Qualification Accredited

Oxford Cambridge and RSA

GCSE (9-1)

Exemplar Candidate Work

COMPUTER SCIENCE

J276
For first teaching in 2016

J276/02 Summer 2019 examination series

Version 1

Contents

Would you prefer a Word version?

Did you know that you can save this pdf as a Word file using Acrobat Professional?

Simply click on **File** > **Export to** and select **Microsoft Word**

(If you have opened this PDF in your browser you will need to save it first. Simply right click anywhere on the page and select ***Save as . . .*** to save the PDF. Then open the PDF in Acrobat Professional.)

If you do not have access to Acrobat Professional there are a number of **free** applications available that will also convert PDF to Word (search for *pdf to word* converter).

We value your feedback

We'd like to know your view on the resources we produce. By clicking on the icon above you will help us to ensure that our resources work for you.

Introduction

These exemplar answers have been chosen from the summer 2019 examination series.

OCR is open to a wide variety of approaches and all answers are considered on their merits. These exemplars, therefore, should not be seen as the only way to answer questions but they do illustrate how the mark scheme has been applied.

Please always refer to the specification https://www.ocr.org.uk/qualifications/gcse/computer-science-j276-from-2016/ for full details of the assessment for this qualification. These exemplar answers should also be read in conjunction with the sample assessment materials and the June 2019 Examiners' report or Report to Centres available from Interchange https://interchange.ocr.org.uk/.

The question paper, mark scheme and any resource booklet(s) will be available on the OCR website from summer 2020. Until then, they are available on OCR Interchange (school exams officers will have a login for this and are able to set up teachers with specific logins – see the following link for further information http://www.ocr.org.uk/administration/support-and-tools/interchange/managing-user-accounts/).

It is important to note that approaches to question setting and marking will remain consistent. At the same time OCR reviews all its qualifications annually and may make small adjustments to improve the performance of its assessments. We will let you know of any substantive changes.

Question 1(a)(i)

1 **(a)** A radio station records an interview with a computer scientist using a computer and audio recording software.

(i) Explain how sampling is used to store audio recordings. **[2]**

Exemplar 1 — 2 marks

Explain how sampling is used to store audio recordings.

Sound is measured by recording the amplitude in regular set intervals. This is known as sampling.

Examiner commentary

The candidate here clearly identifies that the amplitude (or height) is measured and that this is done at regular intervals. The fact that these samples are converted to digital or binary was another mark point that was available, but this candidate has already gained full marks for this question.

Exemplar 2 — 1 mark

Audio frequency is Sampled (recorded) and digitised which is repeatedly done to replicate the audio which is heard. Analogue sounds are turned into digital ones through this method.

Examiner commentary

A very common mistake was that candidates misunderstand what exactly is sampled; here a candidate discusses the frequency being measured which is obviously incorrect. The mark for digitising the sample was given and the candidate in fact repeats this point twice (although only getting one mark). For data representation questions, examiners expect candidates to know the scientific details of exactly what is measured or stored.

Question 1(a)(ii)

A second interview with the computer scientist is recorded. Before this interview, the sampling frequency in the audio software is increased.

(ii) Define what is meant by the term **sampling frequency**. **[1]**

Exemplar 1 — 1 mark

Define what is meant by the term **sampling frequency.**

Sampling frequency is the number of samples recorded per second.

Examiner commentary

Sampling frequency is correctly defined here as the number of samples taken in a given time period (here given as a second). The use of the unit 'hertz' to represent once per second is encouraged to be discussed by centres with their candidates, although this would not by itself constitute a full answer.

Exemplar 2 — 0 marks

This is the frequency by which the sound waves defect sound.

Examiner commentary

Answers which rely on the words given in the question are to be avoided; here the candidate defines frequency using the word 'frequency'. Examiners are therefore unable to ascertain whether or not the candidate is aware of what the term means.

Question 1(b)(i)

(b) The radio station uses a digital camera to take a photograph of the computer scientist for their website. The photograph is stored as a bitmap image.

(i) Describe how bitmap images are represented in binary. **[3]**

Exemplar 1 — 3 marks

The image is split up into pixels. ~~These~~ For each pixel there is a binary number which represents it color. The amount of bits per pixel is the colour depth. It stores the first row of pixels left to right then the next, creating a long list of all the pixels and color in the image.

Examiner commentary

This candidate answer covers the points from the mark scheme in a simple fashion, showing clear understanding of how images are stored at bitmaps. The idea of pixels being the basic element of bitmaps is central to this well-communicated answer.

Exemplar 2 — 0 marks

- They are 4 bits back; ~~and are represented in a 4 binary~~
- Display file size, etc.
- uses binary for colour.
- every colour used will be in a 4 bit binary code.
- Depending on bits, depending on file bite.

Examiner commentary

The answer given by the candidate here attempts to discuss how an image using up to 16 colours could be stored, but gives a list of simple ideas rather than trying to link them together as a whole. Although the answer shows some understanding, it is far too vague and fails to clearly hit the points in the mark scheme. It should be noted that not all images use 4 bits per pixel as this candidate seems to suggest, with the bit depth altering depending on how many colours are represented.

Question 1(b)(ii)

(ii) Explain why computers represent data in binary form. **[2]**

Exemplar 1 — 2 marks

This is so it is direcly exicutable by the CPU. The transistors are eveither on or off represented by 1s and 0s. Binary is all computers can understand.

Examiner commentary

The question of why data is represented using binary is central to the study of Computer Science. Candidates should recognise, as this example shows, that computers store data using electronic switches (transistors in this answer) that can be on or off. Note that in the mark scheme for this question, the mark for 'on or off' was dependent on describing what could be turned on or off, therefore avoiding giving marks to candidates who simply describe what binary is. The candidate shown here clearly understands this central premise and communicates this well.

Exemplar 2 — 0 marks

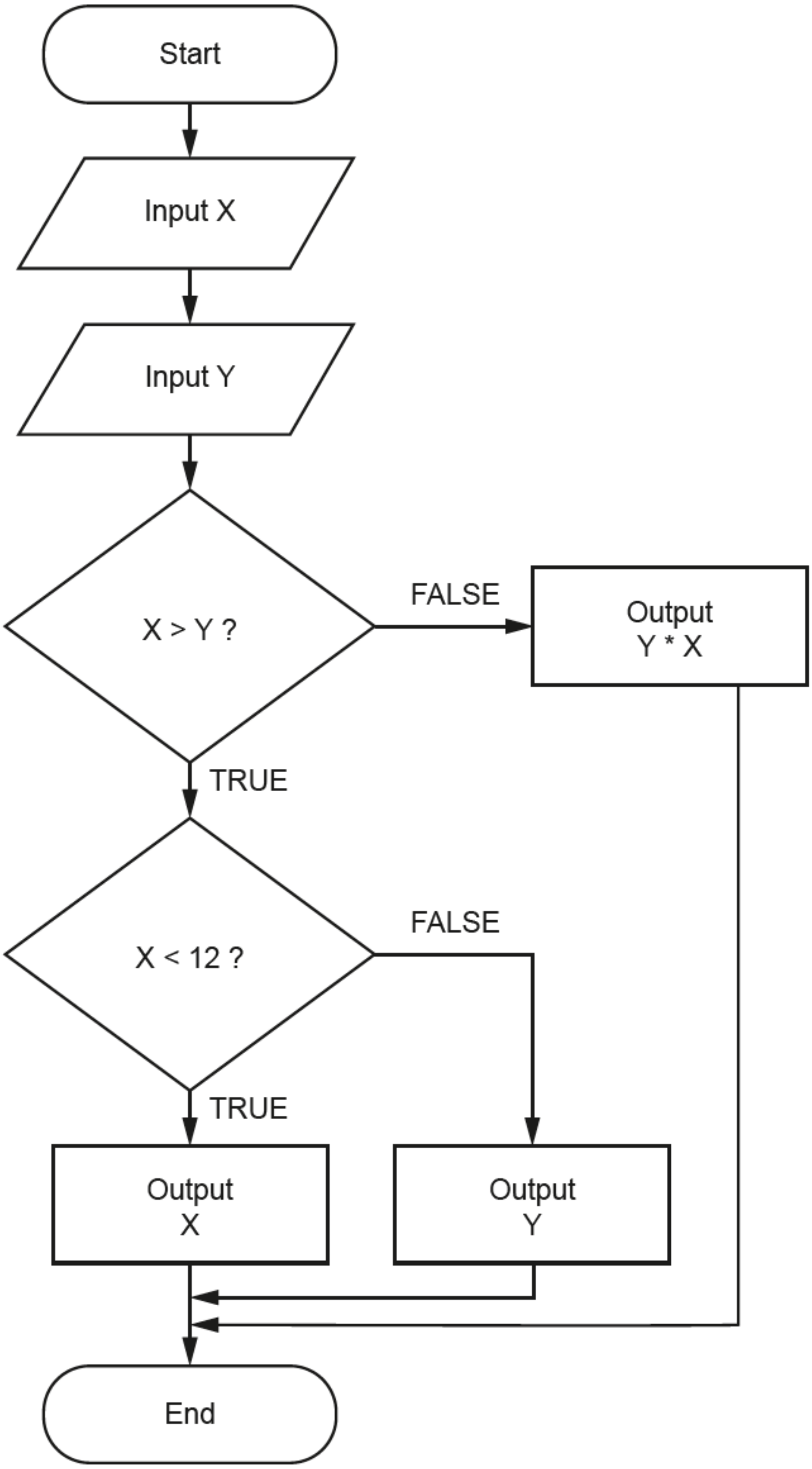
~~Be~~ Computers use binary because its more efficient because they can understand translate it better

Examiner commentary

The candidate in this example is typical of several answers who are clearly unaware of why data is represented in binary. This rather simplistic answer is below the level expected of a Computer Science candidate and centres should strive to make sure that candidates understand the 'why' as well as the 'how'.

Question 2

2 A programmer creates an algorithm using a flow chart.



Question 2(b)

(b) Write this algorithm using pseudocode. **[6]**

Exemplar 1 6 marks

```
START
x = input ('Type a number')
y = input ('Type a number')
if x > y:
        if x < 12:
                output x
        else:
                output y
else:
        x = x ** y
        output x
```

Examiner commentary

This question asked candidates to rewrite an algorithm using pseudocode; the algorithm had previously been given as a flowchart, so the logic required was already given. However, the difficulty in this process was the understanding that a TRUE answer to the first decision meant that another decision was required. This could have been managed, as shown in this example, by using nested IF statements so that the second was only checked on the first being TRUE. However, there were many other ways of completing this, including checking for the opposite (e.g. if x <=y). As a basic principle, if the algorithm produced the correct output for all possible input values of X and Y (including where they are equal) then the candidate would have been given full marks.

Exemplar 2 5 marks

```
X = input ("Please enter a number")
Y = input ("Please enter a number")
if X > Y   then
    if X < 12   then
    print (X)
    else
    print (Y)
endif
~~endif~~
~~elseif~~
if X < Y  then
```

```
print (Y * X)
endif
```

Examiner commentary

This answer is shown as an example of a typical mistake made by candidates. If X is not larger than Y, then the candidate here checks if X is smaller than Y. However, this would not work if X was equal to Y and therefore is not logically valid. Here, the mistake costs a single mark but if done elsewhere it may have removed multiple marks from an otherwise sensible answer. Candidates should be encouraged to think logically about their answers.

Exemplar 3 — 0 marks

```
Print ("Input X")
Print ("Input y")
User = Input
Print ("x > y ?")
If False
Print ("output y*x")
If True y * x
Print ("x < 12?")
If False output y
Print ("output y")
If True output X
Print (" output x")
End
```

[6]

Examiner commentary

This example shows an answer that has multiple issues, all of which mean that no marks were able to be given. Firstly, all of the outputs are string values and give no way of outputting the values of X and Y. Secondly, only one input statement is given to input two numbers, showing a lack of understanding. Thirdly and most importantly, the comparisons between X and Y and then between X and 12 are not shown at all; the fifth line of the candidate answer asks, 'IF False', but what is being compared to False? Taken as a whole, the answer does not logically function in the same way as the flowchart at any discernible level.

Question 2(c)

The algorithm is written in a high-level language. The high level code must be translated into machine code before a computer processor can execute it.

(c) Describe **two** methods of translating high level code into machine code. **[4]**

Exemplar 1 — 4 marks

1 One method is using a compiler. This compiles all your code ~~and~~ checks it and translates it before running. It checks if there's any error/ mistakes before your code/program is executed

2 Another method is using an interpreter. This ~~translated~~ translates code line by ~~[illegible]~~ line from a high level to machine code whilst executing a program/code.

Examiner commentary

The example shown here gains full marks by naming two suitable methods of translating high-level code (compiler and interpreter) and describing clearly how they differ. This is a very good example answer.

Exemplar 2 — 0 marks

1 Binary as computers are able to translate high level code into a language only computers can understand to low level what is which the human people can actually read.

2 The ~~[illegible]~~ translation of transferring the code can also be done by ~~hexadec~~ hexadecimal as it translates the number operations and turns it into a way that computer can process quickly.

Examiner commentary

Interpreters and compilers are given in the specification as examples of translators and naming these plus describing how they work at a basic level was required for full marks to be given. Candidates could have also gained partial marks by simply describing the process of translating instructions one-by-one or as a whole before execution. The candidate shown here perhaps does not understand the concept of translation at all and so provides an answer that does not meet any of the points on the mark scheme. Centres should encourage candidates to be familiar with the specification and particularly the contents list for each unit; this could be used as a revision checklist. In this case, any candidate who had ticked off 'the characteristics of an assembler, a compiler and an interpreter' (section 2.5) would have stood a good chance of gaining marks on this question.

Question 4(a)(i)

4 Elliott plays football for OCR FC. He wants to create a program to store the results of each football match they play and the names of the goal scorers. Elliott wants individual players from the team to be able to submit this information.

(a) (i) Define what is meant by **abstraction**. **[2]**

Exemplar 1 2 marks

Taking a problem and removing all unnecessary information, leaving only relevant information.

Examiner commentary

Abstraction is a new topic to this specification and it is pleasing to see candidates using the term correctly. Here, the candidate gives a clear definition or removal of unnecessary data (not just any data) and focusing on what remains. The point of this is to make the problem simpler to solve. Abstraction in Computer Science can take many forms and much use can be made of the term at much higher levels than this, but the level of question posed here is enough knowledge for GCSE level.

Exemplar 2 0 marks

Taking a big task and breaking it down into smaller, more manageable ones.

Examiner commentary

A very common mistake was to confuse abstraction with decomposition. Here the candidate gives a comprehensive definition of decomposition. However, this was not the question asked!

Question 4(a)(ii)

(ii) Give **one** example of how abstraction could be used when developing this program. **[1]**

Exemplar 1 — 1 mark

taking out data such as what School did you go to, [1]

Examiner commentary

A relatively simple question that followed on from the definition, the mark here could be given for any sensible example of something that could be hidden, ignored, removed or focused on. Here, the candidate sensibly suggests removing information about the school each player goes to and gains the mark.

Exemplar 2 — 0 marks

Developing the Storage program ~~[illegible]~~ Separate of the submission one. [1]

Examiner commentary

Despite any sensible example being able to be credited, where answers were not specific about what exactly could be removed or focused on, no mark was given. The candidate example here suggests a change to be made to the program in order to develop something separately, but this does not include anything removed or focused on. The outcome from this is that the examiner must assume that the candidate does not understand the concept of abstraction.

Question 4(b)

(b) Describe **two** examples of defensive design that should be considered when developing this program. **[4]**

Exemplar 1 4 marks

1 Input sanitisation can be used to prevent hackers from ~~input~~ inputting malicious strings in order to return or find vunerbillities sensitive data. They can ~~do~~ prevent this by removing excess characters on strings from data inputted (length check)

2 Authorisation, in the form of passwords that can only ~~be used only allow~~ allow access to authorised users (players from the OCR FC team) to input data.

Examiner commentary

Defensive design was perhaps not well understood by candidates sitting this paper. The specification makes a list of some example techniques that fit under this umbrella and the candidate in this example makes full use of these, describing how they could be used with the example program given.

Exemplar 2 0 marks

program.

1 Commence ~~[illegible]~~ licences - to protect his work and to have control to what ~~[illegible]~~ the code writer would like ~~to~~ it to be used for.

2 ~~Put~~ or placing a patent on the code, to prevent people from stealing it and keep it closed source to protect people altering it.

Examiner commentary

Conversely, the candidate shown here does not clearly understand the topic of defensive design and discusses various methods of ensuring the work cannot be copied; these would be better suited to a question relating to legal issues.

Question 4(c)

The number of goals scored in each football match is held in an array called `goals`. An example of this array is shown.

```
goals = [0, 1, 3, 0, 4, 5, 2, 0, 2, 1]
```

Elliott wants to count how many matches end with 0 goals.

(c) Complete the following pseudocode for an algorithm to count up how many matches with 0 goals are stored in the array and then print out this value. **[3]**

Exemplar 1 — 3 marks

```
nogoalscount = 0
for count = 0 to (goals.length-1)
    if goals[count] == 0 then
        nogoalscount = nogoalscount + 1
    endif
next count
print(Nogoalscount)
```

Examiner commentary

This candidate can demonstrate understanding of the use of a count variable within a FOR loop, leading to being able to iterate around each item in an array. The use of FOR loops is extremely common when dealing with arrays and centres should encourage candidates to explore linking ideas together in ways such as this. Although the third answer uses a different case than the variable given in the question (capital N used), this was not penalised as not all high-level languages are case sensitive. However, centres may wish to make sure that candidates understand that this may cause problems when programming in the classroom.

Exemplar 2 — 0 marks

```
nogoalscount = 0
for count = 0 to (goals.length-1)
    if goals[$7] == 0 then
        nogoalscount 0
    endif
next count
print("you've got no goals")
```

Examiner commentary

The example shown here illustrates what could potentially happen if candidates are not given enough classroom programming experience. The items for each answer are not only incorrect, they are not logically sensible or in any way related to an answer that would make sense. Examiners are happy to give credit and be generous where the intention is clear but were not able to in this case. The candidate here would perhaps also benefit from spending more time reading the requirement of the question.

Question 6(a)(i)

6 OCR Land is a theme park aimed at children and adults. Entrance tickets are sold online. An adult ticket to OCR Land costs £19.99, with a child ticket costing £8.99. A booking fee of £2.50 is added to all orders.

(a) A function, `ticketprice()`, takes the number of adult tickets and the number of child tickets as parameters. It calculates and returns the total price to be paid.

(i) Use pseudocode to create an algorithm for the function `ticketprice()`. **[6]**

Exemplar 1 — 6 marks

```
FUNCTION ticketprice(adulttickets, childtickets)
    ~~adultprice =~~
    ~~fee = 2.50~~
    FLOAT Fee = 2.50
    adultprice = adulttickets * 19.99
    childprice = childtickets * 8.99
    total = adultprice + childprice + fee
    RETURN total
ENDFUNCTION
```

Examiner commentary

The example shown here is a great example of a candidate able to achieve full marks to this question, hitting every point on the mark scheme by creating a function with parameters passed in which are then used to calculate and return a value. Many candidates presented their answers using keywords from high-level languages (such as def from Python); this was equally acceptable.

Exemplar 2 — 2 marks

```
Input Number of Adult ()
Input Number Of children ()
Price = (number of adult * 19.99) + (Number of children *
8.99)
Booking Fee = 2.50
(Price + Booking fee) = Overall price
Print Overall price ()
```

Examiner commentary

Disappointingly, many candidates chose to ignore the requirement to create the algorithm as a function, such as the candidate shown here. Although the question clearly asks for parameters to be passed in and a value returned, this example shows that values are instead inputted, and the answer printed; these are not analogous and were not given the marks. Centres should make sure that candidates are familiar with the use of subroutines such as procedures and functions, including using existing subroutines and creating their own. In this example, the candidate makes a further logical error when assigning the OverallPrice, with the assignment statement the wrong way around.

Question 6(c)(i)

(c) A list of valid discount codes is shown below.

[NIC12B, LOR11S, STU12M, VIC08E, KEI99M, WES56O, DAN34S]

(i) State **one** reason why a binary search would not be able to be used with this data. **[1]**

Exemplar 1 — 1 mark

~~It is not sorted.~~ The list is not sorted but binary search only works with lists that are.

Examiner commentary

A binary search must be in an order for the search to be able to function properly. The candidate above recognised this and was able to gain the mark for the question.

Exemplar 2 — 0 marks

because its a ~~mix~~ mix of letters and numbers

Examiner commentary

The candidate shown here has perhaps only used searches and sorts in lessons with numeric values and so has the misconception that they can only be applied to numbers. In fact, as long as the values can be compared (e.g. put into alphabetical order or size order) these algorithms will work without issue. Centres may wish to go beyond the basic 'put these numbers into order' exercises with candidates and think about how else items could be organised; by size, by value or by date for example.

Question 6(d)(i)

(d) OCR Land keeps track of the size of queues on its rides by storing them in an array with the identifier `queuesize`. It uses the following bubble sort algorithm to put these queue sizes into ascending numerical order.

```
swaps = True
while swaps
    swaps = False
    for p = 0 to queuesize.length-2
        if queuesize[p] > queuesize[p+1] then
            temp = queuesize[p]
            queuesize[p] = queuesize[p+1]
            queuesize[p+1] = temp
            swaps = True
        endif
    next p
endwhile
```

(i) Explain the purpose of the Boolean variable `swaps` in this bubble sort algorithm. **[2]**

Exemplar 1 2 marks

The 'swaps' variable checks if the bubble sort algorithm had to swap the ~~of~~ queue sizes around and stops the while loop when there were no swaps (i.e. if the ~~h~~ queue sizes were sorted) **[2]**

Examiner commentary

The candidate hits both mark points here. The answer explains that the variable is used to record if a swap has taken place, which is also used as the condition for the while loop. Note that the candidate uses the past tense 'had to swap' to show that the swap variable is set as a result of the swap and not the other way around.

Exemplar 2 0 marks

To decide is there are values in the array need to be swapped or not. Is it is 'True' a swap is required, is 'False' no swap is required

Examiner commentary

The example shown here is a typical misconception from candidates. The value of the variable here is a flag to record if a swap has taken place; the candidate instead reverses this and says that the variable can be used to see if a swap is needed. This is logically incorrect and shows that although the steps of bubble sort algorithm may be well understood, the mechanics of it are perhaps less well known by candidates.

Question 6(d)(ii)

(ii) Explain the purpose of lines **06** to **08** in this bubble sort algorithm. **[2]**

Exemplar 1 — 2 marks

This swaps ~~queues~~ queuesize [p] with the ~~nxt~~ next queuesize [p+1] is queuesize [p+1] has more people in.
Eg. Queue Ride 1 and Ride 2, is there are more people queuing for Ride 1 then the program swaps the 2 values around:
Ride 2, Ride 1

Examiner commentary

The purpose of the lines given was to swap the two values over in the array. The candidate here explains this well and can achieve both marks.

Exemplar 2 — 0 marks

Creates an open array which can then be modified by adding onto it.

Examiner commentary

The candidate here misses the idea of swapping values. They have recognised that an array has been used, but not what happens to the value within it. This could perhaps be related to the use of Python which uses lists by default and so allows values to be added (which is not possible with static arrays).

Question 6(e)

(e) One ride in OCR Land has a minimum height of 140 cm to ride alone or 120 cm to ride with an adult.

Create an algorithm that:

- asks the user to input the height of the rider, in centimetres
- if needed, asks if they are riding with an adult
- outputs whether or not they are allowed to ride
- repeats this process until 8 people have been allowed to ride.

[8]

Exemplar 1 — 2 marks

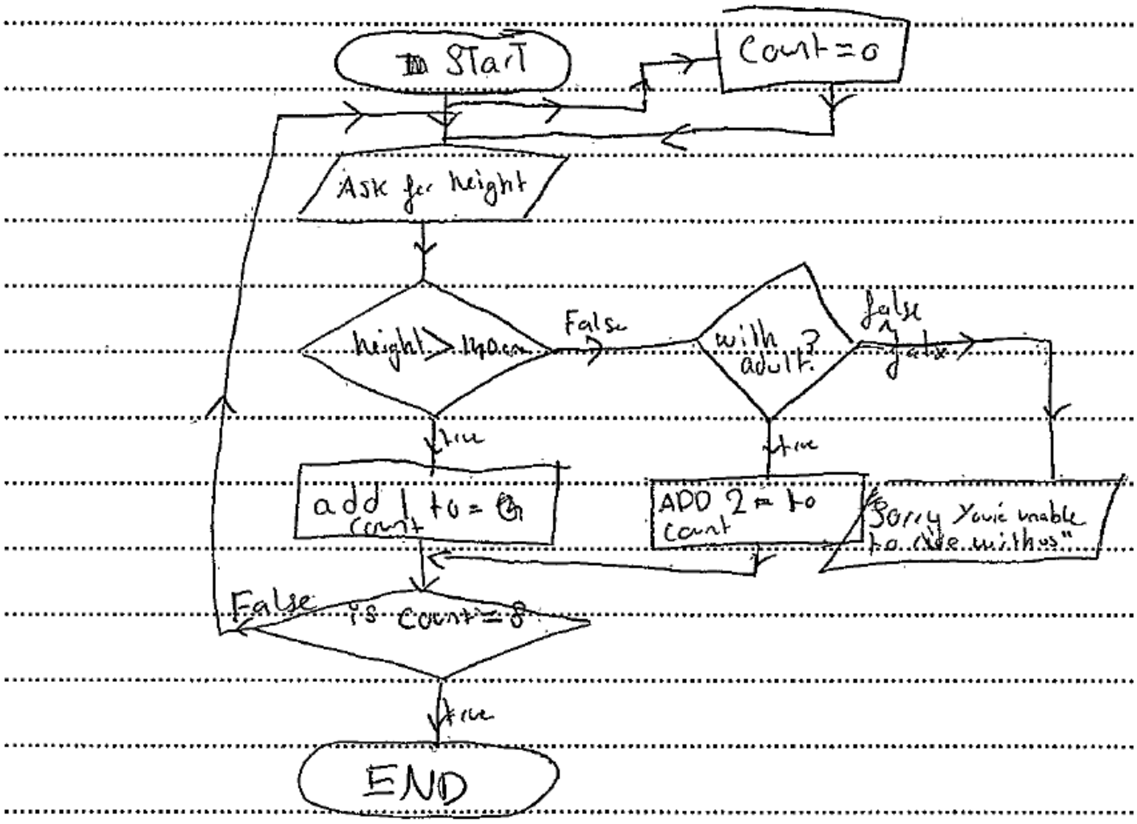
```
Print (" Input your height , in centimetres")
Height = H
If Height [H] < 120
Print (" You are not of the correct height
to ride the ride")
End While
If Height [H] > 120
Print (" You can now ride , Whist accompanied
With an Adult")
Count = +1
If Height [H] > 140
Print (" You can now ride, would you
like to be accompanied by an adult?")
If yes Count +2
If No Count -1
While until Count = 8.
End.
```

Examiner commentary

This example shows a candidate attempting to make sense of the question but perhaps not succeeding as well as they may have hoped. No input is asked for but benefit of doubt given over the use of what looks like an array when using this non-existent input to award the mark for rejecting riders under 120cm. Riders over 140 are also accepted with a partially suitable message. However, the rest of the answer is lacking in logical precision; there is no check for riders between 120 and 140 and the loop is not precise enough in terms of what parts should be repeated.

Exemplar 2

3 marks



Examiner commentary

This example is shown to illustrate that where a question asks for an algorithm, a flowchart could be a suitable response. However, candidates tend to use flowcharts to show what a solution should do rather than how it should do it; in this case, the candidate achieves the mark for asking for height input, for looping eight times and for one of the outputs, but the other outputs are missing. There is also concern about the sequence of instructions, with the 'count = 0' line being included in an ambiguous way. Flowcharts are entirely suitable if done well. It is perhaps mistaken to think that they are somehow 'easier' than using pseudocode. Both type of answer is marked in the same way and with the same expectations of accuracy.

Exemplar 3

8 marks

```
PeopleAllowed = 0
WHILE PeopleAllowed != 8:
    RidesHeight = input(int("What is your height in CM?")
    IF RidesHeight < (120):
        PRINT ("Apologies, you are too short.")
    ELSEIF RidesHeight <140:
        WithAdult = input ("Are you with an adult?")
        IF WithAdult ==("Yes")
            PRINT ("Welcome Aboard")
            PeopleAllowed = PeopleAllowed + 1
        ELSE:
#If they are under 140 without an adult.
            PRINT (" Apologies, you need an adult")
    ELSE:
        PRINT ("Welcome Aboard")
        PeopleAllowed = PeopleAllowed + 1
```

Examiner commentary

The example shown here is an excellent answer that hits all points on the mark scheme. The algorithm uses a WHILE loop to repeat eight times, counting up riders successfully every time. Successive IF/ELSEIF/ELSE statements are used to check for correct heights and produce suitable output, with all three types of rider (under 120, between 120 and 140, over 140) catered for successfully. The ELSE at the very end is skilfully used to catch riders over 140 as all lower heights have already been catered for. This candidate is obviously a competent programmer and deserves to do well.

Did you know?

OCR Resources: *the small print*

Looking for a resource?

There is now a quick and easy search tool to help find **free** resources for your qualification:

www.ocr.org.uk/i-want-to/find-resources/

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**.

General qualifications
Telephone 01223 553998
Facsimile 01223 552627
Email general.qualifications@ocr.org.uk

www.ocr.org.uk