Qualification Accredited

Oxford Cambridge and RSA

GCSE (9-1)

Examiners' report

COMPUTER SCIENCE

J276
For first teaching in 2016

J276/02 Summer 2018 series

Version 1

www.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. 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 can be downloaded from OCR.

Paper J276/02 series overview

J276/02 (Computational thinking, algorithms and programming) is one of two examinations components for the GCSE Computer Science. This component focuses on:

- algorithms
- programming techniques
- producing robust programs
- computational logic
- translators and facilities of languages
- data representation.

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.

Where candidates had extensive practice of producing and completing algorithms using a high-level language in classroom situations, this clearly allowed them to answer questions on this paper more confidently. The mandatory NEA (programming project) task is one chance for all candidates to spend a significant amount of time practising their programming skills. It is a requirement that all centres allocate 20 hours of timetabled time for this to be completed by candidates.

Centres must be aware of the need to cover the whole of the specification content for this component; for example, 'the use of SQL to search for data' is listed under section 2.2. Programming techniques. Not all content can possibly be covered in every examination but questions can be asked on any of the topics listed.

Centres are also encouraged to be aware of the appendices at the rear of the specification, particularly section **5f** which shows the format that pseudocode and Boolean logic will be presented in examinations. Candidates' answers will be accepted in any logical form; they are not required to present answers in any particular form. However, candidates should be aware of these conventions in order for them to successfully understand and access examination questions.

Candidate performance overview

Candidates who did well on this paper generally did the following:

- understood the use of SQL and wildcards in questions 1b(i) and 1b(ii)
- successfully followed an algorithm through in questions 2a(i) 2a(ii) and 6(a)
- demonstrated an understanding of creating or completing algorithms in questions 4a(i), 4a(ii) 7a(i) and 8.

Candidates who did less well on this paper generally did the following:

- demonstrated a lack of understanding of key terminology such as data types, programming constructs, variables, functions or count-controlled loops
- showed poor understanding of the use of functions in question 4a(ii), particularly around successfully utilising a pre-existing function
- were unclear in their demonstration of the steps taken by sorting algorithms in question 4c(i)
- had either not had enough opportunity to practise programming using a high-level language in a classroom situation or had not grasped how these skills transferred across to the questions on this examination paper.

Question 1 (a)

1 OCR High School uses a computer system to store data about students' conduct. The system records good conduct as a positive number and poor conduct as a negative number. A TRUE or FALSE value is also used to record whether or not a letter has been sent home about each incident.

An example of the data held in this system is shown below in Fig. 1:

StudentName	Detail	Points	LetterSent
Kirstie	Homework forgotten	−2	FALSE
Byron	Good effort in class	1	TRUE
Grahame	100% in a test	2	FALSE
Marian	Bullying	−3	TRUE

Fig. 1

(a) State the most appropriate data type used to store each of the following items of data.

- StudentName ....
- Points ....
- LetterSent ....

[3]

This question was answered very well by the majority of candidates, with most achieving full marks. Where candidates did fail to do well, it tended to be through a misunderstanding of the term 'data type', leading the answers such as 'Kirstie' or -3.

Question 1 (b) (i)

(b) The data shown above in Fig. 1 is stored in a database table called **Conduct**.

(i) Write an SQL statement to select the StudentName field for all records that have negative Points.

....

....

.... **[3]**

Despite 'the use of SQL to search for data' being listed under section 2.2 of the specification, the vast majority of candidate answers showed a lack of understanding of SQL in general. It was common to see incorrect answers that attempted to use pseudocode to respond to this and a large number leaving this question blank. Where candidates did use the keywords of SELECT, FROM and WHERE answers were much more successful. The SQL keywords and terms that candidates are required to know are listed in appendix 5f of the specification.

Question 1 (b) (ii)

(ii) State the wildcard that can be used in SQL to show all fields from a table.

....

.... **[1]**

As with the previous question, where candidates were familiar with SQL and understood the term 'wildcard', this question was answered successfully. However, the majority of candidates did not appear to be confident with either of these and so gave answers more suited to algorithm questions. ⓘ

Candidates should also be encouraged to provide answers which match the keyword in the question; in this case 'state' means to give a specific name, value or other brief answer without explanation. The wildcard in question (an asterisk / *) was sometimes surrounded by other SQL code and so it was impossible to tell whether the candidate understood which part of this was the wildcard.

A small number of candidates incorrectly identified '%' as the wildcard; although this would be a valid wildcard in the WHERE clause, it is not used to select all fields from a table.

ⓘ **SQL coverage** Appendix 5f of the specification lists all of the SQL keywords and wildcards that candidates are required to understand. This is available from http://www.ocr.org.uk

Question 1 (c)

(c) A single record from this database table is read into a program that uses an array with the identifier `studentdata`. An example of this array is shown below:

```
studentdata = ["Kirstie", "Homework forgotten", "-2", "FALSE"]
```

The array is zero based, so `studentdata[0]` holds the value "Kirstie".

Write an algorithm that will identify whether the data in the `studentdata` array shows that a letter has been sent home or not for the student. The algorithm should then output either "sent" (if a letter has been sent) or "not sent" (if a letter has not been sent).

....

....

....

....

....

....

....

....

....

....

.... **[4]**

This question was generally well answered, at least in part, by candidates of all abilities. The use of selection (IF) is obviously well understood and this, together with the linked multiple outcomes of outputting 'sent' or 'not sent' were the most commonly credited mark points. However, despite the question providing very specific guidance on how to access the array, many candidates did not even attempt to do this.

Where the array was accessed correctly, a common mistake was to access the wrong element, usually index 4 rather than index 3. Arrays in questions on this specification will always be zero based and this fact was also given in the question by means of an example. Candidates should be encouraged to read all parts of a question before beginning to attempt it.

Exemplar 1

```
BEGIN ✓ ✓ ✓
 IF studentdata[3] = "True" THEN
          PRINT("Sent") ✓
ELIF studentdata[3] = "False" THEN
          PRINT("not sent")
ENDIF
END
```

[4]

In the above exemplar response, the candidate has achieved full marks by accessing the correct element in the array and using this to decide if the letter should be sent.

Exemplar 2

```
INPUT StudentName
INPUT Detail
INPUT Points
INPUT LetterSent
    IF ✓ LetterSent = TRUE THEN
  OUTPUT "Sent" ✓
        ELSE
  OUTPUT "Not Sent"
   END IF
```

[4]

In the above exemplar response, the candidate has not attempted to access the array at all, instead asking the user to input a value; this does not do what is required and so is only given 2 of the 4 marks (for a valid use of selection and outputting a message correctly).

Question 2 (a) (i) and (ii)

2 A programmer has written an algorithm to output a series of numbers. The algorithm is shown below:

```
01  for k = 1 to 3
02      for p = 1 to 5
03          print (k + p)
04      next p
05  next k
06  m = 7
07  print m * m
```

(a) **(i)** Give the first **three** numbers that will be printed by this algorithm.

.... **[1]**

The majority of candidates seemed to understand the use of count-controlled loops to repeat sections of an algorithm, which was pleasing. However, the vast majority were then unable to apply this to nested loops, understanding that the inner loop completes fully for each iteration of the outer loop. The correct answer for a(i) of 2, 3, 4 (with k remaining as 1 but p being 1 then 2 then 3) was only seen in candidates who generally achieved highly overall on the paper. Question part a(ii) tests the same understanding and so approximately the same number of candidates understood that the given lines would repeat 15 times in total.

Question 2 (b)

(b) Identify **two** basic programming constructs that have been used in this algorithm.

1 ....

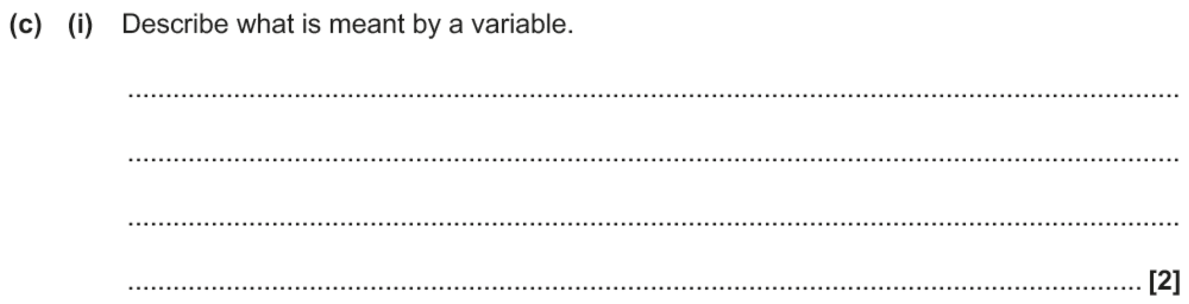
....

2 ....

.... **[2]**

Programming constructs are defined in section 2.2 ('Programming techniques') of the specification. These are given as sequence, selection and iteration. In the algorithm given in the question, sequence and iteration were both used and so candidates were credited if they gave either of these two answers, with loops being accepted as an alternative to iteration. Centres are encouraged to ensure that all of the specification is covered with candidates and that keywords from the specification are highlighted in lessons as appropriate.

Question 2 (c) (i)



(c) **(i)** Describe what is meant by a variable.

....

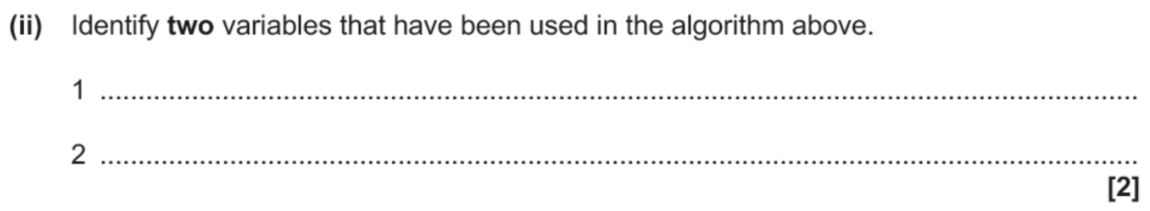
....

....

.... **[2]**

This question was pleasingly answered relatively well, perhaps because of similar questions regarding variables and constants in question on the previous GCSE Computing specification. It was pleasing how many candidates were able to relate variables to an identifier given to a memory location, showing a good understanding of a basic Computer Science concept. A mark was given quite generously for the ability for a variable to be 'changed', but candidates should be aware that ideally their answer should make reference to the value stored being able to be changed, not the variable itself changing somehow.

Question 2 (c) (ii)



(ii) Identify **two** variables that have been used in the algorithm above.

1 ....

2 ....

[2]

There were actually three variables in use (k, p and m) and so any two of these were able to be given to achieve full marks. This question was answered very well by the majority of candidates, with most achieving 2 marks. A common incorrect answer was to give full lines of pseudocode as the variable, e.g. 'm = 7' or 'next k'. In this case it was impossible to decide whether the candidate understood which part of their response was the variable and so no mark was given. As with question 1b(ii), candidates should be encouraged to look at the command word in the question (in this case 'identify') and make sure that their response is suitable.

Question 3 (a) (i) and 3 (a) (ii)

3 The logic diagram below (Fig. 2) shows a system made up of two connected logic gates.

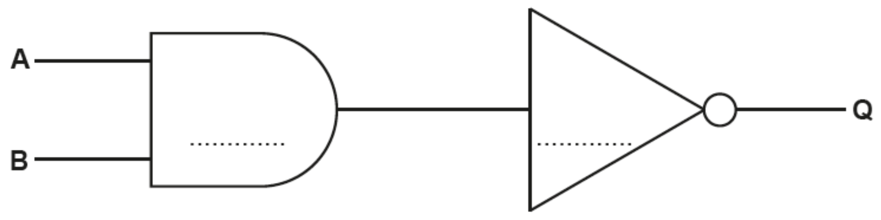


Fig. 2

(a) **(i)** Label the names of the two gates on the diagram above. **[2]**

Candidates and centres are obviously extremely comfortable with the use and understanding of logic gate symbols. The vast majority of responses on part (i) achieved 2 marks. The most common incorrect answer was to mix up AND and OR for the first gate. For part (ii), candidates were able to apply their knowledge successfully with the majority of all candidates achieving 4 marks. This topic has been covered successfully by centres and that this work is reflected in examination responses.

Question 3 (b)

(b) Draw the logic diagram represented by **Q = A ∨ ¬B**

[2]

Boolean symbols and notations that may be used in examinations are shown in appendix 5f of the specification. Candidates will never be expected to use these symbols in their own responses (AND, OR and NOT are perfectly acceptable) but they must be aware of how questions on papers may be written.

For this question, it was obvious that some groups of candidates did not understand the symbols in the question and so were not able to provide a suitable response. However, it was pleasing to see that this was not at all the majority and that large numbers of candidates were able to not only understand the question but also provide a correct or partially correct response.

Where responses were incorrect, a common mistake was to utilise the NOT gate after the output from the OR gate when in fact the NOT gate should be applied to the input from B before this feeds into the OR gate. Another common mistake was to forget to label up inputs as A and B, therefore not allowing the examiner to award the mark for the NOT gate on the correct input.

Question 4 (a) (i)

4 A library gives each book a code made from the first three letters of the book title in upper case, followed by the last two digits of the year the book was published.

For example, "Poetry from the War", published in 2012 would be given the code `POE12`.

(a) **(i)** Complete the following pseudocode for a function definition that will take in the book title and year as parameters and return the book code.

```
01  function librarycode(title, ....................................................)

02      parta = title.subString(0, ....................................................)

03      partb = year.subString(2, 2)

04      .................................................... parta.upper + partb

05  endfunction
```

[3]

Responses to this question were mixed. The majority of candidates were able to decide on the correct passing of `year` as a second parameter; this had to be specifically `year` and nothing else as this identifier was referred to later in the algorithm.

Fewer candidates were able to correctly decide that three characters were required from the title; this was often not completed as expected as `subString` is listed in appendix 5f of the specification as a string handling tool that could be used in the examination and a full example of its use is also given on line 03.

Even fewer candidates understood that a function must return a value, this being the answer to line 04. The most common incorrect answer here was candidates attempting to print / output the book code rather than return it. A mark was given if they assigned the return value to the name of the function (e.g. `librarycode = parta.upper + partb`) as this is a valid method of returning a value in some high-level languages such as Visual Basic.

Question 4 (a) (ii)

(ii) Use pseudocode to write an algorithm that does the following :

- Inputs the title and year of a book from the user.
- Uses the librarycode function above to work out the book code.
- Permanently stores the new book code to the text file `bookcodes.txt`

....

....

....

....

....

....

....

....

....

....

....

....

.... **[6]**

This question was answered extremely poorly by candidates and shows a lack of understanding of the use of functions (and subroutines in general). Where candidates scored highly here, it was a very good indicator that they would score highly across the paper as a whole.

The majority of candidates were able to access the first mark for inputting the two requested values. However, a significant number did not make it clear that the value inputted had to be stored somewhere* (typically in a variable). Therefore although `x = input('enter a title')` was OK, simply stating `input('enter a title')`was too vague to achieve the mark. Another common misunderstanding was that both values could be input at the same time; a variable will only ever store one value and so asking for both in one go is unlikely to achieve the marks.

* Where higher ability candidates showed an understanding of how to do this differently, e.g. using the input value as the argument to pass into the function, this was of course credited by examiners.

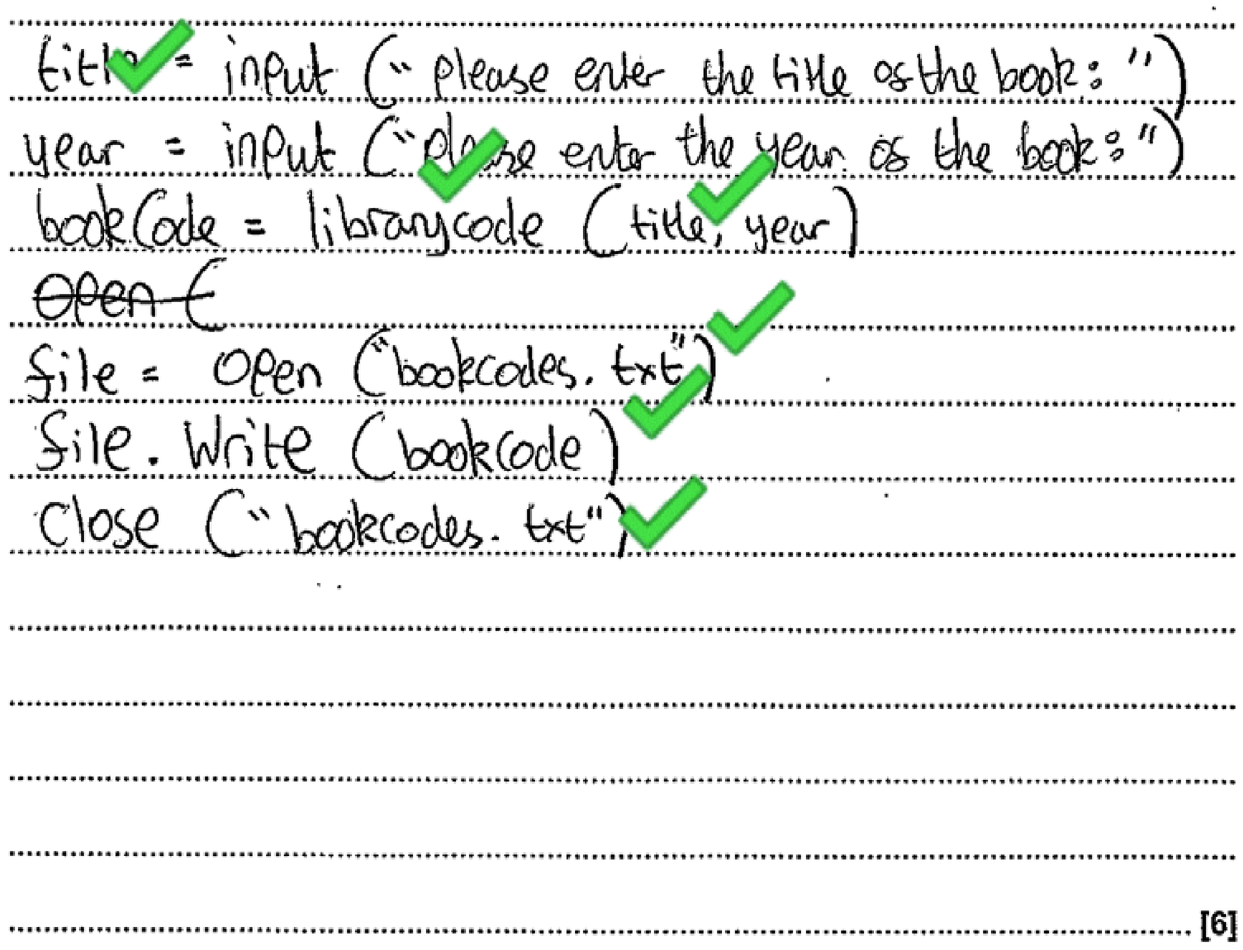
However, the bigger misconception was around the use of pre-existing functions. One of the benefits of subroutines is that code can be modularised and re-used without requiring programmers to copy and paste code if they require it multiple times. Perhaps the most common student answer for this question was to write out the code from the function again to try to calculate the book code. This was not what was required from the question and shows a fundamental lack of understanding of the use of functions.

Marks were given for calling the existing function, passing in the values previously input and then writing the returned code to the text file. The vast majority of candidates achieved none of these marks.

Other independent marks were given for opening and closing the text file and a pleasing number of candidates were able to do this successfully.

Misconception Functions modularise code; they can be written once and used multiple times in a program. If a pre-written function is given as part of a question and candidates are told to use it, they will NOT be credited for simply copying and pasting the code inside the function to use; this shows a lack of understanding.

Exemplar 3

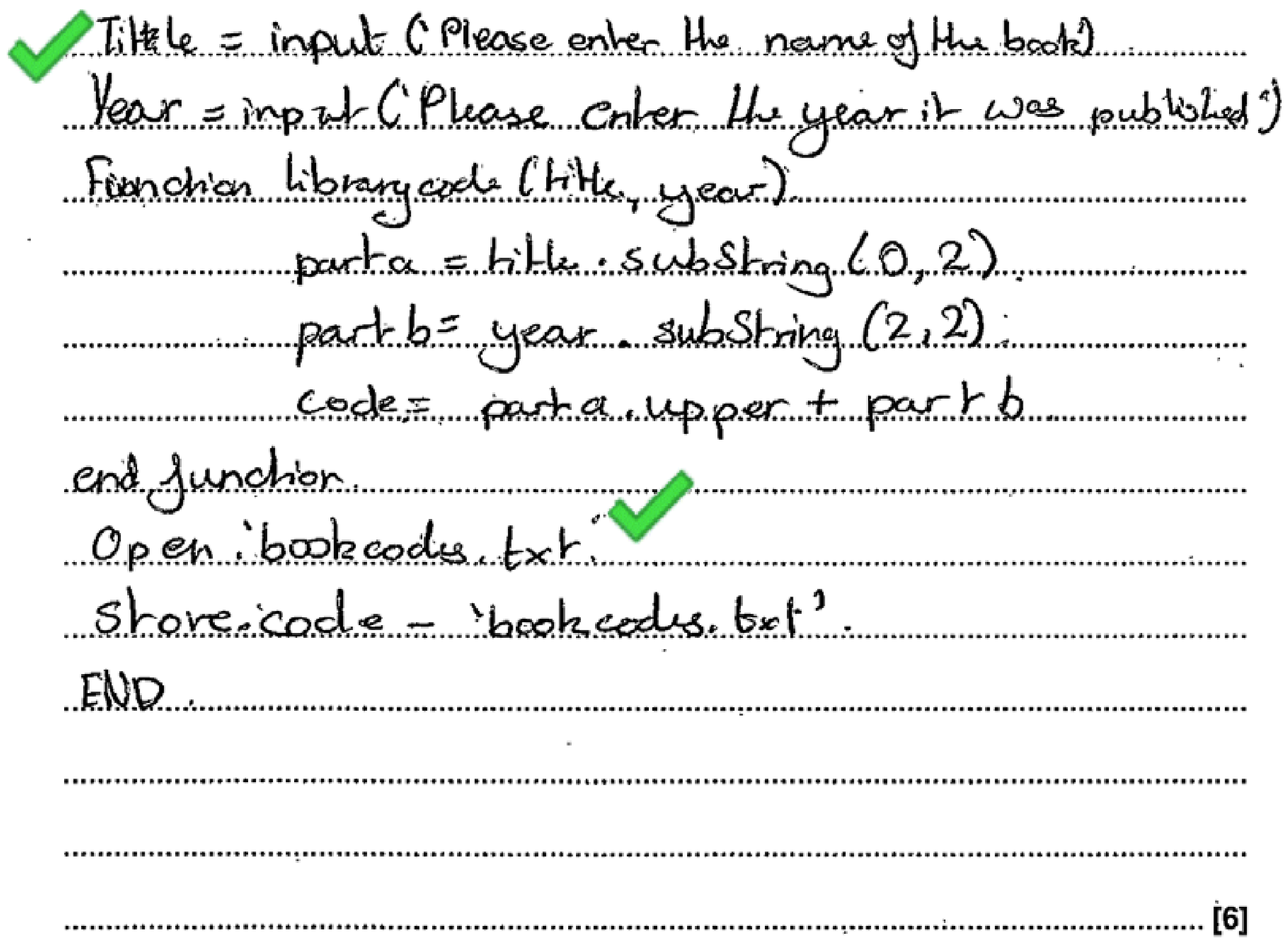


```
title = input ("please enter the title of the book: ")
year = input ("please enter the year of the book: ")
bookCode = librarycode (title, year)
~~open (~~
file = open ("bookcodes.txt")
file.write (bookCode)
close ("bookcodes.txt")
```

[6]

In the exemplar above, the candidate has successfully input two values from the user. They have then called the pre-existing function librarycode() with the values previously input as arguments (in parenthesis). This will return a value which is stored in the variable bookCode. The correct text file is then opened, the returned value is written to the text file and then the file is closed. This response covered everything required from the question and so is credited with full marks [6 out of 6].

Exemplar 4



```
✓ Tittle = input ('Please enter the name of the book')
Year = input ('Please enter the year it was published')
Function librarycode (title, year)
        parta = title . subString (0, 2)
        partb = year . subString (2, 2)
        code = parta . upper + partb
end function
Open 'bookcodes.txt' ✓
Store code - 'bookcodes. txt'.
END.
                                                        [6]
```

In the exemplar above, the candidate has successfully input two values from the user. However, they then misunderstand the purpose of a function and instead of calling this, re-write out the function code. Note that there is no call to this function, which could have made it correct; the assumption is that the candidate thinks they must place the code here for it to run, which is incorrect.

The candidate does gain a mark from opening the text file but they cannot get the mark for writing to it (as no book code was ever calculated) and they do not close the file. This response gained two out of six marks and is typical of the average candidate answer.

Question 4 (b) (ii)

(ii) Describe **two** benefits to a programmer of using sub programs.

....

....

....

....

....

....

....

....

....

.... **[4]**

Although candidates had struggled on the previous questions to demonstrate that they could use functions, a much larger number were able to describe why subprograms such as functions are beneficial. This demonstrates that candidates do understand this *in theory* but may not perhaps have had the practical programming time in lessons to apply this on screen. For example, a common correct answer frequently given was that code did not have to be copied and pasted multiple times but could instead be called from other parts of the program, but candidates then did exactly the opposite of this when asked to use a function themselves.

A good number of candidates were also able to describe other potential benefits, such as making the code easier to maintain and debug or allowing work to split up between programmers of different skill sets.

Centres should be aware of the command word given in questions and match this up to the number of marks allocated; in this case, 'describe' two benefits for 4 marks means that short single sentence answers area unlikely to hit sufficient points on the mark scheme to be given full marks. The command words that will be used consistently in examinations are shown in appendix 5e of the specification.

Question 4 (c) (i)

(c) The library sorts their books based on the book code.

(i) Show the steps that a merge sort would take to put the following list of book codes into ascending alphabetical order (from A to Z).

`POE12 , BAC97 , FLY77 , JAV16 , TAL86 , AND18 , ZAR09 , HOP86`

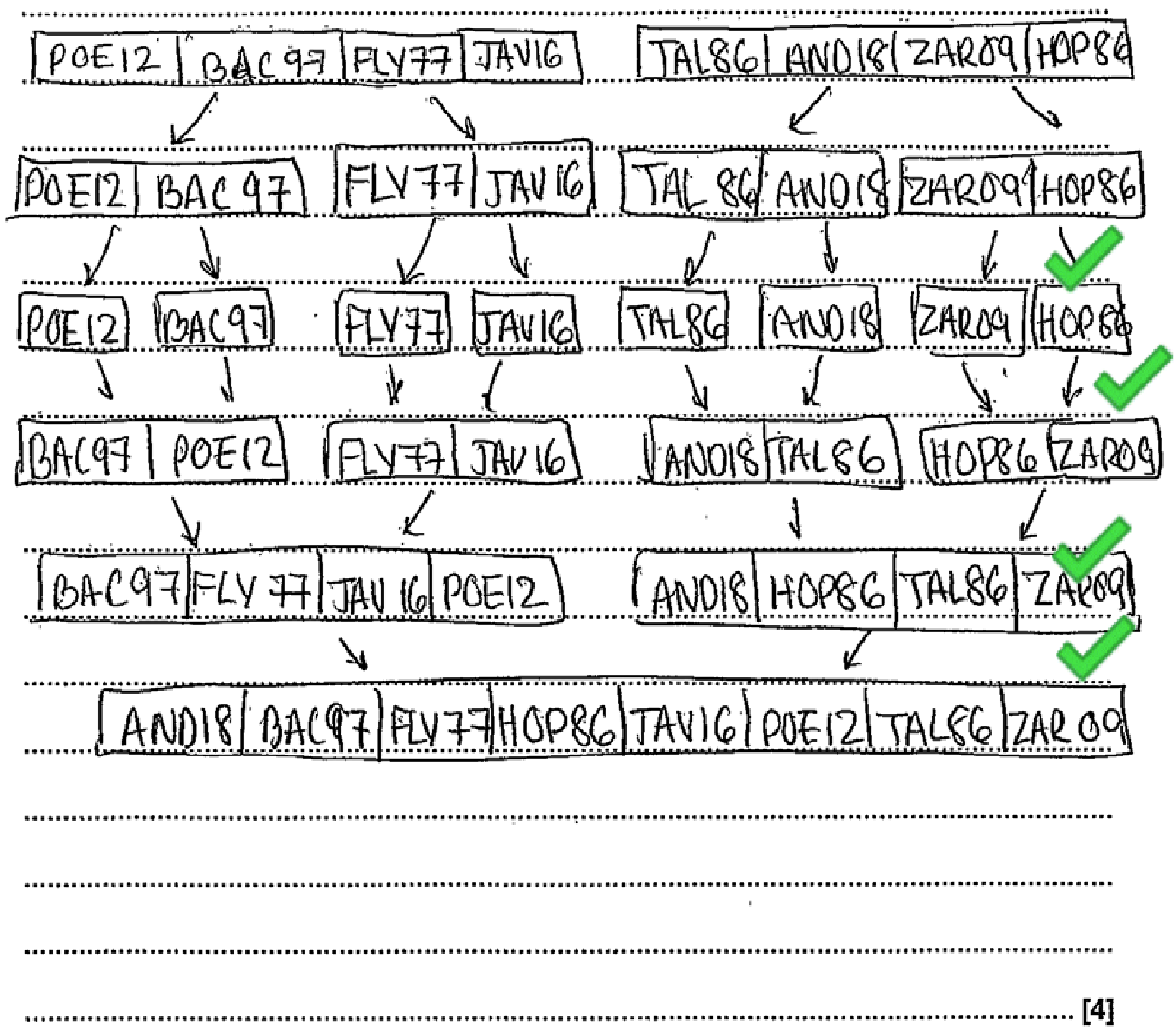
.... **[4]**

Large numbers of candidates understood the divide and conquer strategy applied by the merge sort algorithm; the list is continually divided in half until lists of size 1 are achieved before the lists are then merged together to achieve the sort, with the size of lists doubling with every iteration (i.e. when two lists of size 2 are merged, the result is a list of size 4). Many candidates achieved good marks on this question.

Where mistakes were made, they generally fell into one of three categories. One was applying a different algorithm than had been asked (e.g. showing a bubble sort). Another was misunderstanding where the sorting takes place; some candidates showed the lists being split up correctly, but then these being merged and the list sorted in place afterwards – this is incorrect, it is the act of merging that sorts the values. A third and more common issue was that examiners found it extremely tough to decide where lists were split up or merged from candidate responses that seemed to list all values in one row with no seeming differentiation between one list of eight values and eight lists of one value. Where examiners were not able to see this, marks could not be given.

Misconception When using merge sort to merge together [1, 3] and [2, 4] into ascending order, the algorithm will take the lowest value from the front of either list (in this case 1) and place it into the new, merged list. This will then repeat meaning that the new list will be [1, 2, 3, 4]. It does **not** merge the lists to be [1, 3, 2, 4] and then sort this list.

Exemplar 5


POE12 | BAC97 | FLY77 | JAV16 | TAL86 | AND18 | ZAR09 | HOP86
POE12 | BAC97 | FLY77 | JAV16 | TAL86 | AND18 | ZAR09 | HOP86
POE12 | BAC97 | FLY77 | JAV16 | TAL86 | AND18 | ZAR09 | HOP86
BAC97 | POE12 | FLY77 | JAV16 | AND18 | TAL86 | HOP86 | ZAR09
BAC97 | FLY77 | JAV16 | POE12 | AND18 | HOP86 | TAL86 | ZAR09
AND18 | BAC97 | FLY77 | HOP86 | JAV16 | POE12 | TAL86 | ZAR09
[4]

The exemplar above shows an ideal way to respond to this question. The list is obviously and clearly split up in successive passes into lists of a single value. Each pair of lists is then merged, with the process of merging resulting in sorted lists. Each time the lists are merged, the result is a larger list in ascending order. The candidate response shown here not only shows this process but makes it very clear that the values are held in separate lists and even includes arrows to illustrate the process of merging. This achieved full marks [4 out of 4].

Question 5 (a) (i)

5 **(a)** **(i)** Convert the denary number **132** into an 8 bit binary number.

....

....

....

.... **[2]**

This question was answered correctly by the vast majority of candidates. Pleasingly, conversion of numbers to and from binary is now obviously a comfortable skill for candidates of all levels.

Question 5 (a) (ii)

(ii) Convert the binary number **10110101** to its hexadecimal equivalent.

....

....

....

.... **[2]**

Slightly fewer candidates were able to answer this question successfully compared to 5(a)(i). Most were able to split the binary number up into two nibbles, but then the conversion to binary for each nibble sometimes was incorrectly completed. Common wrong answers included 11 5 (which achieved 1 mark for 5 but did not recognise that 11 in denary equates to B in hexadecimal) or C5, where a mistake was made once the hexadecimal value went over 9. Very few answers showed a complete lack of understanding, but where this was seen, candidates tended to simply convert the binary to denary and ignore the requirement to use hexadecimal. This achieved no marks.

Question 5 (a) (iii) and (iv)

(iv) Describe a shift that can be used to double the value of the binary number **00100100**.

....

....

....

.... **[2]**

Candidates showed a good understanding of binary shifts, which is especially pleasing as this is a new point that was not covered in the old GCSE Computing specification. The majority of candidates were able to both carry out a shift and describe a shift that matched a give outcome. One common mistake was for candidates to describe the direction of a shift but not say how many places to shift (e.g. 'shift left' but missing 'by one place').

Question 5 (b) (i)

(b) The table below (Fig. 3) shows the ASCII codes for a number of characters.

The lower case ASCII code for a character can be found by adding **0100000** to the upper case version.

Character	ASCII code
R	1 0 1 0 0 1 0
r	1 1 1 0 0 1 0
A	1 0 0 0 0 0 1
a	
E	1 0 0 0 1 0 1
e	

Fig. 3

(i) Complete the table above by filling in the missing ASCII codes. **[2]**

This question was relatively simple in what was being asked (the addition only affected one bit per answer) but the context in which it was presented was new. It was pleasing that the majority of candidates were able to extract the question from this and present sensible, and in most cases correct answers.

A common mistake was to end up with an 8 bit ASCII code despite the original code only having 7 bits. This is perhaps more due to time constraints of the examination rather than any lack of understanding but does illustrate the importance of checking answers.

Another common issue on this question was a small number of candidates writing a 0 but then overwriting it with a 1 (or vice versa), leaving both visible; in these circumstances it is impossible for examiners to know which value was intended as the answer and so it cannot be marked as correct. Candidates should be encouraged to clearly and neatly cross out and replace their answer if they make a mistake.

Question 5 (b) (ii)

(ii) Compare the use of ASCII and Extended ASCII to represent characters.

....

....

....

....

....

.... **[2]**

The question here asked for a comparison and the idea of discussing a point from both sides was required to achieve the marks. For example, stating that 'ASCII uses 7 bits' is not a valid comparison, but stating that 'ASCII uses 7 bits whereas Extended ASCII uses 8 bits' or even just 'Extended ASCII uses more bits that ASCII' are both valid comparisons and would achieve a mark.

A large number of candidates understood the use of the character sets and also the technical details behind them. Examiners were instructed to be generous in terms of the exact technical details allowed (e.g. although 7 bits allows for 127 different characters, values approximating this would be accepted) although this was expected to be sensible.

Question 6 (a)

6 An infinite loop is where a section of a program repeats indefinitely.

(a) For each of the pseudocode algorithms shown below, tick the appropriate box to show whether they will loop infinitely or not.

Pseudocode	Will loop infinitely	Will <u>not</u> loop infinitely

`x = 0`
`while True`
`    print x`
`endwhile`		

`x = 0`
`while x < 10`
`    print x`
`endwhile`		

`x = 0`
`while x < 10`
`    print x`
`    x = x + 1`
`endwhile`		

`y = 5`
`for x = 1 to y`
`    print x`
`next`		

[4]

It was thought that candidates of all abilities would be able to attempt this question, with iteration possibly a relatively simple topic. Many candidates showed a lack of understanding on this question, with far fewer receiving full marks than perhaps was expected. Again, perhaps candidates have had fewer opportunities to experience programming on screen than would be expected. Understanding of basic programming constructs and their application is certainly something which centres would do well to focus on.

Question 6 (b)

(b) Using pseudocode, write an algorithm that will use a count-controlled loop to print out the numbers 1 to 10 in ascending order.

....

....

....

....

....

.... **[3]**

Very few candidates achieved full marks on this question, mainly due to a lack of understanding of what a count-controlled loop is – this is defined by example in appendix 5f of the specification. Many candidates used a condition controlled (WHILE) loop and although the condition was set to be a comparison of a variable which was incremented on every iteration, this was still condition controlled and therefore did not achieve the first mark. However, where this met the requirement of the question, the other marks were achievable.

A FOR loop is truly count-controlled and so was able to achieve full marks. However, some candidates then went on to manually increment the counter variable inside the FOR loop which is at best not necessary and at worst harmful to the correct operation of the loop. These candidates did not therefore achieve full marks.

Examiners were instructed to be generous this time where answers were slightly ambiguous because of the way that certain high-level languages (such as Python) handle count-controlled iteration. Centres should be aware that Python is only one of many suitable high-level languages and that it is perhaps best practice to introduce candidates to overall concepts before looking at how these are implemented in the centre's language of choice. Where possible, if candidates were able to use multiple high-level languages during the course of their learning, this would be extremely beneficial.

Question 7 (a) (i)

7 Victoria is writing a program using a high level language to display the meaning of computer science acronyms that are entered. The code for her first attempt at this program is shown below.

```
01  a = input("Enter an acronym")
02  if a == "LAN" then
03        print("Local Area Network")
04  elseif a == "WAN" then
05        print("Wide Area Network")
06  ........................................................................
07  ........................................................................
08  endif
```

(a) (i) Complete the code above to print out an "unknown" message if any other acronym is entered by the user. **[2]**

Most candidates were able to complete line 07 successfully, with an output/print of an acceptable 'unknown'' message being all that was required. Line 06 required candidates to understand that the message on line 07 should be printed out where the conditions in lines 02 and 04 were both false; the simplest way of achieving this was an ELSE (or equivalent). Where candidates put more logically complex statements, these were successful if they were logically correct. However, many candidates struggled with this.

Misconception To compare the contents of a variable against two possible values, it is incorrect to use :

- `if a!='WAN' or 'LAN'`

In the example above, the comparison of a against 'WAN' is logically correct, but it is unclear what 'LAN' is being compared against.

A logically correct way to achieve the same thing would be :

- `if a!='WAN' and a!='LAN'`

Question 7 (a) (ii)

(ii) Describe what is meant by a "high level language".

....

....

.... **[2]**

This question was answered well by many candidates, with responses relating to the use of English-like keywords but needing to be translated before the processor could execute it being the most popular. A minority of candidates confused 'high-level' with 'difficult' and gave incorrect answers regarding it being too hard to programmers to use. The opposite is in fact true, with 'high-level' referring to the level of abstraction away from the underlying hardware.

Very few answers were seen that in any way discussed this abstraction or portability between different processors. Hopefully centres will become more confident with the delivery of the subject and so more complex and technically complete answers will be seen across all questions from high ability candidates.

Question 7 (b)

(b) Victoria creates her program using an Integrated Development Environment (IDE).

Describe two tools or facilities that an IDE commonly provides.

....

....

....

....

....

.... **[4]**

Most candidates were able to gain at least some marks from this question, with a significant number able to access all 4 marks. A list of common tools and facilities available in an IDE is given in the specification (section 2.5 'Translators and facilities of languages') although other sensible tools that candidates may have had experience with were also accepted. Although many candidates were able to name the tools, they often struggled to describe these and give more comprehensive answers. Again, extensive practical use of a high-level language programming language alongside an IDE would have been extremely beneficial to candidates and centres.

Question 8

8 OCR town are holding an election with three candidates (A, B and C). An electronic voting booth will be used to allow people to vote.

Write an algorithm that:

- Allows voters to enter either A, B or C.
- Keeps track of how many times each candidate has been voted for.
- As soon as one person has finished voting, allows the next person to vote.
- At any point allows the official to type in "END", which will print out the number of votes for each candidate and the total number of votes overall.

[6]

This question required candidates to write an algorithm (which could equally have been pseudocode or a flowchart) to count how many votes three candidates received in an election. The essential elements of the algorithm were given as bullet points to help candidates to decompose the problem.

The majority of candidates attempted to allow the user to input their choice and then increment a counter variable based on this choice. However, for many candidates, the response provided was not logically correct and so did not achieve all of the marks available. One typical problem was the comparison missing the required quotation marks (or similar) around the literal string value, so while `if vote == ‘B’` checks the value of the variable vote against the string ‘B’, `if vote == B` instead compares two variables. Other common issues included candidates mixing up the left and right side of an assignment operation (e.g. `c = 1` is not the same as `1 = c`) or overwriting the value of their counters rather than incrementing them (e.g. `a = 1` rather than `a = a + 1` or `a +=1`).

Many candidates stopped at this point with relatively few even attempting to use iteration (or other methods) to ensure that further votes could be counted. Where this was attempted, it was pleasing to see candidates understanding condition controlled loops to successfully end when required.

The final criterion was to print the number of votes for each candidate and the number of votes overall. Many candidates successfully completed the first part of this but completely missed out even an attempt at the second part even though it appears to be relatively straightforward; centres should encourage candidates to carefully read the questions and check their answers for completeness.

Examiners saw many answers which used alternative techniques (such as storing the vote counts in an array, or achieving repetition by use of subprograms) and where these were logically correct they were able to achieve full marks.

Exemplar 6

```
Vote = input("Enter A,B or C to vote")
A = 0
B = 0
C = 0
if vote = "A"
        A = A+1
        print("you voted for A")
elif vote = "B"
        B = B+1
        print("you voted B")
elif vote = "C"
        C = C+1
        Print ("you voted C")
elif vote = "END"
        Print (A,B,C)
        END
end if
Return to line 1
```

[6]

This response gained 3 marks. The candidate successfully takes an input from the user and stores this in the variable `vote` [1 mark]. This is then compared against string values 'A', 'B' and 'C', incrementing the appropriate value [1 mark]. The counter variables have previously been initialised to zero [1 mark]. There is some attempt to loop on the very last line, but this is weak and would have the effect of re-initialising the counter values back to zero every time and so is incorrect. The candidate attempts to print the three variables but misses off outputting the total vote count (which would have just been A+B+C).

Exemplar 7

```
A = 0     B = 0     C = 0
END = FALSE
WHILE    END = FALSE
         Vote = user INPUT ('A, B, or C: ')
         IF  Vote == A
                  A = A + 1
         ELIF  Vote == B
                  B = B + 1
         ELIF  Vote == C
                  C = C + 1
         ELIF  Vote = END
                  print ("A = " + A + "B = " + B + "C = " + C + "Total = "
                                int(A+C+B))
                  END = TRUE
         ELSE
                  print ("Invalid")
END WHILE
```

[6]

This response achieved 5 marks, just 1 mark short of the maximum. The three counter values are initialised [1 mark] and user input successfully taken and stored in a variable [1 mark]. This input is then compared against A, but the lack of quotation marks (or equivalent) around A means that this is a variable (which has been defined on the very first line), meaning that the comparison being undertaken is effectively 'is the user input equal to zero?', which is incorrect. However, there is a loop present around the relevant instructions [1 mark] and this does repeat until the user enters 'END' [1 mark] (the initial error of missing the quotation marks on comparisons being followed through and only penalised once), at which point the values of A, B, C and the total are all output [1 mark]. It is pleasing to see that the candidate also includes some basic input validation, although this is not asked for in the question and so cannot be credited.

Supporting you

For further details of this qualification please visit the subject webpage.

Review of results

If any of your students' results are not as expected, you may wish to consider one of our review of results services. For full information about the options available visit the OCR website. If university places are at stake you may wish to consider priority service 2 reviews of marking which have an earlier deadline to ensure your reviews are processed in time for university applications.

activeresults

Active Results offers a unique perspective on results data and greater opportunities to understand students' performance.

It allows you to:

- Review reports on the **performance of individual candidates**, cohorts of students and whole centres
- **Analyse results** at question and/or topic level
- **Compare your centre** with OCR national averages or similar OCR centres.
- Identify areas of the curriculum where students excel or struggle and help **pinpoint strengths and weaknesses** of students and teaching departments.

http://www.ocr.org.uk/administration/support-and-tools/active-results/

CPD Hub
Your route to OCR's teacher training

Attend one of our popular CPD courses to hear exam feedback directly from a senior assessor or drop in to an online Q&A session.

https://www.cpdhub.ocr.org.uk