Qualification Accredited

OCR
Oxford Cambridge and RSA

AS LEVEL

Examiners' report

COMPUTER SCIENCE

H046

For first teaching in 2015

H046/02 Summer 2024 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.

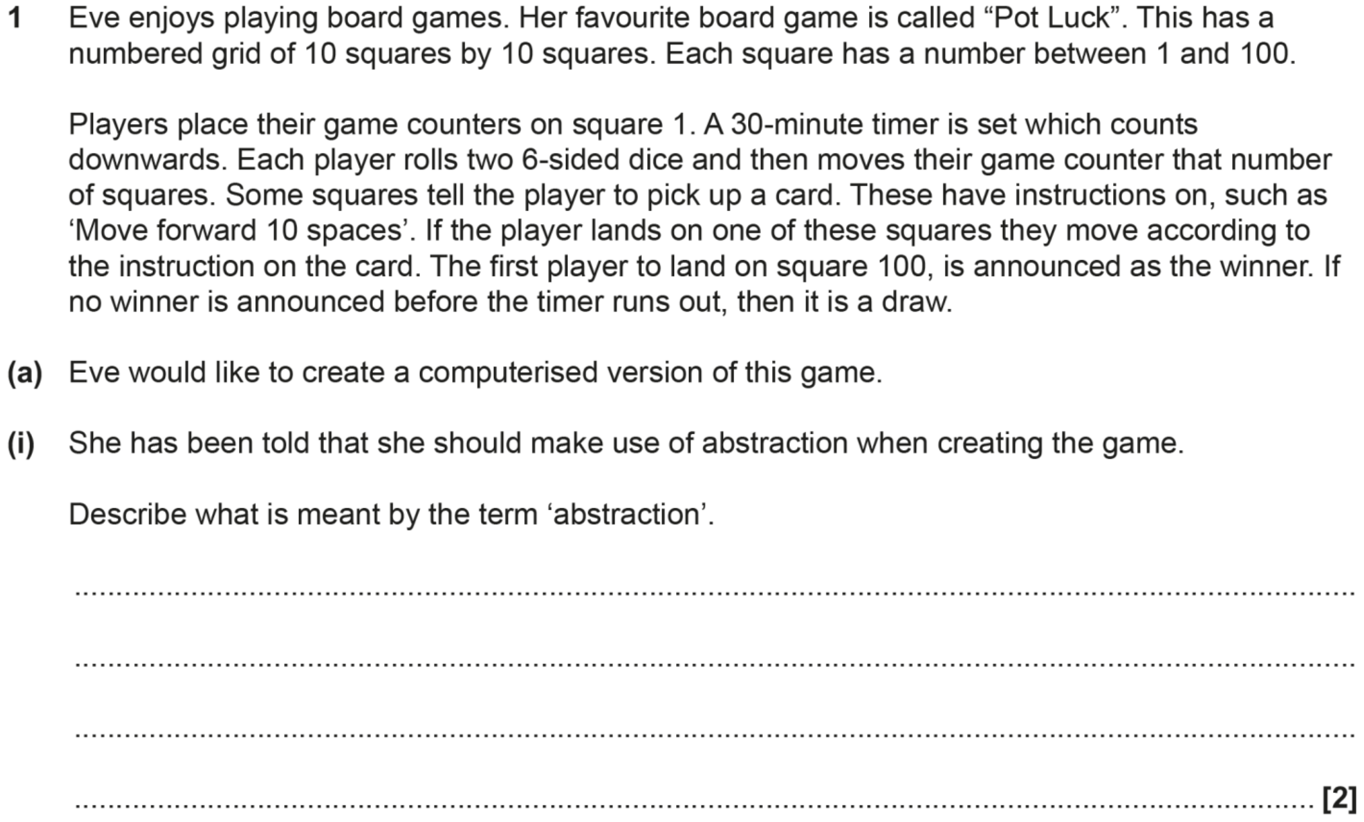
A full copy of the question paper and the mark scheme can be downloaded from OCR.

Paper 2 series overview

Paper 2 focuses on algorithms and problem solving. It tests candidates' computational thinking ability to analyse and solve problems. Candidates are expected to be able to write algorithms fluently in either pseudocode or program code, and to be able to trace algorithms.

Candidates who did well on this paper generally:	Candidates who did less well on this paper generally:
• demonstrated good factual knowledge of terms on the specification • demonstrated an ability to find logical errors in algorithms • demonstrated an ability to analyse questions and apply computational thinking to give relevant responses within the context of a given scenario • demonstrated an ability to evaluate the efficiency of algorithms and to complete standard algorithms such as binary search • demonstrated an ability to fluently write well-structured and logical pseudocode to solve a problem.	• showed some factual knowledge but were less able to show application • showed a limited knowledge of standard sorting and searching algorithms such as insertion sort and binary search • struggled to identify logical conditions or logical errors within algorithms • showed a limited ability to write pseudocode or program code.

Question 1 (a) (i)



1 Eve enjoys playing board games. Her favourite board game is called "Pot Luck". This has a numbered grid of 10 squares by 10 squares. Each square has a number between 1 and 100.

Players place their game counters on square 1. A 30-minute timer is set which counts downwards. Each player rolls two 6-sided dice and then moves their game counter that number of squares. Some squares tell the player to pick up a card. These have instructions on, such as 'Move forward 10 spaces'. If the player lands on one of these squares they move according to the instruction on the card. The first player to land on square 100, is announced as the winner. If no winner is announced before the timer runs out, then it is a draw.

(a) Eve would like to create a computerised version of this game.

(i) She has been told that she should make use of abstraction when creating the game.

Describe what is meant by the term 'abstraction'.

....

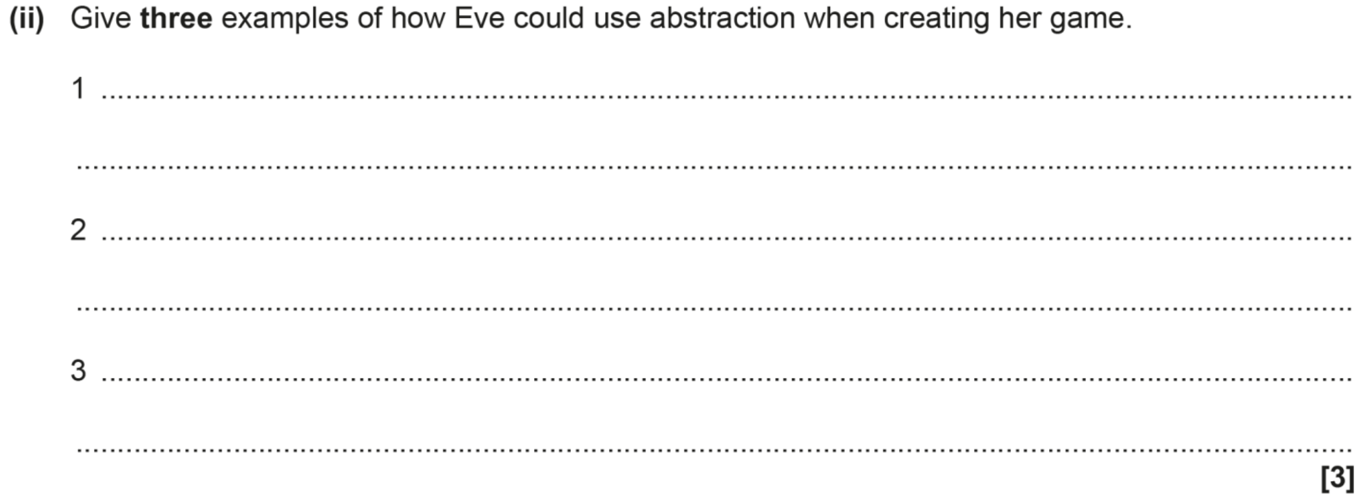
....

....

.... **[2]**

Many candidates could give a definition for abstraction with many specifying that it removed unnecessary detail to simplify a problem.

Question 1 (a) (ii)



(ii) Give **three** examples of how Eve could use abstraction when creating her game.

1 ....

....

2 ....

....

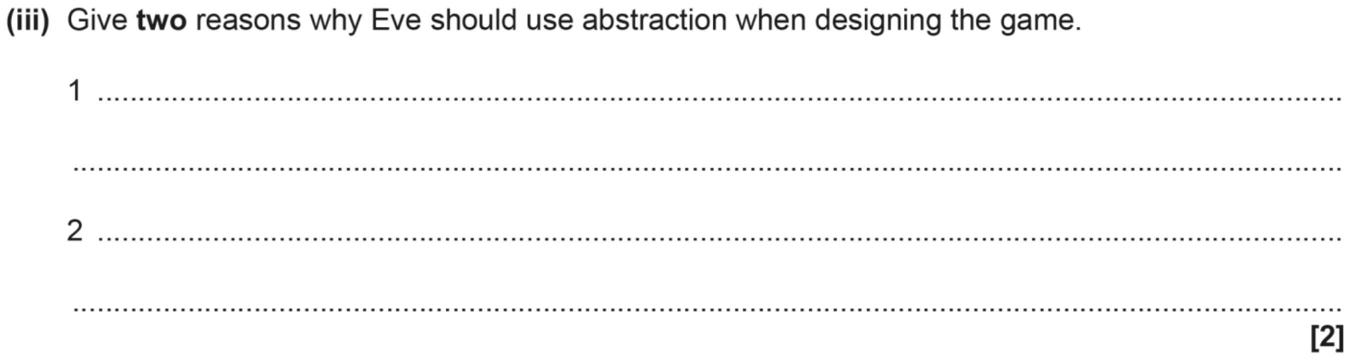
3 ....

....

[3]

Some candidates struggled to use the context of the question to identify components such as the dice, counters, squares, cards or the timer. Candidates need to be able to apply knowledge of abstraction to give suitable responses within context.

Question 1 (a) (iii)



(iii) Give **two** reasons why Eve should use abstraction when designing the game.

1 ....

....

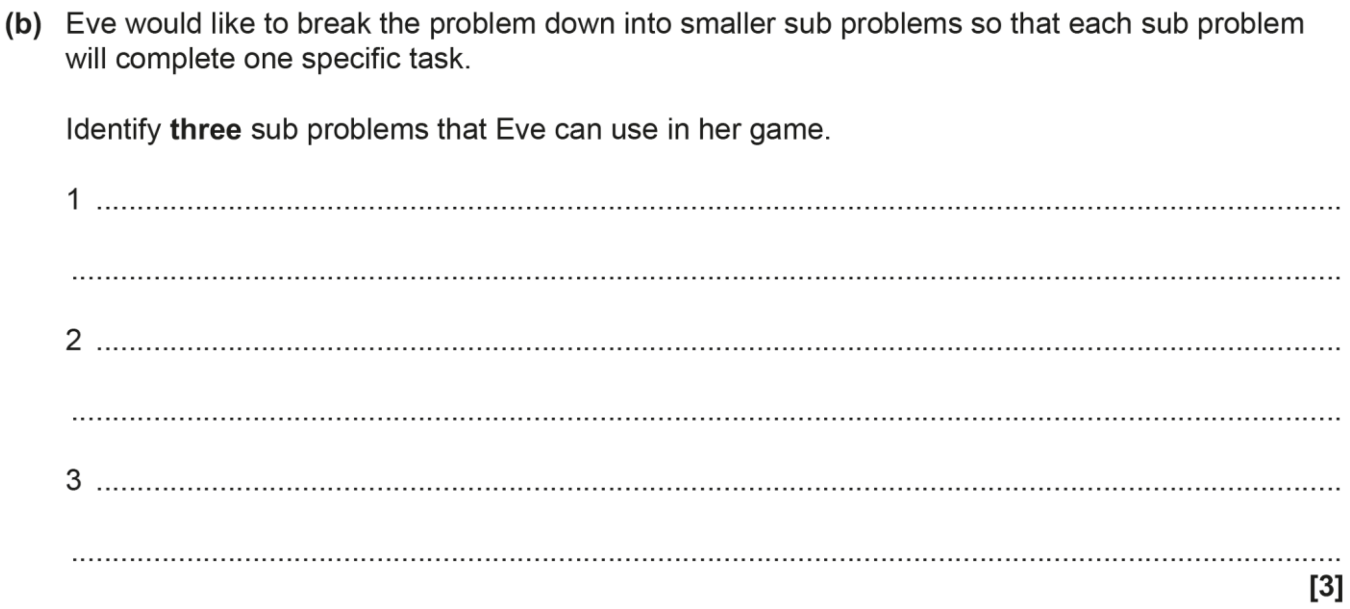
2 ....

....

[2]

Many candidates successfully identified at least one reason why abstraction should be used, such as decreasing the computational processing demands for the solution. Some candidates did however repeated definitions of abstraction, rather than giving the advantages of using it. Another common error was to give an example of how it would benefit a player of the game rather than focusing on the design stage.

Question 1 (b)



(b) Eve would like to break the problem down into smaller sub problems so that each sub problem will complete one specific task.

Identify **three** sub problems that Eve can use in her game.

1 ....

....

2 ....

....

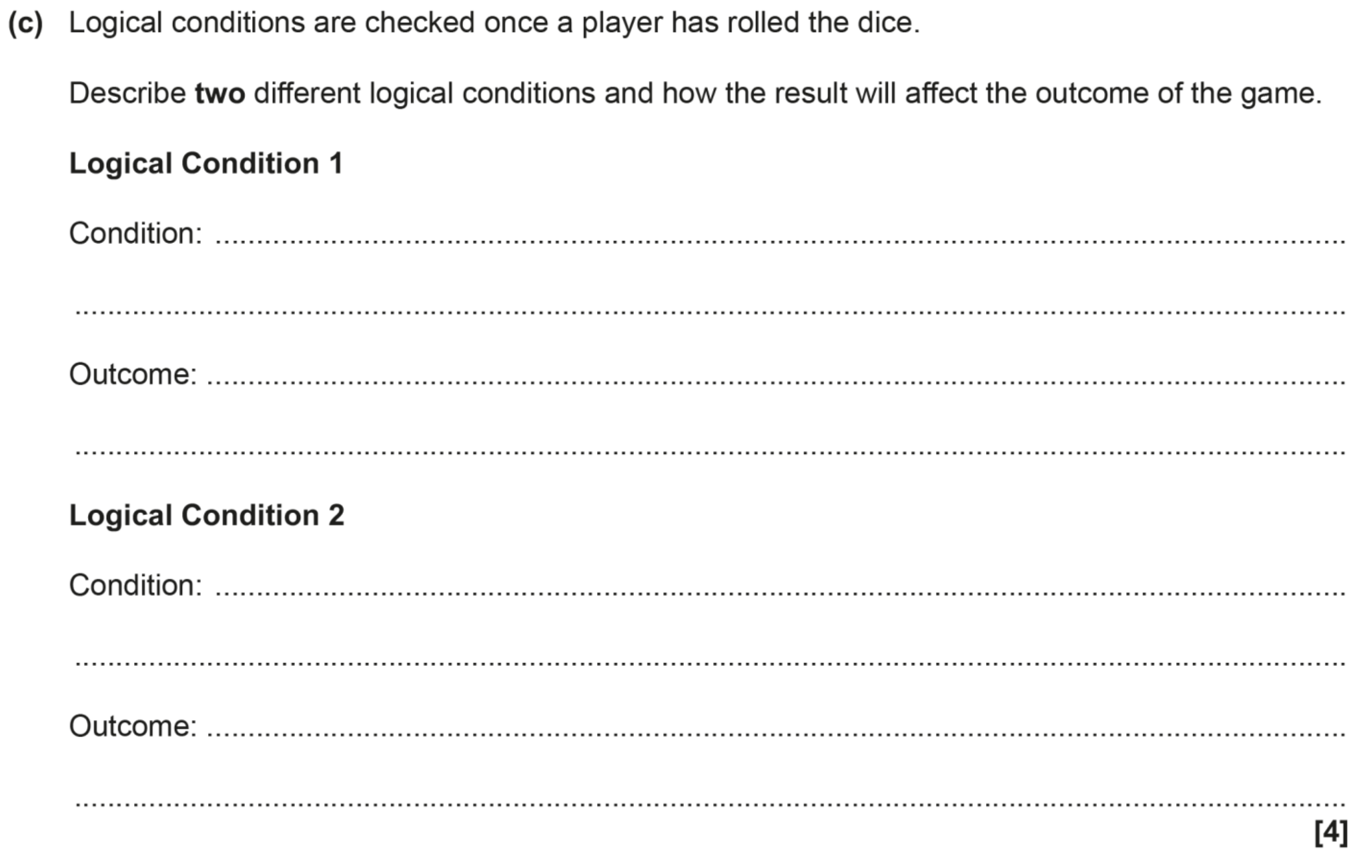
3 ....

....

[3]

Some candidates struggled to apply decomposition to identify parts of the scenario that could be broken down into sub problems. Examples needed to relate to the gameplay for the given scenario and needed to be qualified in some way. For example, ’Dice’ on its own was insufficient, whereas ‘Generating dice score’ was sufficient.

Question 1 (c)



(c) Logical conditions are checked once a player has rolled the dice.

Describe **two** different logical conditions and how the result will affect the outcome of the game.

Logical Condition 1

Condition: ....

....

Outcome: ....

....

Logical Condition 2

Condition: ....

....

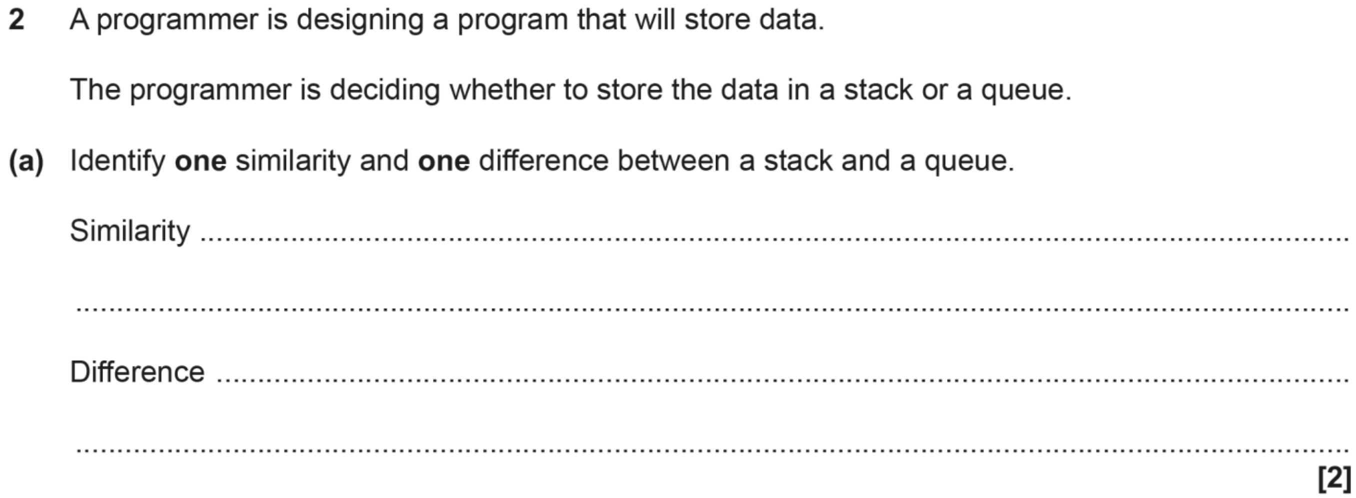
Outcome: ....

....

[4]

Some candidates did not present clear conditions but gave descriptions of actions to perform within the game such as 'roll dice'. The condition required an identification within the context of something that would lead to a Boolean result with consequent outcome. For example, *if* the timer runs out *then* the game will end in a draw.

Question 2 (a)



2 A programmer is designing a program that will store data.

The programmer is deciding whether to store the data in a stack or a queue.

(a) Identify **one** similarity and **one** difference between a stack and a queue.

Similarity ....

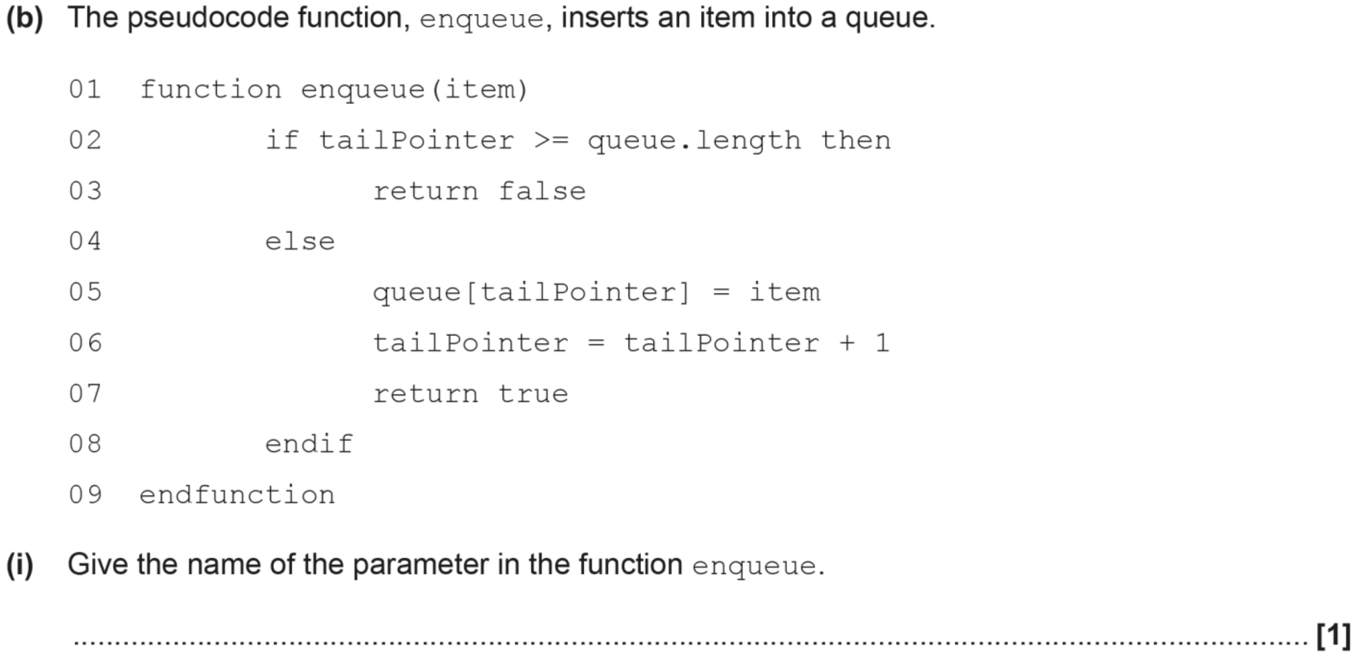
....

Difference ....

....
[2]

Most candidates demonstrated knowledge of the purpose of stacks and queues. For the similarity, a frequent response was that both stacks and queues are data structures used to store data. For the difference, most candidates identified that a stack was a LIFO structure while a queue was a FIFO structure.

Question 2 (b) (i)



(b) The pseudocode function, `enqueue`, inserts an item into a queue.

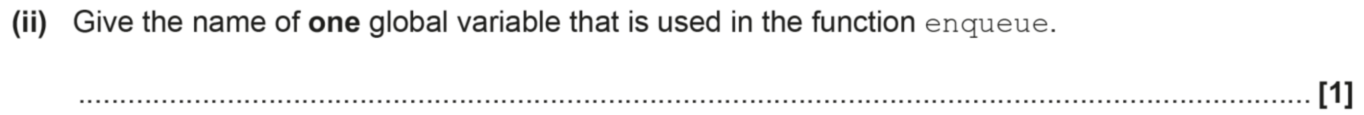
```
function enqueue(item)
       if tailPointer >= queue.length then
             return false
       else
             queue[tailPointer] = item
             tailPointer = tailPointer + 1
             return true
       endif
endfunction
```

(i) Give the name of the parameter in the function `enqueue`.

.... **[1]**

The vast majority of candidates identified that `item` was the parameter.

Question 2 (b) (ii)

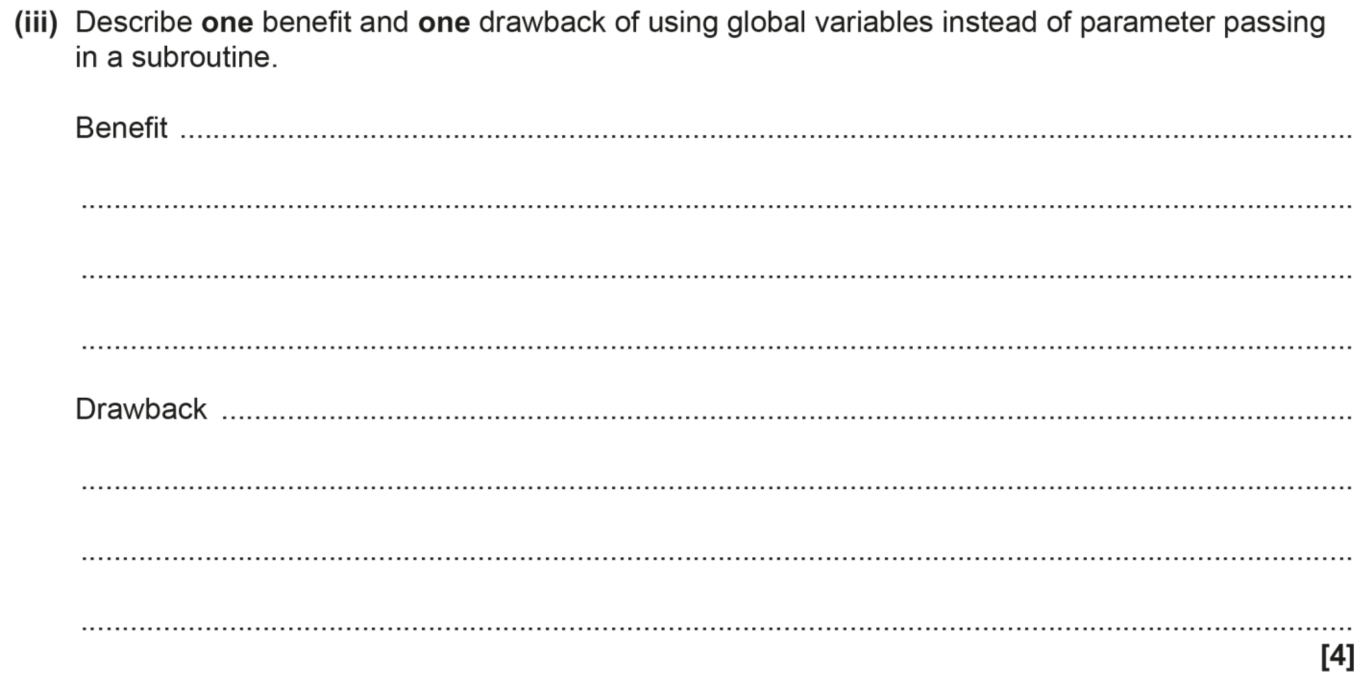


(ii) Give the name of **one** global variable that is used in the function `enqueue`.

.... **[1]**

Many candidates identified that `tailPointer` was an example of a global variable in the code.

Question 2 (b) (iii)



(iii) Describe **one** benefit and **one** drawback of using global variables instead of parameter passing in a subroutine.

Benefit ....

....

....

....

Drawback ....

....

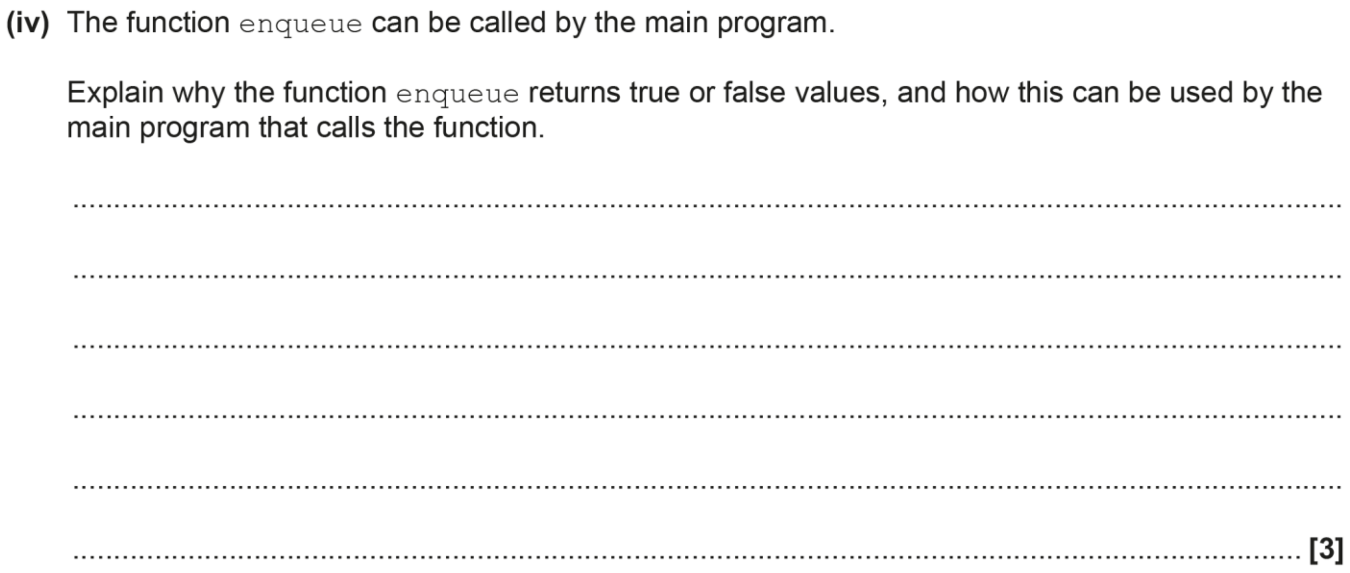
....

....

[4]

The majority of candidates struggled to describe the implications of using global variables, and this highlighted a lack of appreciation of parameter passing and variable scope. At AS Level candidates should become more adept at using parameters and develop an understanding of why this is beneficial – both in theoretical terms and within their personal programming practice. There were few responses that were able to give complete descriptions.

Question 2 (b) (iv)



(iv) The function `enqueue` can be called by the main program.

Explain why the function `enqueue` returns true or false values, and how this can be used by the main program that calls the function.

....

....

....

....

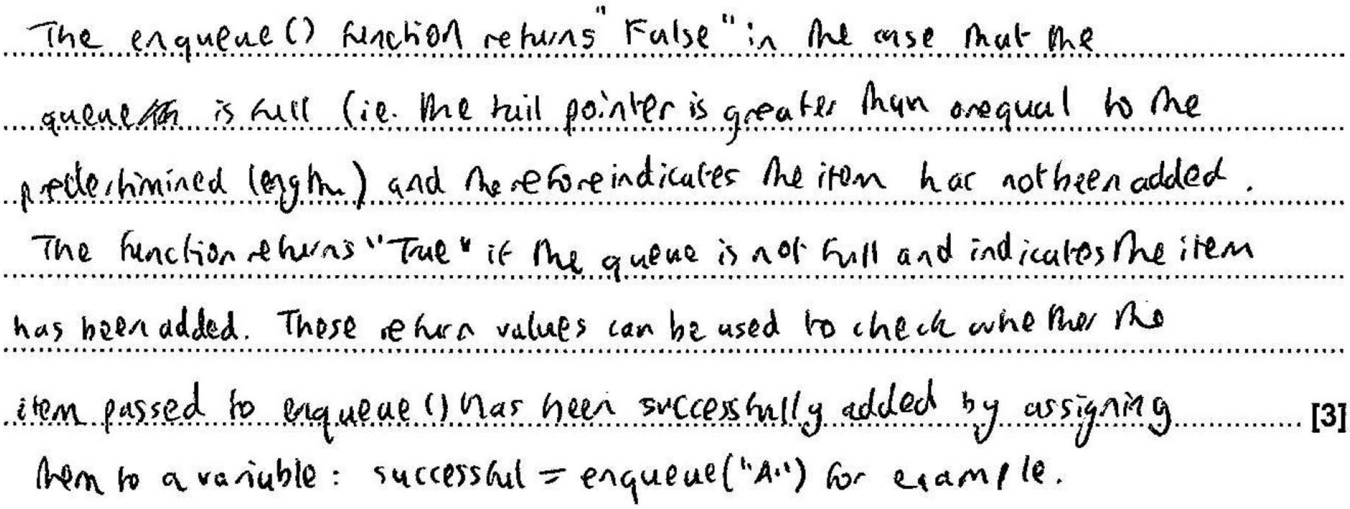
....

.... **[3]**

Some candidates thought that if `enqueue()` returned `True` it meant that there would be space to add additional values. This was not the case; it meant that the item had been enqueued successfully, but it could have been the last item that then filled the queue to capacity.

Sometimes candidates did explain the reasons why `True` and `False` would be returned but were then unable to go on to say how they could be used. Good responses were seen such as 'the main program can give an error message if the operation didn't work if the queue was full'.

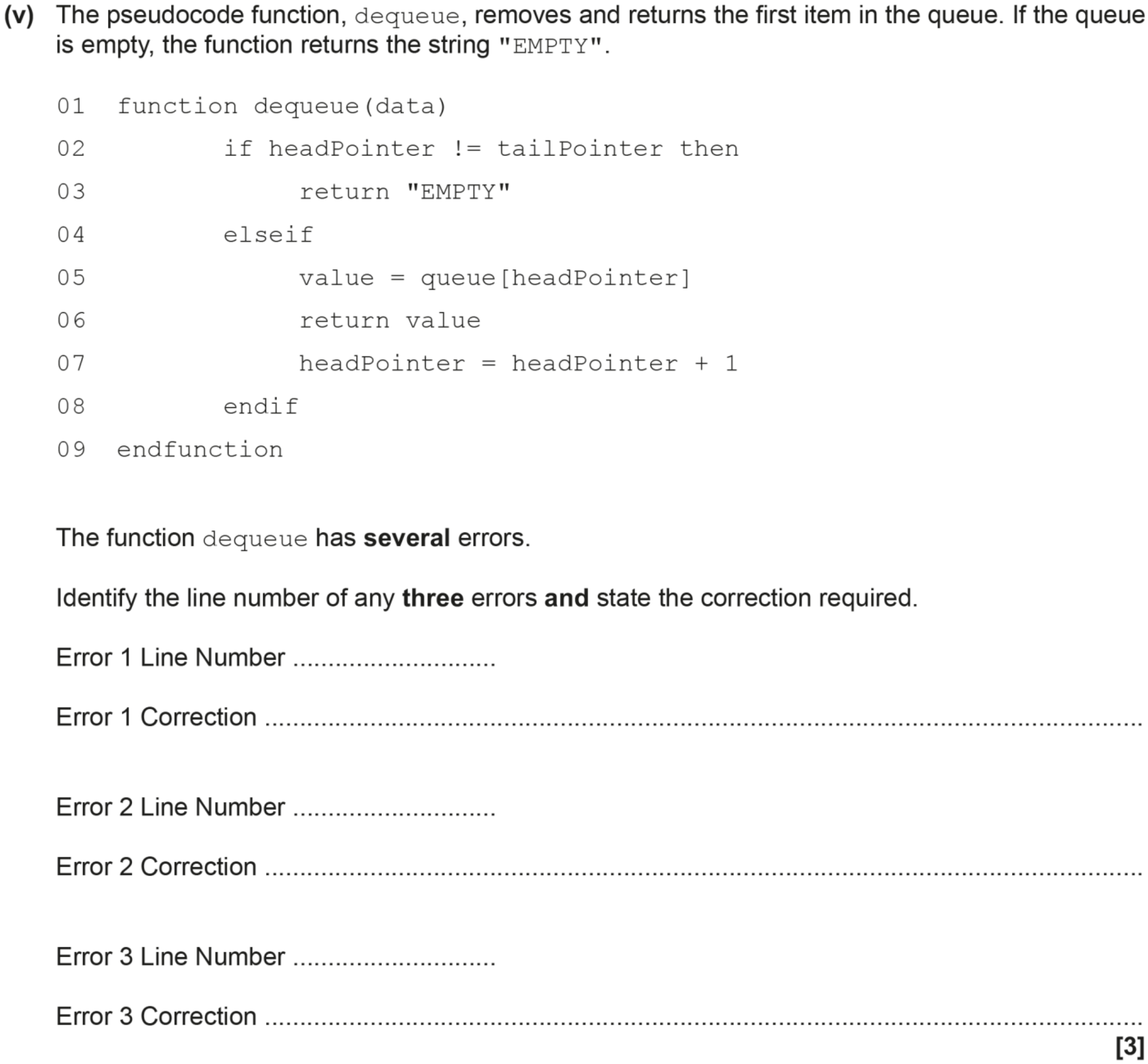
Exemplar 1



The enqueue() function returns "False" in the case that the queue is full (ie. the tail pointer is greater than or equal to the predetermined length) and therefore indicates the item has not been added. The function returns "True" if the queue is not full and indicates the item has been added. Those return values can be used to check whether the item passed to enqueue() has been successfully added by assigning them to a variable: successful = enqueue("A") for example. **[3]**

This response showed clear understanding of the need to store the result returned by `dequeue()` to be used in the conditional loop test.

Question 2 (b) (v)



(v) The pseudocode function, `dequeue`, removes and returns the first item in the queue. If the queue is empty, the function returns the string `"EMPTY"`.

```
01  function dequeue(data)
02          if headPointer != tailPointer then
03               return "EMPTY"
04          elseif
05               value = queue[headPointer]
06               return value
07               headPointer = headPointer + 1
08          endif
09  endfunction
```

The function `dequeue` has **several** errors.

Identify the line number of any **three** errors **and** state the correction required.

Error 1 Line Number ....

Error 1 Correction ....

Error 2 Line Number ....

Error 2 Correction ....

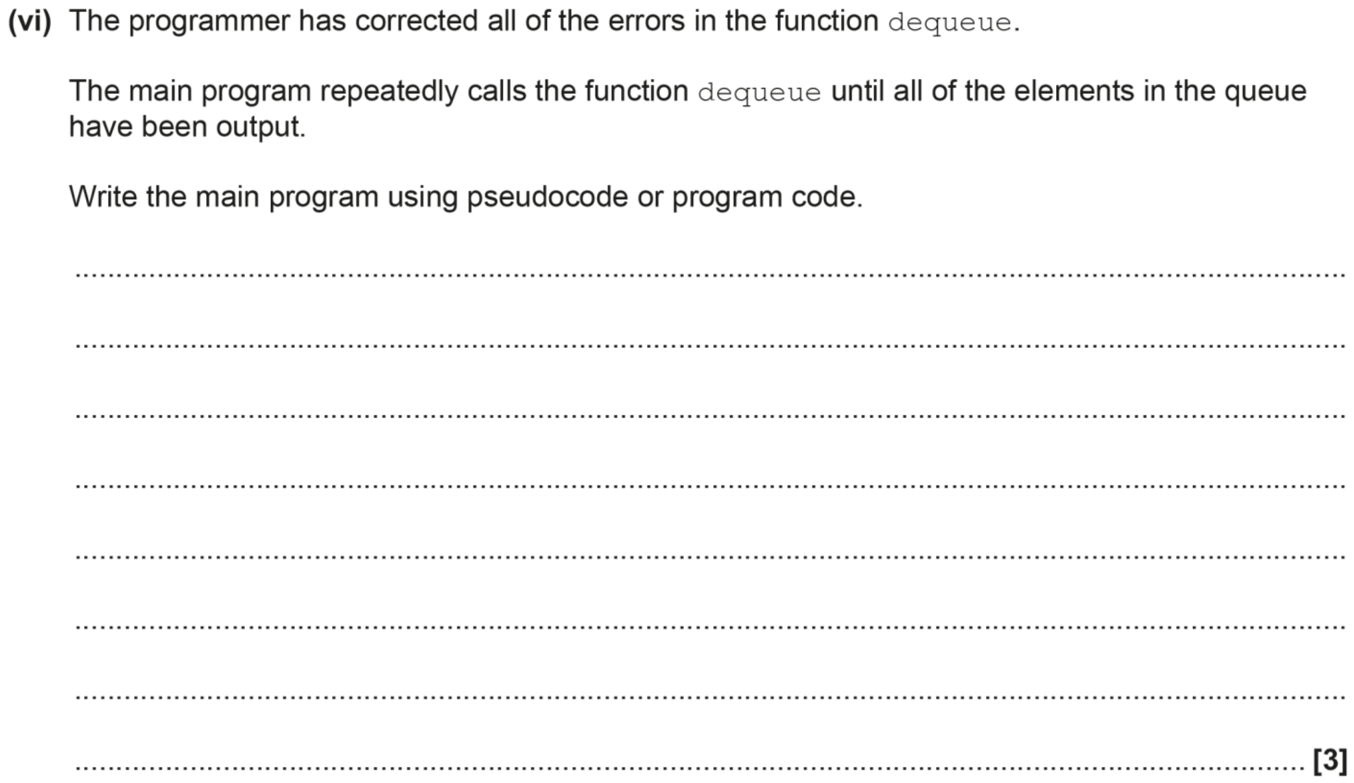
Error 3 Line Number ....

Error 3 Correction ....

[3]

Many candidates identified at least one logical error, but few were able to identify three errors. The most common error identified was the error in line `02`.

Question 2 (b) (vi)



(vi) The programmer has corrected all of the errors in the function `dequeue`.

The main program repeatedly calls the function `dequeue` until all of the elements in the queue have been output.

Write the main program using pseudocode or program code.

....

....

....

....

....

....

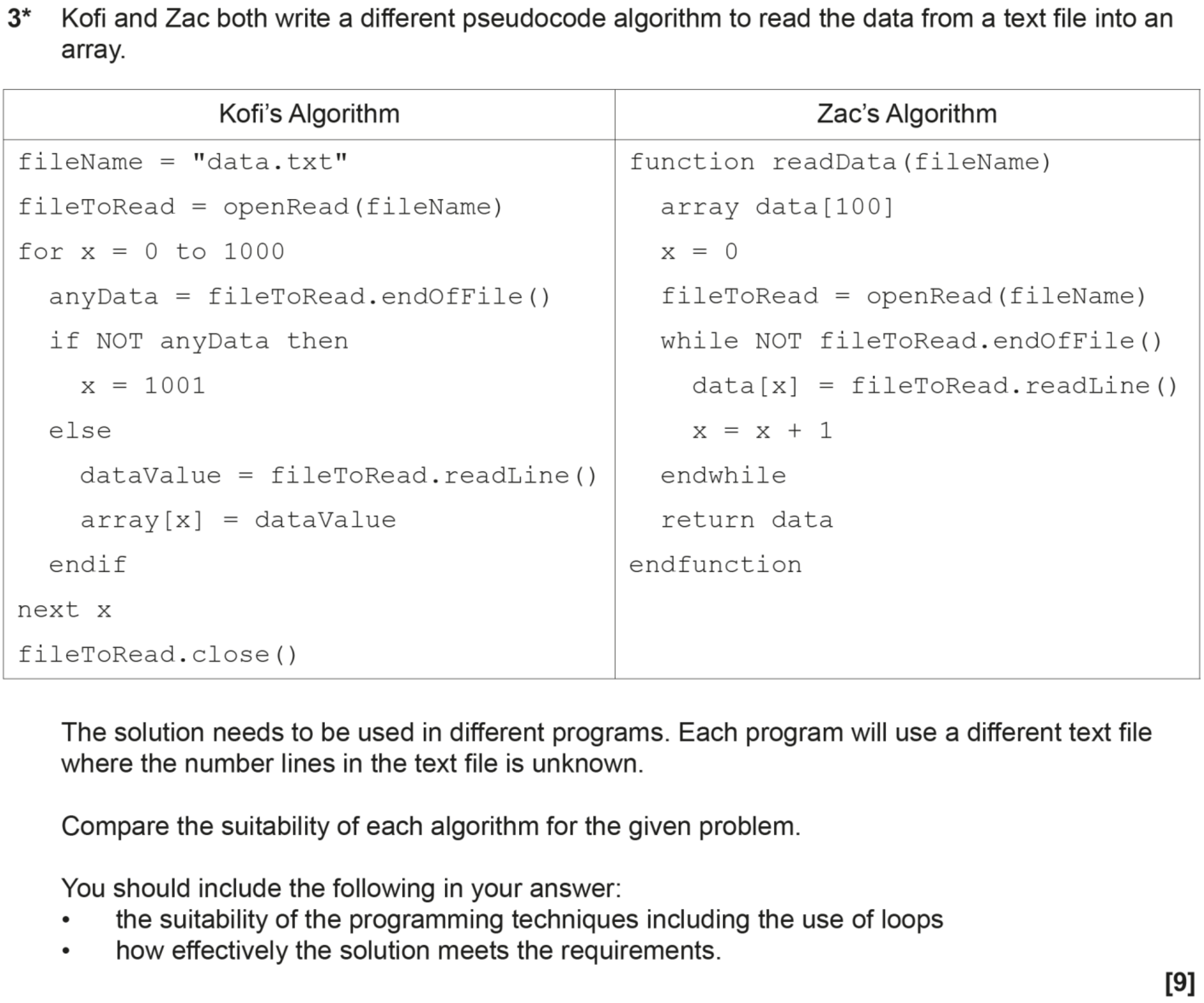
....

.... **[3]**

Some candidates mistakenly gave a definition for the dequeue() function instead of answering the question set. Candidates need to be mindful to read the question carefully.

Some candidates used a count-controlled loop whereas a conditional loop was required. Those who made a good attempt at a response often did not assign the result of `dequeue()` to a variable, so could not effectively use the return value from the function as well as print the value returned.

Question 3*



3* Kofi and Zac both write a different pseudocode algorithm to read the data from a text file into an array.

<table>
<tr><th>Kofi's Algorithm</th><th>Zac's Algorithm</th></tr>
<tr><td><pre>
fileName = "data.txt"
fileToRead = openRead(fileName)
for x = 0 to 1000
  anyData = fileToRead.endOfFile()
  if NOT anyData then
    x = 1001
  else
    dataValue = fileToRead.readLine()
    array[x] = dataValue
  endif
next x
fileToRead.close()
</pre></td><td><pre>
function readData(fileName)
  array data[100]
  x = 0
  fileToRead = openRead(fileName)
  while NOT fileToRead.endOfFile()
    data[x] = fileToRead.readLine()
    x = x + 1
  endwhile
  return data
endfunction
</pre></td></tr>
</table>

The solution needs to be used in different programs. Each program will use a different text file where the number lines in the text file is unknown.

Compare the suitability of each algorithm for the given problem.

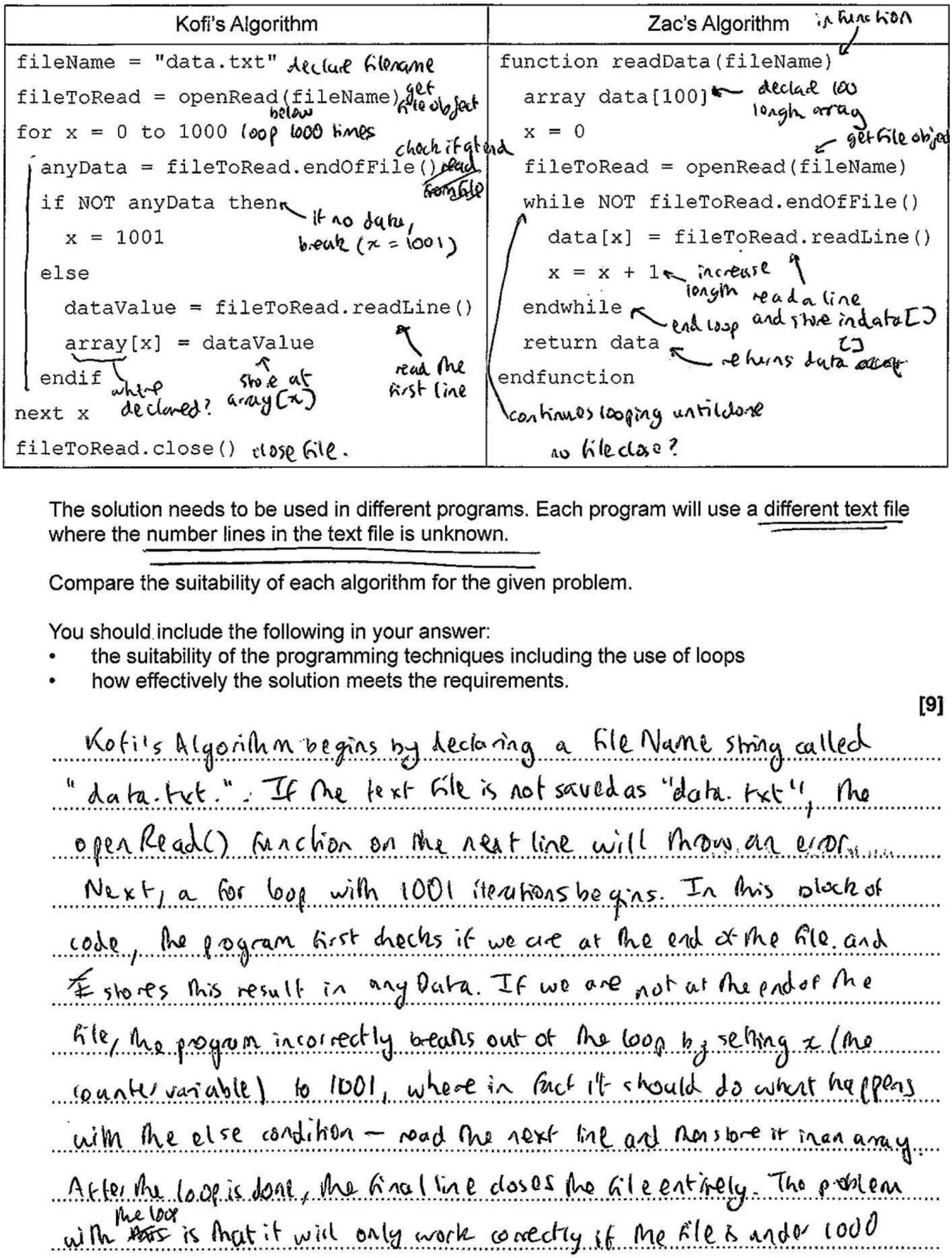
You should include the following in your answer:
- the suitability of the programming techniques including the use of loops
- how effectively the solution meets the requirements.

[9]

There were a large number of either blank or very brief responses where candidates displayed no ability to read and interpret code. The question source material was designed to help candidates to comment on a range of programming practices such as file opening and closing, conditional versus count-controlled loops and the benefits of the use of functions with parameters.

Level 1 responses often just focused on the use of two different loop types. There were few Level 3 responses that gave insightful evaluative comments. Where candidates were able to do so they often focused on memory usage or the ability to cope with different data sets that could be read.

Exemplar 2



Kofi's Algorithm

```
fileName = "data.txt"                  declare filename
fileToRead = openRead(fileName)        get file object
                                  below
for x = 0 to 1000                      loop 1000 times
  anyData = fileToRead.endOfFile()     check if at end ~~read from file~~
  if NOT anyData then                  If no data, break (x = 1001)
    x = 1001
  else
    dataValue = fileToRead.readLine()
    array[x] = dataValue               where declared?   store at array[x]   read the first line
  endif
next x
fileToRead.close()                     close file.
```

Zac's Algorithm in function

```
function readData(fileName)
  array data[100]                      declare 100 length array
  x = 0
  fileToRead = openRead(fileName)      get file object
  while NOT fileToRead.endOfFile()
    data[x] = fileToRead.readLine()    read a line and store in data[ ]
    x = x + 1                          increase length
  endwhile                             end loop
  return data                          returns data ~~array~~ [ ]
endfunction
continues looping until close
   no file close?
```

The solution needs to be used in different programs. Each program will use a different text file where the number lines in the text file is unknown.

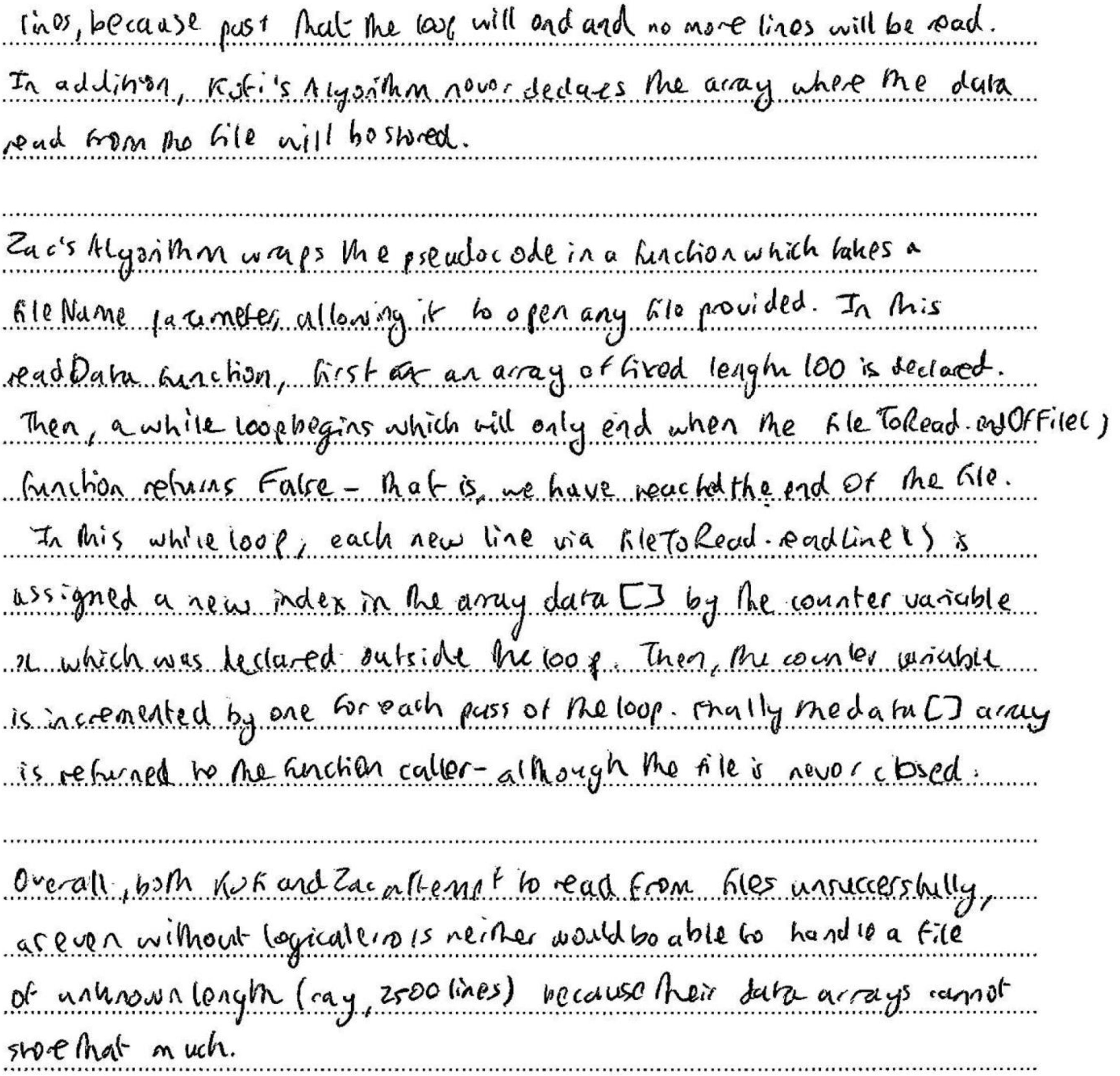
Compare the suitability of each algorithm for the given problem.

You should include the following in your answer:

- the suitability of the programming techniques including the use of loops
- how effectively the solution meets the requirements.

[9]

Kofi's Algorithm begins by declaring a file Name string called "data.txt.". If the text file is not saved as "data.txt", the openRead() function on the next line will throw an error. Next, a for loop with 1001 iterations begins. In this block of code, the program first checks if we are at the end of the file. and stores this result in anyData. If we are not at the end of the file, the program incorrectly breaks out of the loop by setting x (the counter variable) to 1001, where in fact it should do what happens with the else condition – read the next line and then store it in an array. After the loop is done, the final line closes the file entirely. The problem with ~~this~~ the loop is that it will only work correctly if the file is under 1000



lines, because past that the loop will end and no more lines will be read.
In addition, Kofi's Algorithm never declares the array where the data read from the file will be stored.

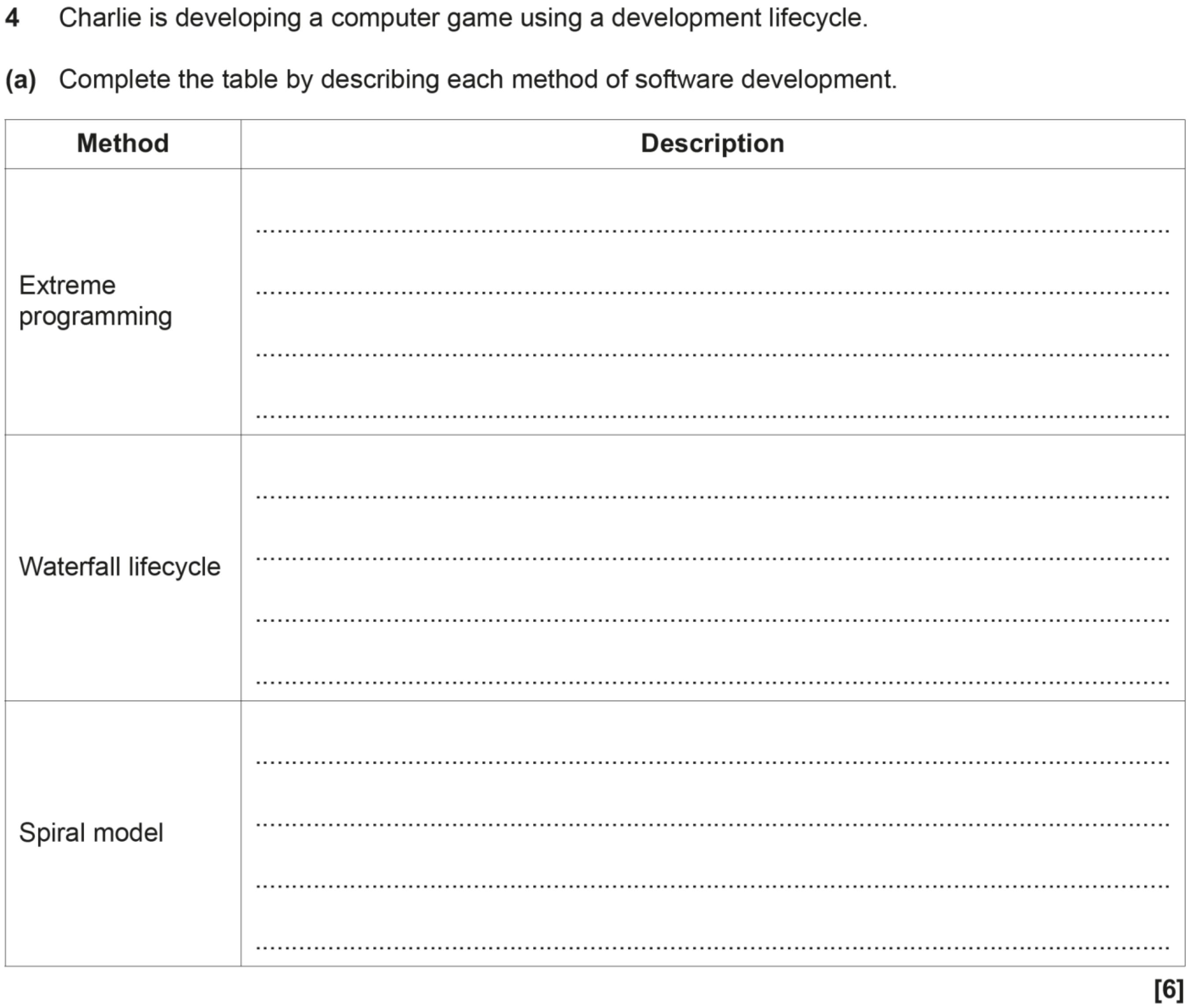
Zac's Algorithm wraps the pseudocode in a function which takes a fileName parameter, allowing it to open any file provided. In this readData function, first ~~a~~ an array of fixed length 100 is declared. Then, a while loop begins which will only end when the fileToRead.endOfFile() function returns False – that is, we have reached the end of the file. In this while loop, each new line via fileToRead.readLine() is assigned a new index in the array data[] by the counter variable x which was declared outside the loop. Then, the counter variable is incremented by one for each pass of the loop. Finally the data[] array is returned to the function caller – although the file is never closed.

Overall, both Kofi and Zac attempt to read from files unsuccessfully, as even without logical errors neither would be able to handle a file of unknown length (say, 2500 lines) because their data arrays cannot store that much.

This response showed that the candidate has thought about the issues related to the question and had carefully annotated the code when analysing it.

The response was well-structured taking each of the algorithms in turn and displayed a clear level of evaluative insight.

Question 4 (a)



4 Charlie is developing a computer game using a development lifecycle.

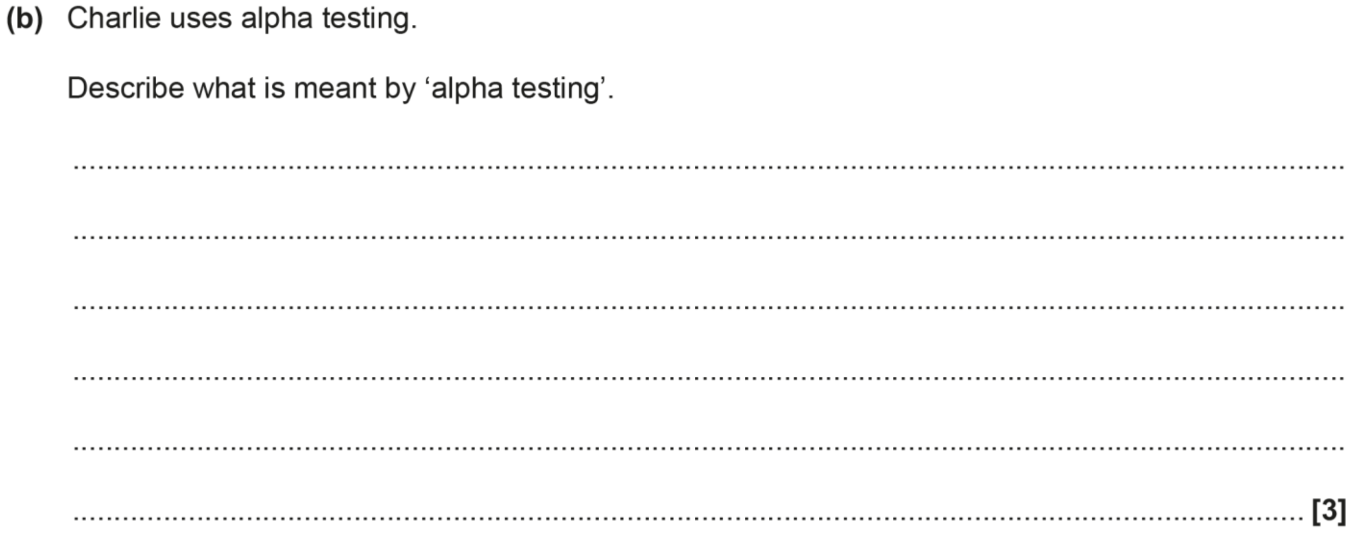
(a) Complete the table by describing each method of software development.

Method	**Description**
Extreme programming	....
Waterfall lifecycle	....
Spiral model	....

[6]

The topic of software design methodologies was one where candidates could have done well if they had learnt definitions and characteristics for each methodology. While many candidates displayed a very poor knowledge of this there were some more comprehensive responses. Extreme programming often focused on agile methodologies and paired programming. The Waterfall model elicited responses that indicated that it was a more linear model. For the Spiral model many successful responses identified that risk management was at the heart of the model.

Question 4 (b)



(b) Charlie uses alpha testing.

Describe what is meant by 'alpha testing'.

....

....

....

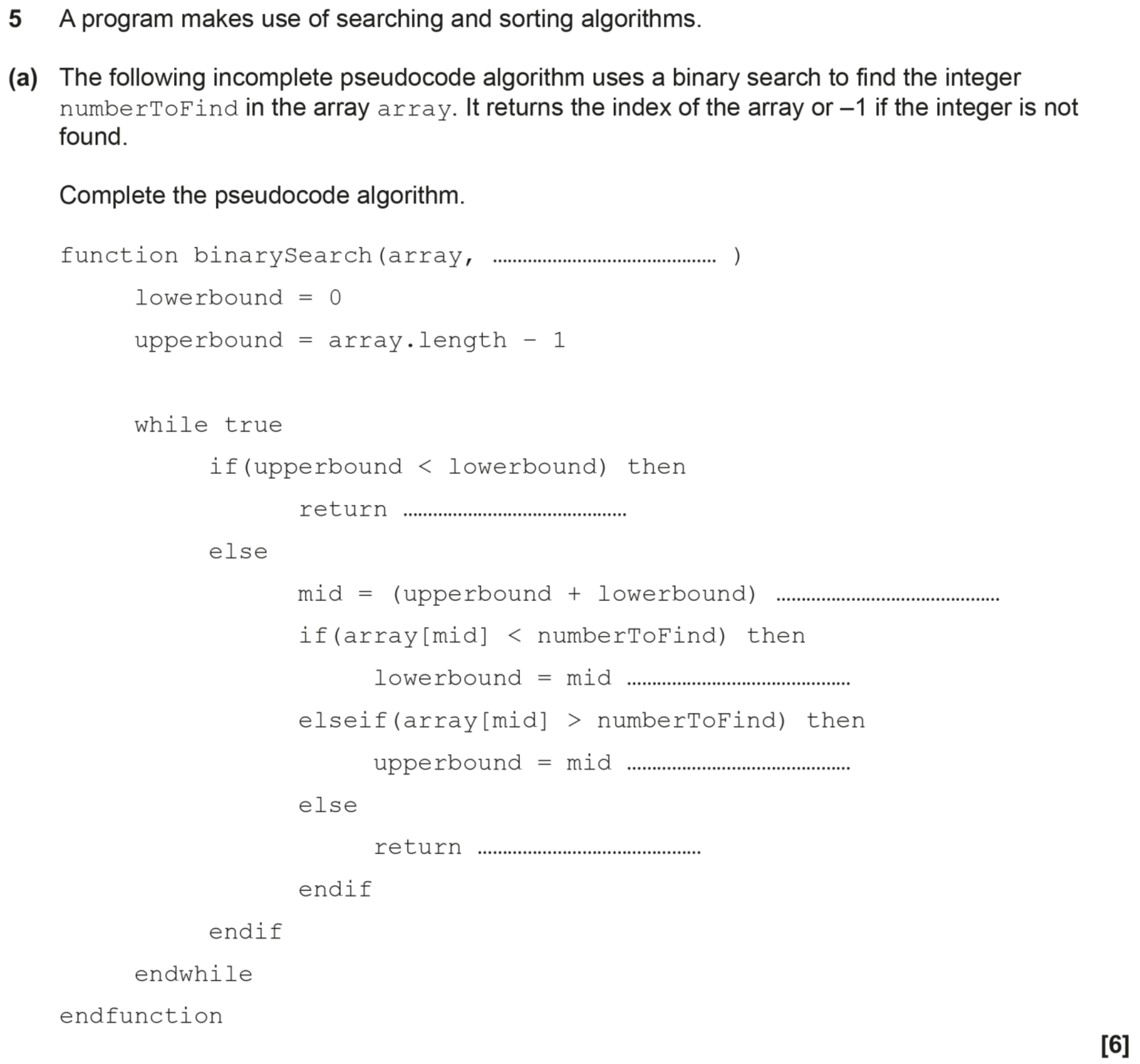
....

....

.... **[3]**

A number of candidates thought that alpha testing involved giving the software to end users to test rather than final tests in-house, confusing testing in a way that an end-user would use the software, with actually giving the software to end users. Most creditworthy responses focused on in-house testing by the development team or the place of alpha testing in the testing sequence, with it occurring before a more general beta testing phase.

Question 5 (a)



5 A program makes use of searching and sorting algorithms.

(a) The following incomplete pseudocode algorithm uses a binary search to find the integer `numberToFind` in the array `array`. It returns the index of the array or –1 if the integer is not found.

Complete the pseudocode algorithm.

```
function binarySearch(array, ............................................ )
     lowerbound = 0
     upperbound = array.length - 1

     while true
          if(upperbound < lowerbound) then
                return ............................................
          else
                mid = (upperbound + lowerbound) ............................................
                if(array[mid] < numberToFind) then
                     lowerbound = mid ............................................
                elseif(array[mid] > numberToFind) then
                     upperbound = mid ............................................
                else
                     return ............................................
                endif
          endif
     endwhile
endfunction
```

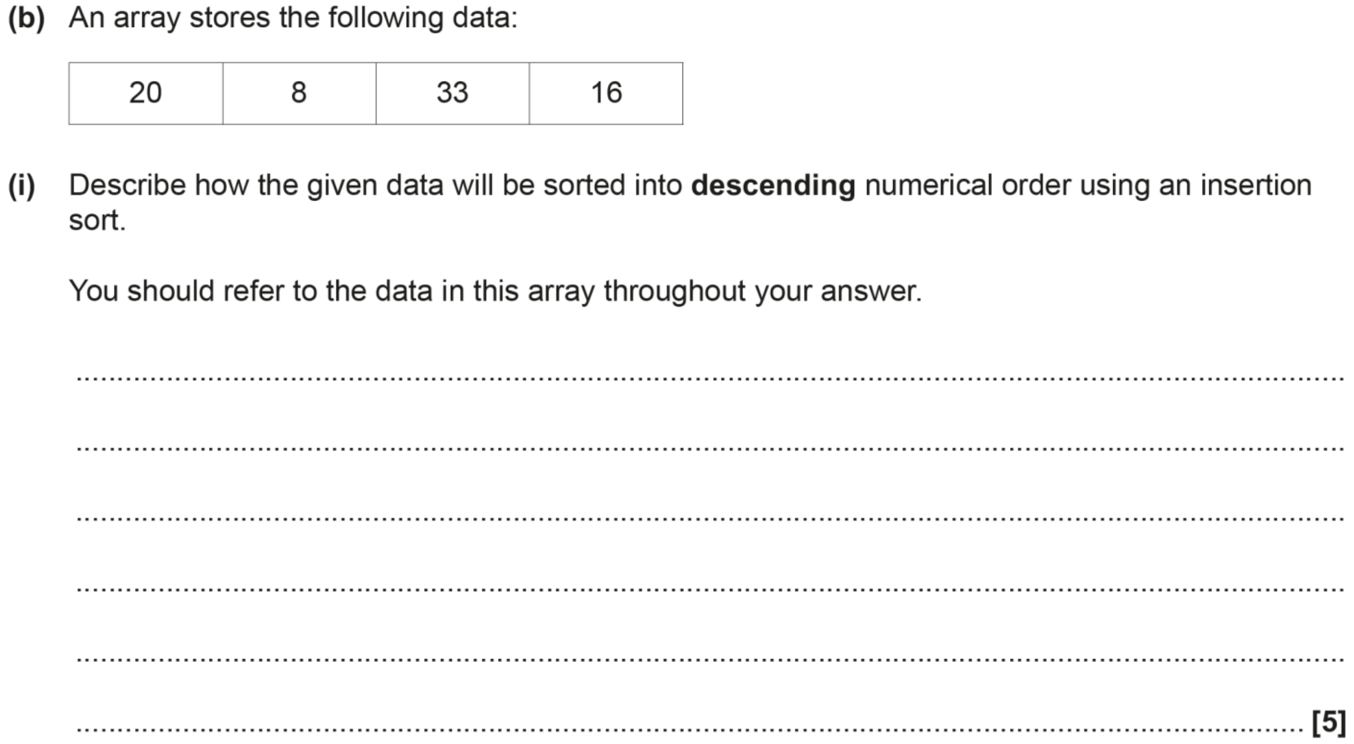
[6]

Binary search (in both iterative and recursive form) is a standard algorithm that candidates are expected to be able to code at AS Level. Candidates made a number of common errors including not using integer division (`DIV` and `//` were accepted) for the midpoint calculation.

Some candidates returned the value being searched for at the array index rather than returning mid as the index as specified in the question.

Another common error that was made was to update `mid` by the wrong value, swapping the +1 and -1 round.

Question 5 (b) (i)



(b) An array stores the following data:

20	8	33	16

(i) Describe how the given data will be sorted into **descending** numerical order using an insertion sort.

You should refer to the data in this array throughout your answer.

....

....

....

....

....

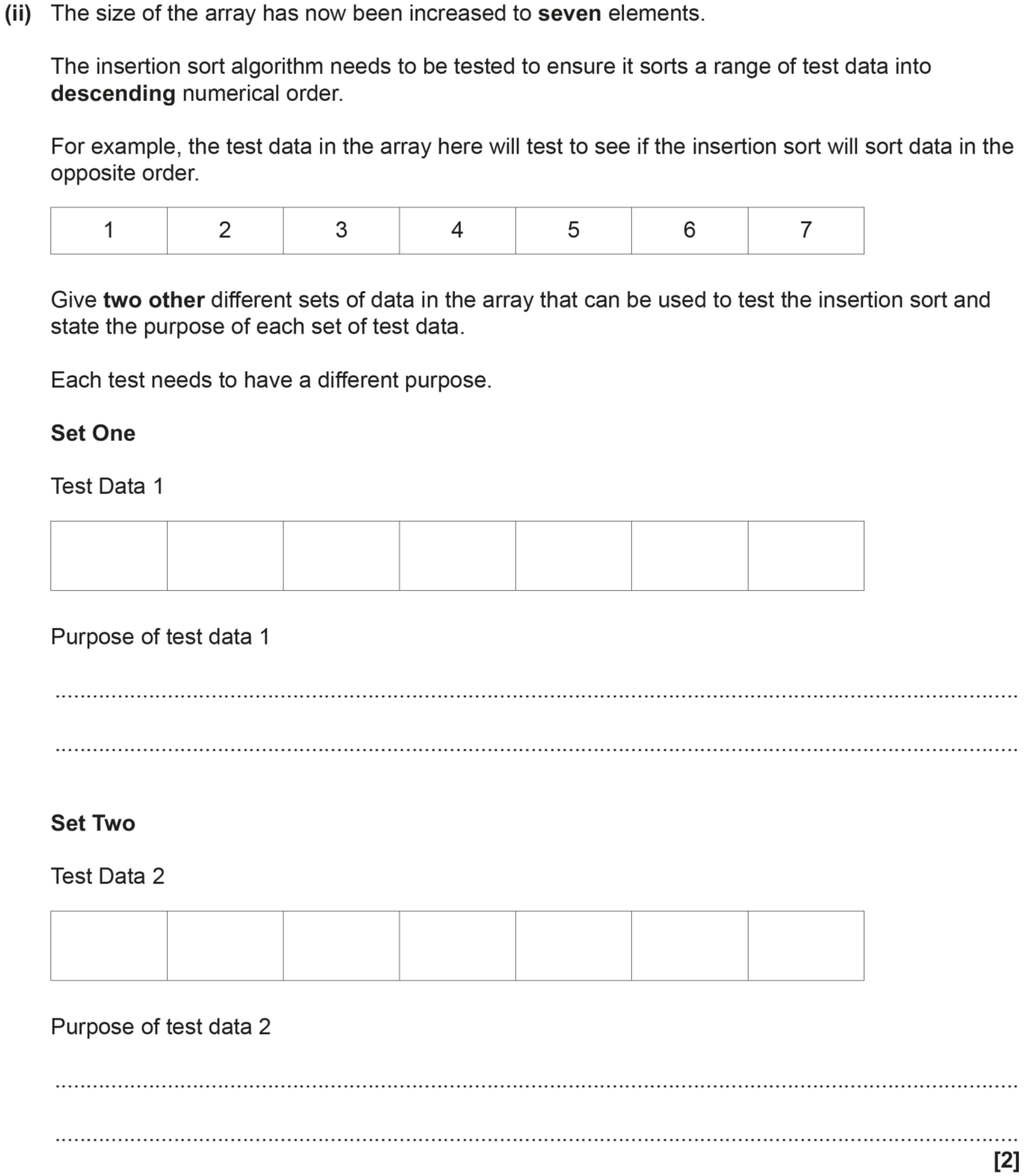
.... **[5]**

Insertion sort is one of the standard sorting algorithms that candidates need to be familiar with using. A pleasing number of candidates made a reasonable start to the response, but fewer could produce a comprehensive response.

The question required candidates to describe and not just show how insertion sort worked, so responses that only gave diagrams showing numbers moving with no annotation did not gain credit. Some candidates mistakenly described bubble sort or merge sort.

Quite often, where candidate scored four marks, they omitted the first pass of the sort where the first item (`20`) is set as the sorted list. Some potentially quite good responses were too vague as to how the next item at the start of the next pass filtered back through the sorted list to its correct position.

Question 5 (b) (ii)



(ii) The size of the array has now been increased to **seven** elements.

The insertion sort algorithm needs to be tested to ensure it sorts a range of test data into **descending** numerical order.

For example, the test data in the array here will test to see if the insertion sort will sort data in the opposite order.

1	2	3	4	5	6	7

Give **two other** different sets of data in the array that can be used to test the insertion sort and state the purpose of each set of test data.

Each test needs to have a different purpose.

Set One

Test Data 1

Purpose of test data 1

....

....

Set Two

Test Data 2

Purpose of test data 2

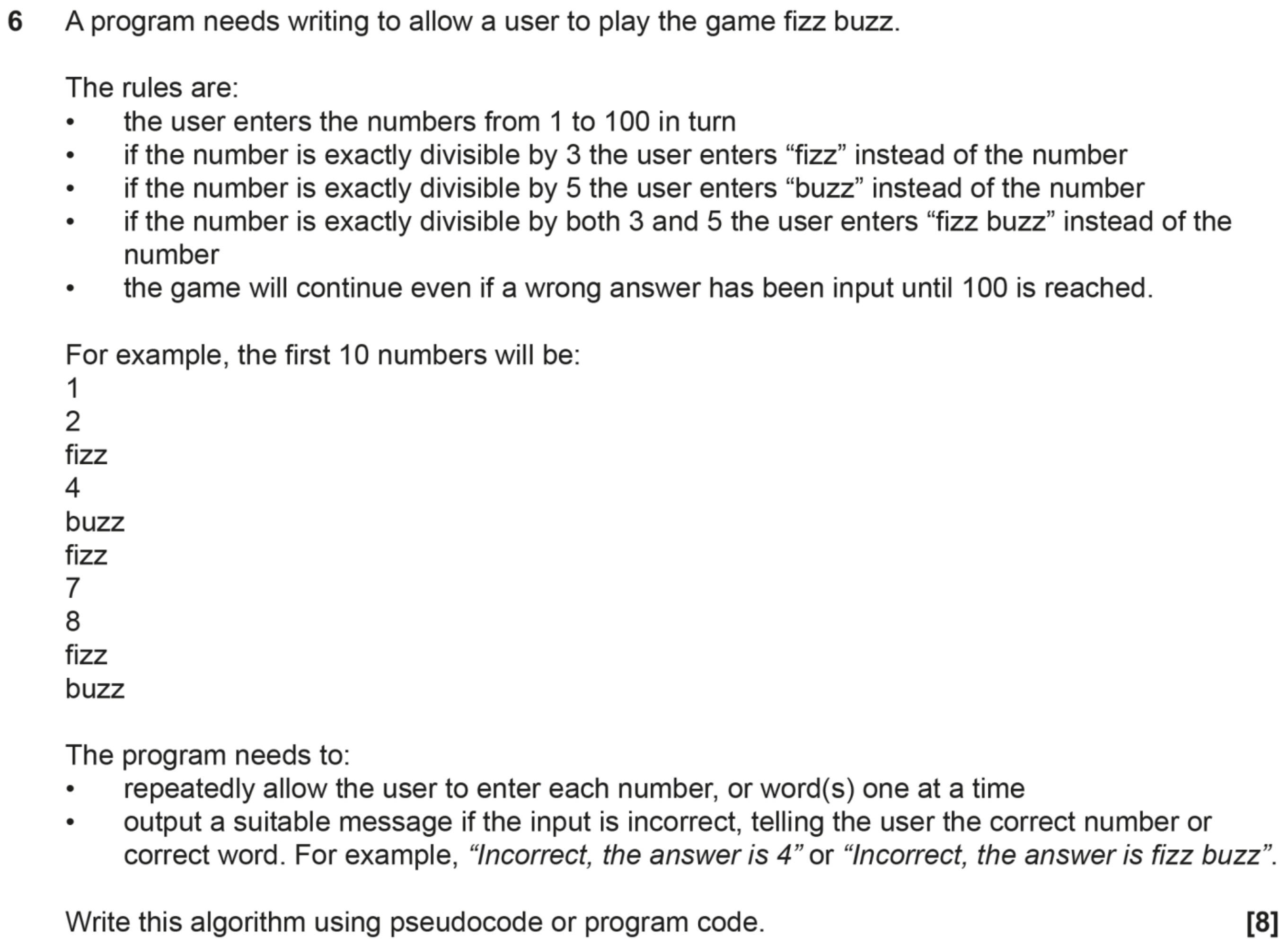
....

....

[2]

Many candidates gained some credit for giving a valid reason for a set of test data that they presented. Clear responses included testing decimals, negatives, different data types, and testing on data that was not already in order.

Question 6



6 A program needs writing to allow a user to play the game fizz buzz.

The rules are:

- the user enters the numbers from 1 to 100 in turn
- if the number is exactly divisible by 3 the user enters "fizz" instead of the number
- if the number is exactly divisible by 5 the user enters "buzz" instead of the number
- if the number is exactly divisible by both 3 and 5 the user enters "fizz buzz" instead of the number
- the game will continue even if a wrong answer has been input until 100 is reached.

For example, the first 10 numbers will be:

1
2
fizz
4
buzz
fizz
7
8
fizz
buzz

The program needs to:

- repeatedly allow the user to enter each number, or word(s) one at a time
- output a suitable message if the input is incorrect, telling the user the correct number or correct word. For example, *"Incorrect, the answer is 4"* or *"Incorrect, the answer is fizz buzz"*.

Write this algorithm using pseudocode or program code. **[8]**

While candidates tended to struggle with longer pseudocode questions there was evidence of an improvement in the overall level this year. It is possible that was due to familiarity with a commonly programmed task.

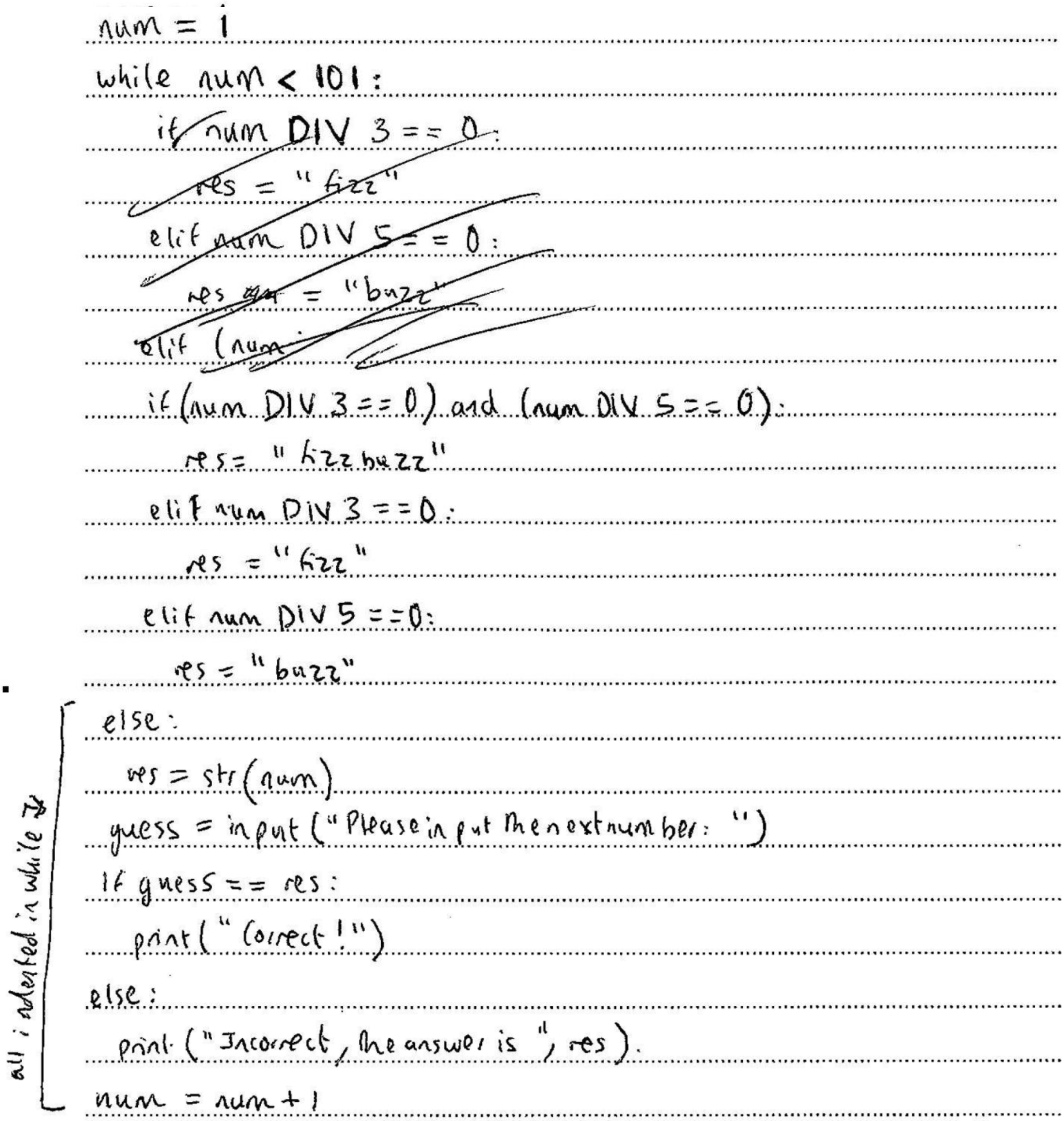
Some common errors included not using strings and thus confusing the string "fizz" with `fizz` as an identifier. Candidates frequently used the user input to perform the calculation for fizz / buzz / fizzbuzz on rather than the loop variable. There were also logical errors such as not checking the fizzbuzz case before then checking fizz and buzz.

There were some particularly good solutions that were implemented in a number of interesting and efficient ways too.

Assessment for learning

It is always beneficial to get candidates to try to program solutions to questions that require pseudocode responses. This will help candidates to test and refine solutions and will help to identify potential misconceptions and logical errors.

Exemplar 3



```
num = 1
while num < 101:
    ~~if num DIV 3 == 0:~~
        ~~res = "fizz"~~
    ~~elif num DIV 5 == 0:~~
        ~~res = "buzz"~~
    ~~elif (num~~
    if (num DIV 3 == 0) and (num DIV 5 == 0):
        res = "fizzbuzz"
    elif num DIV 3 == 0:
        res = "fizz"
    elif num DIV 5 ==0:
        res = "buzz"
else:
    res = str(num)
guess = input("Please input the next number: ")
if guess == res:
    print("Correct!")
else:
    print("Incorrect, the answer is ", res).
num = num + 1
```

all indented in while

This response presented a clear well-structured solution that covered each of the require fizzbuzz, fizz and buzz cases as well as remembering to type cast when comparing the user (string) input to the loop counter (integer) value.

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 a free test-building platform, providing unlimited users exclusively for staff at OCR centres with an Interchange account.

Choose from a large bank of questions to build personalised tests and custom mark schemes, with the option to add custom cover pages to simulate real examinations. You can also edit and download complete past papers.

Find out more.

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.

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

Need to get in touch?

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

Call us on
01223 553998

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

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

We really value your feedback

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

👍 **I like this** 👎 **I dislike this**

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

CAMBRIDGE
UNIVERSITY PRESS & ASSESSMENT

OCR acknowledges the use of the following content: N/A

Whether you already offer OCR qualifications, are new to OCR or are thinking about switching, you can request more information using our Expression of Interest form.

Please get in touch if you want to discuss the accessibility of resources we offer to support you in delivering our qualifications.