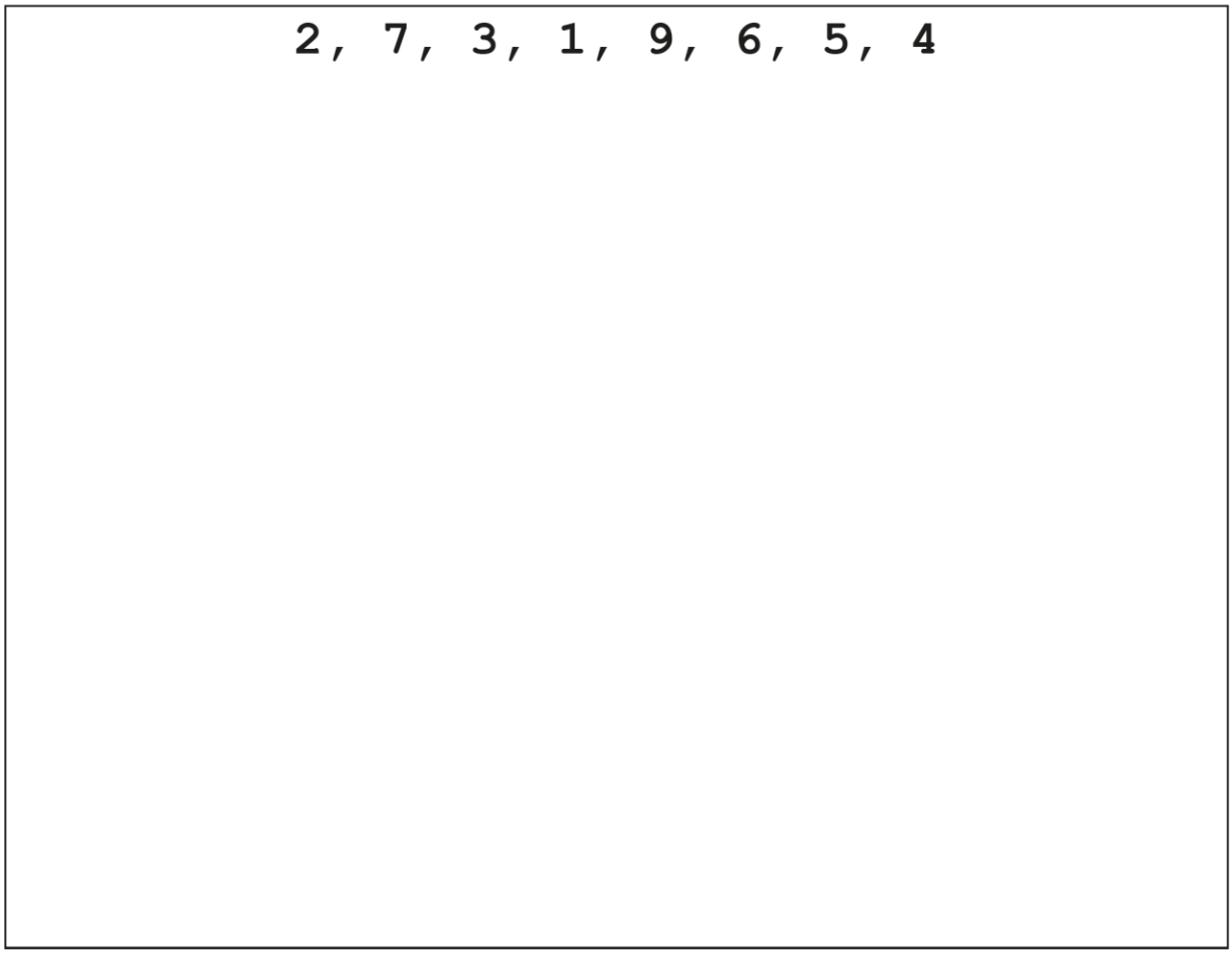
Many other sensible tools and sub-tools such as a variable watch window, stepping, keyword completion or Artificial Intelligence (AI) tools were all also credited.

A small number of candidates mistakenly described tools contained within programming languages, such as commenting or use of variables. Descriptions were generously given where they added to the tool name, but repetition (such as "a code editor / edits code") was not given both mark parts.

Question 3 (a)

3

(a) Show the steps that a merge sort would take to put the data set into ascending numerical order.



[4]

The majority of candidates showed clearly that they understood the steps a merge sort takes to sort values into ascending order. The strongest responses clearly showed both the divide stage (splitting values up into individual lists) and then the conquer stage (merging lists together in order).

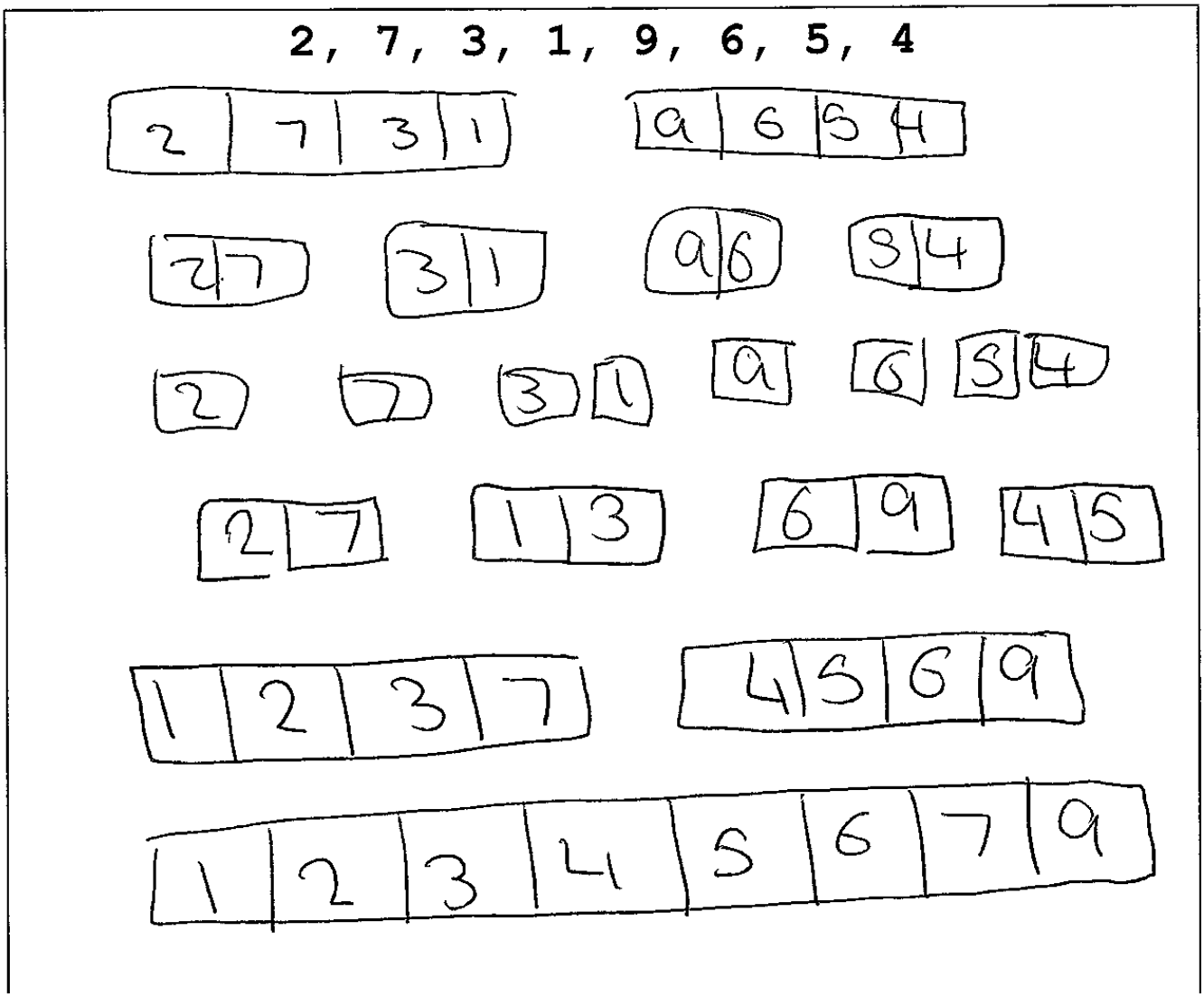
Only a minority of candidates attempted a different sorting algorithm or showed a lack of understanding of the abstract idea behind the algorithm; this is very pleasing to see.

However, several candidates still merge lists together into an unsorted list and then take a second step to sort the list. This has been commented on in previous Examiner's Reports but remains a common misconception. Where this was the case in candidate answers, the mark scheme instructed examiners not to give the relevant mark.

Misconception

A merge sort merges smaller lists together to form a larger sorted list. At each step of the merge process, **all** lists are **always** in order; it is the process of merging together that performs the sort. A very common misconception is for candidates to merge lists together by simply appending one to the other (to form an unsorted list) and then sorting this list. Of course, this begs the recursive question - what algorithm is being used to carry out the sort?

Exemplar 1



In this example, the candidate has successfully achieved full marks. The list is repeatedly divided into smaller and smaller lists until lists of size one are present. These are then merged together into lists of size 2, then lists of size 4 before finally being merged into one final list of size 8.

Note that it is the act of merging that does the sorting; values are **not** merged out of order and then sorted.

The candidate puts boxes around their numbers to clearly indicate the different lists. This is an example of best practice.

Question 3 (b)

(b) Tick (✓) **one** box in each table to complete the descriptions.

An insertion sort…

	Tick (✓) **one** box
uses an array of numbers with a sorted part and unsorted part	
picks out the middle value to begin with	
merges arrays of numbers together	

A bubble sort compares pairs of values and then:

	Tick (✓) **one** box
always swaps them	
swaps them if they are in the correct order	
swaps them if they are in the incorrect order	

A sorting algorithm can sort data in an array. An array:

	Tick (✓) **one** box
is declared with a fixed length	
stores multiple values that cannot be changed while the program is running	
can only be declared with a maximum of one-dimension (1D)	

[3]

The majority of candidates were able to select the correct answers for the first two tick boxes, showing good understanding of sorting algorithms. Fewer candidates were able to define an array as being declared with a fixed length. This may be due to Python being the most common programming language used, and Python support **lists** by default instead of arrays.

Lists are dynamic data structures and so can be appended to, whereas arrays are static data structures with a fixed length.

Assessment for learning

If Python is used as the main programming language in a centre, care must be taken to make sure that elements of the specification not easily covered in Python, (e.g. arrays, `do...until` loops, `switch/case`) are covered suitably in a different way – for example, perhaps by introducing a second programming language.

Note : `Switch/case` is supported in Python version 3.10 and later using the `match` and `case` keywords.

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

(c) An algorithm can be used to search for a value in an array. The algorithm follows these steps:

- compares the first value in the array to the value being searched for
- stops if the comparison is `true`
- moves to the next value in the array if the comparison if `false`.

(i) Give the name of the searching algorithm described.

..... **[1]**

(ii) Identify **one other** condition that will cause the algorithm to stop.

..... **[1]**

Most candidates were able to identify the algorithm given as a linear search.

The other stopping condition for part (ii) had to be precisely and correctly defined by candidates. Many responded "when the value is not found", but how would this be written as a condition for a high-level programming language to check? The condition is technically when the end of the list is reached **and** the value has not been found, but responses relating to simply reaching the end of the list or having checked each item were accepted.

Question 4 (a) and (b)

4 An algorithm is written to read numbers from the external text file `data.txt`

Each number is checked when it is read from the external text file. The number is output to the user if it is not 0.

(a) Complete the algorithm using the OCR Exam Reference Language statements in the box.

Not all statements are used.

`data`	`for`	`if`
`myFile.close()`	`myFile.readLine()`	`input()`
`myFile.writeLine()`	`open`	`print`
`temp`	`txt`	`x`

```
myFile = ..... ("data.txt")
while NOT myFile.endOfFile()
    temp = .....
    if int (.....) != 0
        .....(temp)
    endif
endwhile
.....
```

[5]

(b) Identify how casting is used in this algorithm.

.....

..... **[1]**

Casting was well understood by most candidates. Casting converts one data type to another data type. This question was an AO2 question and so requires an answer in context.

Candidates were required to identify how casting is used **in the given algorithm.** In this case, this meant identifying that a value was converted to an integer (on the 4th line). Candidates could have identified any value or variable to be converted to an integer, as they were not double penalised if they had incorrectly answered the previous question.

Question 5 (a)

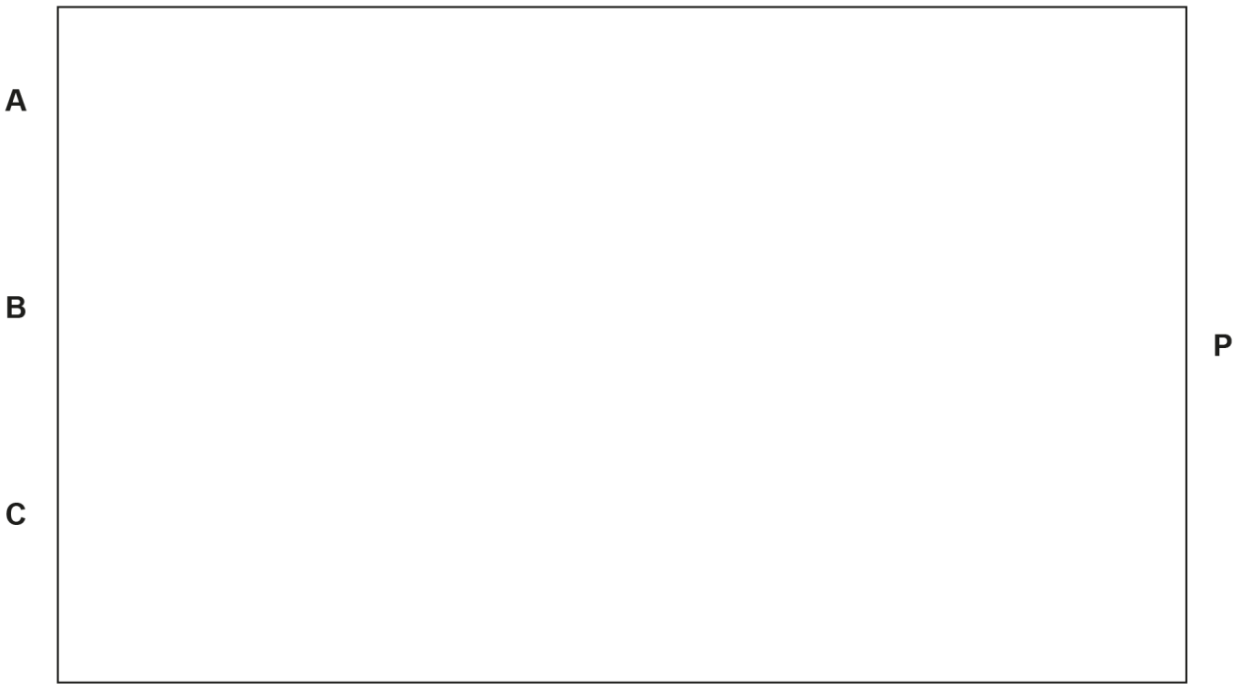
5 A security system protects online credit card payments. The system has three inputs:

Input	Description
A	`True` when the user enters their correct password
B	`True` when the user enters their correct PIN
C	`True` when the card is blocked

The output (`P`) is `True` if both of the following conditions are `True`:

- the user has entered either their correct password or has entered their correct PIN
- the card is **not** blocked.

(a) Draw the logic circuit for the security system



[3]

This question was extremely well answered by candidates, showing excellent understanding of constructing logic gate diagrams from a real-world scenario.

Misconceptions or mistakes were rare, but where they did occur they tended to be around shapes of gates. Candidates should be aware that the circle on the NOT gate is crucial (as otherwise a triangle with no circle is simply a buffer) and that AND and OR gates do not need circles, or they would instead of NAND or NOR gates.

Examiners are instructed to mark the shapes of the gates and ignore any annotation, so it is crucial that candidates can accurately draw shapes that are clearly AND and OR gates.

Question 5 (b)

(b) Complete the truth table for **P = A AND B**

A	B	P

[2]

This question was also extremely well answered. Most candidates filled in columns A and B with every possible combination of inputs. 1s and 0s were the expected options, but Yes/No and True/False were also accepted if given by candidates.

Inputs do not have to be in any particular order, but best practice to start with 00 and then essentially count up in binary, through 01, 10 then 11. This means that where larger systems are presented (perhaps with 3 or 4 inputs), no input combination is missed.

The final column (P) should have contained the output for the system, and with P = A AND B, this means that only the row with A and B as both 1 should have had a 1 output, with every other output being 0.

Question 5 (d)

(d) As part of the security system, the user is asked to choose a 4-digit value for their PIN. This PIN cannot be 1234 or 4321

Create an algorithm that:

- takes the PIN as input as a string
- checks that the PIN is 4 characters in length
- checks that the PIN is **not** either "1234" or "4321"
- outputs `"VALID PIN"` if the PIN meets all of the criteria
- outputs `"INVALID PIN"` if the PIN does not meet all of the criteria.

You do **not** need to check that the digits entered are numeric.

.....

.....

.....

.....

.....

..... **[6]**

As a question in Section A, candidates are free to use a high-level programming language, flowcharts or structured English. Pleasingly, the vast majority attempted to use a high-level programming language and this proved to be very successful for many.

Candidates proved able to take input as a string, to use in-built length functions to check for length and to output messages based on these checks.

The major misconception that affected many students related to checking two invalid PINs, 1234 or 4321. Where two checks are done within a single IF statement it is important that each condition **must** be a valid comparison and therefore include the variable to check against for both comparisons.

This is the same misconception highlighted on Question 1 (b) (i).

Misconception

Selection statements using two conditions joined together with a Boolean condition such as OR must make sure that both conditions are logically valid comparisons. In many cases, this will mean referencing a variable twice, once in each condition.

For example, `if PIN == "1234" OR "4321"` is incorrect because the second condition is not a valid comparison – what is being compared to "4321" in order for the condition to be evaluated as True? The correct statement would be `if PIN == "1234" OR PIN == "4321"`.

Section B overview

Section B consists of multiple sub-questions all centred around one scenario (in this case, a music festival). Where algorithms are asked for, candidates must respond using either OCR Exam Reference Language or a high-level programming language that they are familiar with. Where responses are presented using flowcharts or structured English in Section B, no marks are given.

However, it is important for centres and candidates to understand that examiners are not told to penalise responses because they would not work in a particular language. Candidates do not have to state which language they are writing in, and so answers are marked for their logical correctness and consistency. This includes where responses combine features from multiple high-level languages.

Answers are never marked down for use of case.

Question 6 (a) (i)

6 OCR Rockfest is a yearly music festival. Visitors buy tickets to attend and watch the music acts.

A computer system is created to sell tickets and store information about visitors and the music acts.

(a) Visitors are asked to choose a password. The password is entered twice and stored in the variables `passA` and `passB`

(i) Tick (✓) **one** box to identify the correct OCR Exam Reference Language statements for the design of this task.

OCR Exam Reference Language to enter two passwords	Tick (✓) **one** box
`passA and passB = input("enter password twice")`	
`input("enter your password") = passA` `input("enter your password again") = passB`	
`passA = input("enter your password")` `passB = input("enter your password again")`	
`"passA" = input("enter your password")` `"passB" = input("enter your password again")`	

[1]

Pleasingly, the vast majority of candidates were able to identify the correct syntax for code to store data in two distinct variables, despite the distractors being designed around mistakes and misconceptions that have been seen commonly in previous examination series. Centres should be applauded for preparing candidates so well to avoid these common pitfalls.

Question 6 (a) (ii)

(ii) Design an algorithm that:

- checks if the passwords stored in `passA` and `passB` are the same
- outputs "`passwords match`" if they are the same
- outputs "`passwords do not match`" if they are **not** the same.

.....

.....

.....

.....

.....

..... **[3]**

This question was answered exceptionally well by most candidates - including candidates who struggled on other questions. This shows that basic fundamental programming skills are being taught consistently across centres and candidates are able to complete questions such as this to a high standard.

This shows a clear progression from previous years and is extremely pleasing to see. Although the constructs assessed here should be relatively simple, many candidates were not able to design algorithms of this nature historically.

Question 6 (b) (i)

(b) The database table `TblMusic` stores data about the music acts.

Some of the data in `TblMusic` is shown:

ActID	Stage	Day	SetLength
1	Platinum	Saturday	0.5
2	Hills	Saturday	1.5
3	Triangle	Sunday	1.0
4	Platinum	Sunday	0.5

An SQL statement has been written to show `ActID` and `Stage` for all acts playing on Saturday. The SQL statement is incorrect.

```
SELECT ActID AND Stage
FROM TblActs
IF Day = Saturday
```

(i) Refine the SQL statement to correct the errors.

.....

.....

.....

..... **[4]**

This question was surprisingly challenging for many candidates, with very few candidates achieving full marks even when they scored very highly across the rest of the paper.

Most candidates were able to identify the incorrect table name and that **`IF`** should have been **`WHERE`** to be valid SQL. However, far fewer candidates replaced the AND between the field names with a comma or noticed the missing string delimiters (either single or double quotation marks) around **Saturday**.

Where candidates introduced other errors, this was recognised by the mark scheme. A very common error was introducing spaces into field or table names.

Misconception

In SQL, field names are separated by commas and string data has to be delimited using quotation marks. The correct, full mark answer for this question was

```
SELECT ActId, Stage
FROM TblMusic
WHERE Day = "Saturday"
```

SQL is case-insensitive, so capitals **do not** have to be used for keywords in candidate answers but are in examination paper questions by convention.

However, field and table names do not contain spaces, so **`Tbl Music`** is incorrect.

Question 6 (b) (ii)

(ii) Identify **one** example of a record from `TblMusic`

.....

..... **[1]**

This was the lowest scoring question on the entire examination paper. Only a small percentage of candidates were able to give an example of a record as an entire row in the database table, such as `[1, Platinum, Saturday, 0.5]`.

Examiners were instructed to be generous and give this mark wherever four field values were identified, whether they were already in the table or even the correct data type or not. However, even with this generosity, most candidates were not successful.

Assessment for learning

"The use of records to store data" is bullet point 3 in section 2.2.3 of the specification, with the further clarification that "use of 2D arrays to emulate database tables of a collection of fields, and records" is required content to be covered.

Question 6 (b) (iii)

(iii) Give the most appropriate data type for each field shown.

Field name	Data stored	Data type
`Stage`	The name of the stage	
`SetLength`	The length of each set in hours	

[2]

This question required candidates to give the most appropriate data type for each field listed. Where candidates understood the concept of data types, the vast majority were able to correctly identify string and real/float (either was acceptable) as the correct answers.

Iincorrect answers tended to show that candidates did not know what was meant by "data type" and so gave responses such as "variable" or "input".

Data types that exist with common DBMS software such as MySQL or Microsoft Access were also accepted. This meant that (for example), varchar and single/double were accepted as synonyms for string and real/float.

Question 6 (c) (i) and (ii)

(c) Tickets to the festival are £60 each. Visitors can buy multiple tickets. For example, a visitor buying 4 tickets would pay £240.

(i) Write a function, `calculatePrice()` that:

- takes the number of tickets to be purchased as a parameter
- calculates and returns the total price to pay.

You must use either:

- OCR Exam Reference Language, or
- a high-level programming language that you have studied.

.....

.....

.....

.....

.....

..... **[4]**

(ii) Write an algorithm that:

- uses the function `calculatePrice()` to find the price for a visitor buying 3 tickets
- stores the result in the variable `totalPrice`

You must use either:

- OCR Exam Reference Language, or
- a high-level programming language that you have studied.

.....

.....

.....

.....

.....

..... **[2]**

These questions proved to be excellent discriminators, with only candidates who achieved very highly overall able to do well. Although teaching of programming by centres has improved students' general overall responses, it is clear that defining functions and then utilising them is an area that still requires focus.

For part (c) (ii), candidates were asked to define a function. Many candidates were able to begin to define a function. However, a common misconception was either the lack of a parameter in the definition, or overwriting/ignoring the parameter and asking for separate input within the function.

Another very common misconception was to print or otherwise output the calculated price instead of returning it as required.

For part c (iii), candidates were asked to then use this function to calculate the price of 3 tickets and store the answer in specified variable. Very few candidates did this successfully. Instead many candidates attempted to redefine the function again or to work out the price of the tickets in some other way.

Misconception

When a function is defined, all data passed to the function should be done via parameters. It is not appropriate to ask for input inside the function, especially when this overwrites or replaces an existing parameter.

Functions also return a value instead of printing anything, as clearly requested in the question.

Exemplar 2

```
def calculatePrice(tickets):
    tickets = int(input("Enter number of tickets:"))
    price = tickets * 60
    print("The total price is:£", price)
```

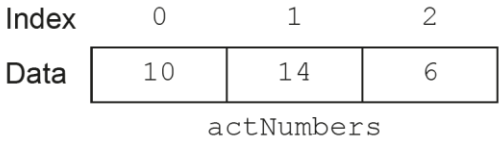
In this example, the candidate has successfully defined a function with a single parameter, tickets. This first line would have achieved mark points 1 and 2. However, the tickets' parameter is then overwritten with an unnecessary input statement.

The price is then correctly calculated but is printed instead of returned. As a result, this example receives 2 out of 4 marks.

Question 6 (d) (i)

(d) The number of different music acts on each of the three stages is stored in the array `actNumbers`.

Each index of the array represents one stage.



This algorithm adds up the total number of acts on all three stages.

```
01  count = 0
02  for x = 0 to 2
03      count = count + actNumbers[x]
04  next x
05  print(count)
```

(i) Complete the trace table for the algorithm. You may not need to use all rows in the table.

Line number	count	x	Output

[4]

Trace tables are commonplace in Paper 2 and have examined before. Candidates showed better understanding of tracing a program through line by line, and this confidence was clear in candidate responses, with many able to correctly do this.

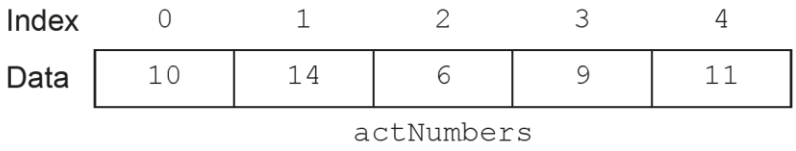
This question differed slightly from historical examination questions in two ways.

Firstly, this is the first time that a for loop has been asked about in a trace table. The challenge of the counter variable being updated was handled with ease by most candidates.

Secondly, although line numbers were asked for, these were not used in the marking of candidates' responses and instead one mark was given per set of changes to variables. This process worked extremely well and will be used in the marking of any future trace table questions.

Question 6 (d) (ii)

(ii) The array `actNumbers` is changed to include data for two more stages. The new array is shown:



The algorithm from part **(i)** needs refining to work for the new array. The algorithm is repeated here:

```
01  count = 0

02  for x = 0 to 2

03      count = count + actNumbers[x]

04  next x

05  print(count)
```

Identify the line number that will need to be changed for the new array and write the updated line of code.

Line number .....

Updated code .....

.....

[2]

Candidates were able to confidently identify line 02 as the line to be changed and also identified the upper bound of the for loop as the value to be changed. Where candidates provided other solutions that logically worked, these were of course accepted.

Question 6 (d) (iii)

(iii) Refine line 05 in the algorithm to output `"The total act count is "` and then the content of the variable `count`

For example:

```
The total act count is 50
```

New line 05 .....

..... **[1]**

As with many other smaller programming questions on this paper, candidates performed exceptionally well. The majority of candidates were able to use the **`count`** variable to output the correct value. Where this was not successfully done, candidates tended to include the variable identifier inside the string delimiters, thereby outputting the word count and not the value held in the variable.

Examiners were instructed to credit responses that used string interpolation, such as f strings in Python or the use of $ before strings in PHP.

Question 6 (e)

(e) There are 500 tickets available for the music festival. The number of tickets currently available is stored in the variable `tickets`

Complete the algorithm to:

- prompt the user to enter how many tickets they would like and take this value as input
- output "`Tickets booked`" and reduce the value stored in `tickets` if there are enough tickets available
- output "`Not enough tickets`" if there are not enough tickets available
- repeat the steps until there are no tickets left.

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

```
tickets = 500
```

.....

.....

.....

.....

.....

..... **[6]**

This year, candidates appeared much more willing to attempt the question and therefore many received marks for their efforts.

Very few candidates left their response completely blank. This is pleasing to see, as usually we see an increase in the number of candidates who do not attempt this question at all.

Marks were allocated on the mark scheme for the correct input from the user and any attempt at iteration, with most candidates achieving at least the first of these.

Many candidates made a very strong attempt at checking for tickets available. Where this was done correctly, it included the scenario where exactly the available number of tickets were requested, (e.g. 7 tickets remaining and 7 tickets were requested). If, however, a candidate checked **`if tickets > num`** (where num is the number requested), this was not correct as it would not allow this exact scenario.

Other common mistakes included:

- checking if tickets was less than 500 instead of checking against the number requested
- subtracting the number of tickets required even when the booking was not successful.

Altogether however, this was an well answered question with most candidates achieving some marks and many achieving all or almost all available marks.

Exemplar 3

```
tickets = 500
while tickets != 0:
    TNum = int(input('how many tickets'))
    if TNum <= tickets:
        print('tickets booked')
        tickets = tickets - TNum
    else:
        print('Not enough tickets')
endwhile
```

In this example, the candidate successfully achieves all mark points for a 6 mark response.

- The number of tickets requested is entered and stored in the variable **`TNum`**.
- This value is then checked against the **`tickets`** variable, using <= to make sure that there are enough tickets remaining.
- If this check was successful, a message is printed and the value in tickets reduced correctly.
- If the check was not successful then a "not enough tickets" message is printed instead.
- All of this is repeated while the number of **`tickets`** does not equal 0.

This was a typical response that achieved full marks.

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

We've made it easier for Exams Officers to download copies of your candidates' completed papers or 'scripts'. 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 professional development 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 exam paper builder platform for a range of our qualifications. Find out more.

Free for all OCR centres with an Interchange account, it allows unlimited users per centre. We require an Interchange username to verify your centre's users for ExamBuilder.

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

Once you have an Interchange username, use the form on this page to sign up for ExamBuilder, or ask an existing user at your centre to create an ExamBuilder account for you.

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 (examined units only).

Find out more.

Tell us what you think

Your feedback plays an important role in how we develop, market, support and resource qualifications now and into the future. We want you and your students to enjoy and get the best out of our qualifications and resources, but to do that we need your honest opinions to tell us whether we're on the right track or not.

You can email your thoughts to resources.feedback@ocr.org.uk or visit our feedback page to learn more about how you can help us improve our resources.

Designing and testing in collaboration with you and your students

Helping young people develop an ethical view of the world

Equality, diversity, inclusion and belonging (EDIB) are part of everything we do

---

Contact the team at:

01223 553998

ocr.org.uk

To stay up to date with all the relevant news about our qualifications, register for email updates at ocr.org.uk/updates

Visit our Online Support Centre at support.ocr.org.uk

CAMBRIDGE
UNIVERSITY PRESS & ASSESSMENT