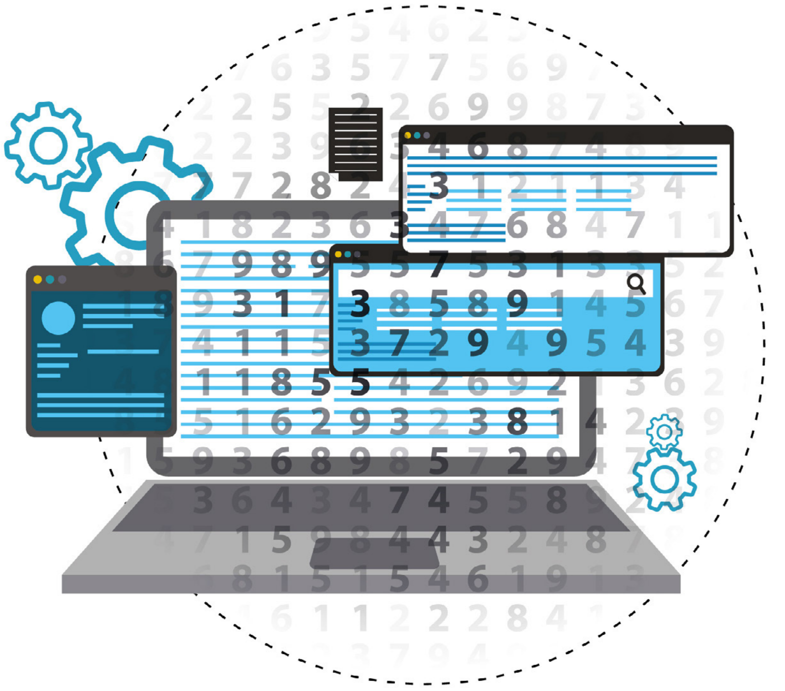
Cambridge Assessment International Education

Specimen Paper Answers – Paper 2

Cambridge International AS & A Level

Computer Science 9618

For examination from 2021



Version 2

Cambridge Pathway

Copyright © UCLES July 2019 (updated December 2021)
Cambridge Assessment International Education is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

UCLES retains the copyright on all its publications. Registered Centres are permitted to copy material from this booklet for their own internal use. However, we cannot give permission to Centres to photocopy any material that is acknowledged to a third party, even for internal use within a Centre.

Contents

Introduction

The main aim of this booklet is to exemplify standards for those teaching Cambridge International AS & A Level Computer Science 9618, Paper 2: Fundamental Problem-solving and Programming Skills, and to show examples of very good answers. The questions are from the published Specimen Paper, which contains seven questions, each consisting of several parts. There are no optional questions. The format of the Specimen Paper consists of questions and related sub-questions. This may be taken as a guide to actual examination question papers, although the actual number of questions and sub-questions may alter from series to series.

In this booklet, we have provided answers for all questions in the Specimen Paper.

Each question is followed by an example of a high-grade answer with an examiner comment on performance. Comments are given to indicate where and why marks were awarded, and how additional marks could have been obtained. In this way, it is possible to understand what candidates have done to gain their marks and how they could improve.

The mark schemes for the Specimen Papers are available to download from the School Support Hub at www.cambridgeinternational.org/support

2021 Specimen Paper 2 Mark Scheme

Past exam resources and other teacher support materials are available on the School Support Hub www.cambridgeinternational.org/support

Assessment overview

Paper 2 – Fundamental Problem-solving and Programming Skills

Written paper, 2 hours, 75 marks

Externally assessed

Paper 2 will assess sections 9 to 12 of the syllabus content.

Candidates will need to write answers in pseudocode.

Candidates answer all questions.

50% of the AS Level

25% of the A Level

Many questions require a solution to a problem, which will require the application of problem-solving and programming skills. Due to the nature of the subject it is likely that there will be several different solutions. In these cases the mark schemes will reflect the essential components of the solution rather than reference one specific method.

When writing pseudocode answers, candidates should limit themselves to the list of functions and operators that are provided in the insert. Whilst it may be tempting to make use of functions or methods that are available in a certain programming language to provide an easy solution to a given problem, this could disadvantage those candidates who have not studied that particular language. By using only the defined set of functions and operators all candidates are given an equal opportunity to provide a correct answer.

Centres are referred to the published pseudocode guide for further information.

Assessment objectives

The assessment objectives (AOs) are:

AO2: Apply knowledge and understanding of the principles and concepts of computer science, including to analyse problems in computational terms.

AO3: Design, program and evaluate computer systems to solve problems, making reasoned judgements about these.

Question 1

Question 1 (a) (i)

1 **(a)** Program variables have values as follows:

Variable	**Value**
`Today`	`"Tuesday"`
`WeekNumber`	`37`
`Revision`	`'C'`
`MaxWeight`	`60.5`
`LastBatch`	`TRUE`

(i) Give an appropriate data type for each variable.

Variable	**Data type**
`Today`	
`WeekNumber`	
`Revision`	
`MaxWeight`	
`LastBatch`	

[5]

Specimen Paper Response

Variable	**Data type**
`Today`	STRING
`WeekNumber`	INTEGER
`Revision`	CHAR
`MaxWeight`	REAL
`LastBatch`	BOOLEAN

Examiner comment:

The given values in the table in part (a) are interpreted as follows:

- `"Tuesday"` is a literal string enclosed by quotation marks and so is of data type STRING
- `37` is a numeric value without decimal places so is of data type INTEGER
- `'C'` is a single character enclosed by quotation marks so is of data type CHAR
- `60.5` is a numeric value with decimal places so is of data type REAL
- `TRUE` indicates a Boolean value so the data type is BOOLEAN

Total mark awarded = 5 out of 5

Question 1 (a) (ii)

(ii) Evaluate each expression in the following table.
If an expression is invalid then write ERROR.

Refer to the **Insert** for the list of pseudocode functions and operators.

Expression	Evaluates to
`MID(Today, 3, 2) & Revision & "ape"`	
`INT(MaxWeight + 4.2)`	
`LENGTH(MaxWeight)`	
`MOD(WeekNumber, 12)`	
`(Revision <= 'D') AND (NOT LastBatch)`	

[5]

Specimen Paper Response

Expression	Evaluates to
`MID(Today, 3, 2) & Revision & "ape"`	"esCape"
`INT(MaxWeight + 4.2)`	64
`LENGTH(MaxWeight)`	ERROR
`MOD(WeekNumber, 12)`	1
`(Revision <= 'D') AND (NOT LastBatch)`	FALSE

Examiner comment:

The expressions are evaluated as follows:

Row 1
Variable `Today` contains the string `"Tuesday"`. The `MID()` function extracts 2 characters from this string starting from position 3 (i.e. characters 3 and 4) which gives the string `"es"`. Variable `Revision` contains the character `'C'`, so the expression simplifies to:

```
"es" & 'C' & "ape"
```

The `&` represents the concatenation operator, so this expression evaluates to the string `"esCape"`
Quotation marks must be used to indicate that the value represents a literal string and not an identifier name. A common mistake is for these to be omitted.

Row 2
Variable `MaxWeight` has the value `60.5`. Adding this to `4.2` gives `64.7`. The `INT()` function returns the `INTEGER` value of the parameter (in this case `64.7`) , which is `64`.

Row 3
The `LENGTH` function expects a parameter of type `STRING`. In this expression a variable of type `REAL` has been used, hence the expression is invalid and `ERROR` should be written, as directed in the question.

Row 4
Variable `WeekNumber` contains the value `37`. The `MOD()` function returns an `INTEGER` value representing the remainder when one number is divided by another using integer arithmetic. In this case `37` divided by `12` gives a remainder of `1`.

Row 5
Variable Revision contains the character `'C'` which is less than character `'D'` so the contents of the first bracket evaluates to `TRUE`. Variable `LastBatch` contains the value `TRUE` so the contents of the second bracket evaluates to `FALSE`. This gives the simplified expression:

```
TRUE AND FALSE
```

This in turn evaluates to `FALSE`.

Total mark awarded = 5 out of 5

Question 1 (b)

(b) Simple algorithms usually consist of input, process and output.

Complete the table to show if each statement is an example of input, process or output. Place one or more ticks (✓) for each statement.

Item	Statement	Input	Process	Output
1	`SomeChars ← "Hello World"`			
2	`OUTPUT RIGHT(SomeChars, 5)`			
3	`READFILE MyFile, MyChars`			
4	`WRITEFILE MyFile, "Data is " & MyChars`			

[4]

Specimen Paper Response

Item	Statement	Input	Process	Output
1	`SomeChars ← "Hello World"`		✓	
2	`OUTPUT RIGHT(SomeChars,5)`		✓	✓
3	`READFILE MyFile, MyChars`	✓	✓	
4	`WRITEFILE MyFile, "Data is " & MyChars`		✓	✓

Examiner comment:

The table has been completed correctly, as follows:

Row 1
The assignment of the literal string value to the identifier is an example of a Process

Row 2
The statement involves generating a sub-string by extracting 5 characters from the identifier `SomeChars` (the Process) and outputting the resulting sub-string (the Output)

Row 3
The statement reads a line from the file `MyFile` (the Input) and assigns the value to the identifier `Mychars` (the Process).

Row 4
The statement writes a line to the file `MyFile` (the Output) after first forming a string by concatenating (the Process) the literal string `"Data is "` and the value of identifier `MyChars`.

Total mark awarded = 4 out of 4

Question 1 (c)

(c) Write in pseudocode a **post-condition loop** to output all the odd numbers between 100 and 200.

....

....

....

....

....

....

....

....

.... [4]

Specimen Paper Response

```
MyCount ← 101
REPEAT
   OUTPUT MyCount
   MyCount ← MyCount + 2
UNTIL MyCount = 199
```

Examiner comment:

The question specifies a post-condition loop, which implies a `REPEAT ... UNTIL` construct. The loop termination condition should be based on the final value required (200). Prior to the loop, the start value should be assigned to a variable and this variable will be increased by two each time around the loop. Note that the start value given in the question is not an odd number and so should not be output.

The answer above could be considered to be an "optimised" solution in the sense that the learner has taken into account the fact that the start value given in the question is **not** an odd number and modified the initial value accordingly. They have also realised that adding 2 to an odd number will give the *next* odd number.

A mistake has been made with the terminating condition, meaning that the final value output would be 197 rather than 199, losing one mark.

An alternative solution could be as follows:

```
MyCount ← 100
REPEAT
   IF MOD(MyCount, 2) = 1 // Check for a remainder of 1 when divided by 2 i.e. an
odd number
      THEN
         OUTPUT MyCount
   ENDIF
   MyCount ← MyCount + 1
UNTIL MyCount = 200
```

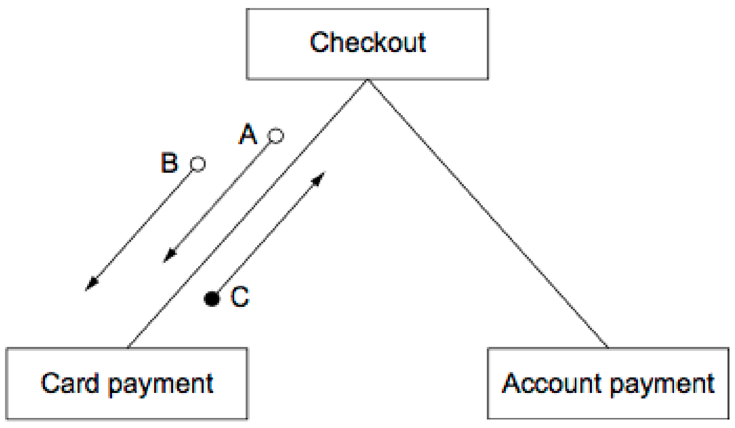
Note that this alternative solution includes a correct terminating condition and would gain full marks.

Total mark awarded = 3 out of 4

Question 2

Question 2 (a) (i)

2 Roberta downloads music from an online music store. The diagram shows part of a structure chart for the online music store program.



(a) State **three** items of information that the diagram shows about the design of the program.

1 ....

....

2 ....

....

3 ....

....
[3]

Specimen Paper Response

1 The names of the modules

2 The relationship between the modules

3 The parameters passed between the modules

Examiner comment:

The labelled rectangles represent three program modules. The lines from module Checkout to the other two modules at the lower level indicate that these two are sub-modules that are called from within Checkout. The labelled arrows indicate the parameters that are passed between modules; the shaded circle of parameter C indicates a Boolean value.

The answer correctly identifies three items of information.

In this case, an alternative mark could have been obtained by reference to the module sequence. For example by answering:

"Module sequence – Checkout calls Card payment and then Account payment"

It is important that candidates read questions carefully to be clear about the scope of the question being asked. In this case the question refers to the diagram given and not structure charts in general. Consequently, only features that are present in the given diagram will be awarded a mark. For example, "iteration" and "selection" are features that may be included in a structure chart but they would not be acceptable answers as the symbols for these are not included in the given diagram.

Total mark awarded = 3 out of 3

Question 2 (b)

(b) Examples of the data items that correspond to the arrows are given in the table.

Arrow	**Data item**
A	`234.56`
B	`"Ms Roberta Smith"`
C	`TRUE`

Use pseudocode to write the function header for the Card payment module.

....

.... [3]

Specimen Paper Response

FUNCTION CardPayment (Amount : REAL, Name : STRING) RETURNS BOOLEAN

Examiner comment:

The example values given for the Data Items indicate the required data types for the parameters. The single-headed arrows used in the diagram for parameters A and B indicate that these parameters are passed by value and not by reference. This is the default mechanism and does not need to be explicitly stated in the function declaration, but use of the `BYVALUE` keyword would obviously be correct.

The names given to parameters A and B are not important although they must be present and unique and given the example data items it is expected that they would be given appropriate identifier names. The parameter order for A and B in the function header is not important.

Total mark awarded = 3 out of 3

Question 3

Question 3 (a)

3 A stack is created using a high-level language. The following diagram represents the current state of the stack. The Top of Stack pointer points to the last item added to the stack.

Address	**Value**	**Pointer**
99		
100		
101	E	⇐ TopOfStack
102	D	
103	C	
104	B	
105	A	

(a) Two operations associated with this stack are `PUSH()` and `POP()`.

Describe these operations with reference to the diagram.

`MyVar = POP()`

....

....

....

`PUSH('Z')`

....

....

....
[4]

Specimen Paper Response

`MyVar = POP`

The value 'E' is removed from the stack
The TopofStack pointer is then incremented to point to location 102

`PUSH('Z')`

The TopOfStack pointer is decremented to point to location 101
Value 'Z' is loaded into location 101

Examiner comment:

There are several different possible ways in which stack operations can function. For clarity, this question states that the `TopOfStack` pointer points to the last value added to the stack, in this case the value `E` in location `101`. Locations `102` to `105` are shown to contain values whereas locations `100` and `99` are shown as 'empty'. This should be sufficient to indicate to learners that the stack in this case will grow 'upwards' through decreasing address locations. In this case a 'Push' operation would be pre-decrement operation but this terminology is not something learners would be expected to be familiar with.

The answer given fully describes the two operations.

It is intended that the two operations described in the question be applied in sequence to the stack as shown. If the answer given to the first part is incorrect, then whatever value had been assigned to the `TopOfStack` would be carried forward when marking the second part of the question.

Total mark awarded = 4 out of 4

Question 3 (b)

(b) Two programs use a stack to exchange data. Program `AddString` pushes a string of characters onto the stack one character at a time. Program `RemoveString` pops the same number of characters off the stack, one character at a time. The string taken off the stack is different from the string put on the stack.

Explain why the strings are different.

....

....

....

.... [2]

Specimen Paper Response

The strings are different because one is the reverse of the other. e.g. "CAT" becomes "TAC"

This is because a stack operates as a First-In-Last-Out (FILO) mechanism

Examiner comment:

The answer describes the difference between the two strings - that is the strings are in reverse order. In this case the answer also includes an example, which in itself would be sufficient to gain the first mark.

The answer explains that the reason for this is that a stack operates as a FILO (or LIFO) mechanism.

Total mark awarded = 2 out of 2

Question 4

Question 4 (a)

4 **(a)** Parameter `x` is used to pass data to procedure `MyProc` in the following pseudocode:

```
x ← 4
CALL MyProc(x)
OUTPUT x
```

```
PROCEDURE MyProc(x : INTEGER)
    DECLARE z : INTEGER
    x ← x + 1
    z ← x + 3
ENDPROCEDURE
```

There are two parameter passing methods that could be used.

Complete the following table for each of the two methods.

Name of parameter passing method	Value output	Explanation

[6]

Specimen Paper Response

Name of parameter passing method	Value Output	Explanation
Call by reference	*5*	*A pointer to the variable is passed. Original variable is changed when parameter changed in MyProc.*
Call by value	*4*	*A copy of the variable is passed. Original variable not changed when parameter changed in Myproc.*

Examiner comment:

In the first two columns the answer correctly names the two possible methods and gives the corresponding value output in each case. The explanation section requires a more in-depth answer and here the answer describes the mechanism in each case and also the effect this has on the parameter that is passed.

For each method, a mark is awarded for the name plus the corresponding value output, and two marks for the explanation.

It is important that the explanation is sufficiently detailed and unambiguous. For example, the statement *"the value is not changed"* would not be given a mark as it is not clear which value is being referred to.

Total mark awarded = 6 out of 6

Question 4 (b)

(b) The pseudocode includes the use of parameters.

State **two** other features in the pseudocode that support a modular approach to programming.

1 ....

....

2 ....

....
[2]

Specimen Paper Response

Procedures

Local variables

Examiner comment:

The answer correctly identifies two features that are contained in the example. In this case the word 'subroutine' would have been an acceptable alternative to 'procedure'.

Total mark awarded = 2 out of 2

Question 5

Question 5 (a)

5 A company keeps details of its product items in a 1D array, `Stock`. The array consists of 1000 elements of type `StockItem`.

The record fields of `StockItem` are:

Field	**Typical value**
`ProductCode`	`"BGR24-C"`
`Price`	`102.76`
`NumberInStock`	`15`

(a) Write pseudocode to declare the record structure `StockItem`.

....

....

....

....

.... [3]

Specimen Paper Response

```
TYPE Stock
   DECLARE ProductCode : STRING
   DECLARE Price : REAL
   DECLARE NumberInStock : INTEGER
END
```

Examiner comment:

The answer requires the definition of a user-defined record type named `StockItem` that includes the three fields as required. In the answer above, the data type for each field have been correctly inferred from the typical values given, but the record type has been named "Stock", rather than "StockItem"

Total mark awarded = 2 out of 3

Question 5 (b)

(b) Write pseudocode to declare the `Stock` array.

....

.... [3]

Specimen Paper Response

```
DECLARE StockItem : ARRAY [1:1000] OF StockItem
```

Examiner comment:

The array declaration includes the array size and the data type, but the name of the array has been given as 'StockItem" rather than "Stock".

Total mark awarded = 2 out of 3

Question 5 (c)

(c) Write pseudocode to modify the values to element 20 as follows:

- set the price to 105.99
- increase the number in stock by 12

....

....

....

.... [2]

Specimen Paper Response

```
Stock[20].Price ← 105.99
Stock[20].NumberInStock ← Stock[20].NumberInStock + 12
```

Examiner comment:

The answer correctly demonstrates how individual data items are accessed using the dot notation.

Total mark awarded = 2 out of 2

Question 5 (d)

(d) A stock report program is developed.

Write pseudocode to output the information for each stock item that has a price of at least 100.

Output the information as follows:

```
Product Code: BGR24-C Number in Stock: 15
```

....

....

....

....

....

....

....

.... [4]

Specimen Paper Response

```
DECLARE n : INTEGER
FOR n ← 1 to 1000
   IF Stock[n].Price >= 100
      THEN
         OUTPUT "ProductCode: " & Stock[n].ProductCode  & Number in Stock:  &
                      Stock[n].NumberInStock
    ENDIF
NEXT
```

Examiner comment:

The answer contains one error, but otherwise correctly implements the simple algorithm, the essential elements of which are as follows:

- A loop to process all the elements in the array
- A conditional statement to check the price, correctly translating the phrase 'at least' into the comparison '>='. Note that the expression '> 99' would not be valid in this case as the numbers are not integer values. For example, information would be output for a stock item with a price of 99.5, which is not what is required.
- An output statement that includes the required field values and is formatted as required in the question, but the necessary quotation marks around the literal string "Number in Stock: " have been omitted.

The answer provides simple 'in-line' pseudocode as the question does not ask for the solution to be provided as either a procedure or a function.

Total mark awarded = 3 out of 4

Question 6

Question 6 (a)

6 Members of a family use the same laptop computer. Each family member has their own password.

To be valid, a password must comply with the following rules:

1 At least two lower-case alphabetic characters
2 At least two upper-case alphabetic characters
3 At least three numeric characters
4 Alpha-numeric characters only

A function, `ValidatePassword`, is needed to check that a given password follows these rules. This function takes a string, `Pass`, as a parameter and returns a boolean value:

- TRUE if it is a valid password
- FALSE otherwise

(a) Write pseudocode to implement the function `ValidatePassword`.

Refer to the **Insert** for the list of pseudocode functions and operators.

....

....

....

....

....

....

....

....

.... [9]

Specimen Paper Response

```
FUNCTION ValidatePassword(Pass : STRING) RETURNS BOOLEAN
DECLARE LCaseChar, UCaseChar, NumChar, n : INTEGER
DECLARE NextChar : CHAR
DECLARE ReturnFlag : BOOLEAN
ReturnFlag ← TRUE
LCaseChar ← 0
UCaseChar ← 0
NumChar ← 0
n ← 1
```

```
WHILE n < LENGTH(Pass) AND ReturnFlag = TRUE
   NextChar ← MID(Pass, n, 1)
   IF NextChar >= 'a' AND NextChar <= 'z'
      THEN
         LCaseChar ← LCaseChar + 1
      ELSE
         IF NextChar >= 'A' AND NextChar <= 'Z'
            THEN
               UCaseChar ← UCaseChar + 1
            ELSE
              IF NextChar >= '0' AND NextChar <= '9'
                 THEN
                    NumChar ← NumChar + 1
                 ELSE
                    ReturnFlag ← False    //illegal character
              ENDIF
        ENDIF
   n ← n + 1
NEXT

IF NOT (LCaseChar > 1 AND UCaseChar > 1 AND NumChar > 2 AND ReturnFlag)
   THEN
       ReturnFlag ← FALSE
ENDIF
OUTPUT ReturnFlag
ENDFUNCTION
```

Examiner comment:

There are many ways of solving this problem – the answer given represents a typical 'straightforward' solution. There are opportunities for simplification and a solution based on a `CASE` construct may provide a more elegant solution but neither of these is a requirement. A functionally correct solution will gain full marks.

Following the function header and local variable declarations and initialisation, the solution breaks down into two parts:

A loop is used. Within this loop each character of the given password string is extracted in turn. The character is compared with the three ranges described by each of the first three rules in the question, and a separate count is made in the number of characters in each range. If the character is not one of the three types specified then it is an illegal character and a flag is set. In the answer above, the loop is terminated as soon as an illegal character is encountered. This is not essential to the solution but would be expected from a top-grade answer.

Following the loop, the four rules given are tested via an `IF` statement. This assigns a Boolean value to the variable, which is then returned.

The answer contains three errors as follows:

- The loop does not check the final character of the string. The condition should be:
  `n <= LENGTH(Pass)`

- There is a missing `ENDIF`. (The first `IF` clause is unterminated.)

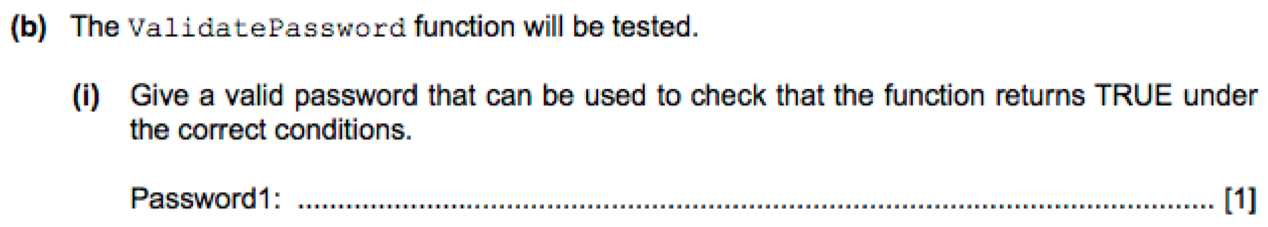
- The boolean value is output rather than returned

The available mark scheme points are shown below. For this answer, marks have been awarded for the points that have not been crossed through. (Note that the question is for a maximum of 9 marks, but there are 10 possible mark points)

1. Correct Function heading (including parameter) and ending
2. Declaration and initialisation of local counter integer variables
3. ~~Correct loop~~
4. Loop terminates if illegal character found
5. Picking up `NextChar` from `Pass`
6. ~~Correct check and increment for lower case~~
7. Correct check and increment for upper case
8. Correct check and increment for numeric
9. Correct check for invalid character
10. ~~Correct final format check and returning correct Boolean value~~

Total mark awarded = 7 out of 9

Question 6 (b) (i)



(b) The `ValidatePassword` function will be tested.

(i) Give a valid password that can be used to check that the function returns TRUE under the correct conditions.

Password1: .... [1]

Specimen Paper Response

PASSword123

Examiner comment:

The answer satisfies the three rules as it contains:

- 4 uppercase alphabetic characters
- 4 lowercase alphabetic characters
- 3 numeric characters
- no non-alphanumeric characters.

Note: The question asks simply for the password, rather than for a STRING value, so double quotation marks are not required. These would be ignored if present.

Total mark awarded = 1 out of 1

Question 6 (b) (ii)

(ii) Password1 is modified to test each rule separately. Give **four** modified passwords and justify your choice.

Password to test rule 1: ....

Reason: ....

....

....

Password to test rule 2: ....

Reason: ....

....

....

Password to test rule 3: ....

Reason: ....

....

....

Password to test rule 4: ....

Reason: ....

....

....

[4]

Specimen Paper Response

Password: PASSw123

Reason: only one lower-case alphabetic characters – should be at least two

Password: Pword123

Reason: only one upper-case alphabetic characters – should be at least two

Password: PASSword12

Reason: only two numeric characters – should be at least three

Password: PASSw-123

Reason: contains a non-alphanumeric character

Examiner comment:

Each password should break a different rule, and each password should break only one rule. A password that breaks more than one rule would be of limited use for test purposes.

A mark is given for each password together with a suitable explanation.

The final password is incorrect. Although it contains an illegal character it also contains an incorrect number of lowercase alphabetic characters.

Total mark awarded = 3 out of 4

Question 6 (b) (iii)

(iii) When testing the `ValidatePassword` function a module it is necessary to test all possible paths through the code.

State the name given to this type of validation testing.

.... [1]

Specimen Paper Response

White-box

Examiner comment:

The correct term has been given.

Total mark awarded = 1 out of 1

Question 6 (b) (iv)

(iv) A program consisting of several functions can be tested using a process known as 'stub testing'

Explain this process.

....

....

....

.... [2]

Specimen Paper Response

Stub testing is the process of testing a function before it has been fully written.

Examiner comment:

The answer addresses the first of the required two points, which are that the function can be tested *before* it is fully written.

For the second mark, the answer would need to say how this could be achieved. For example:

The function contents are replaced with simple code to return a fixed value or to output a message.

Total mark awarded = 1 out of 2

Question 7

Question 7

7 `LogArray` is a 1D array containing 500 elements of type `STRING`.

A procedure, `LogEvents`, is required to add data from the array to the end of the existing text file `LoginFile.txt`

Unused array elements are assigned the value `"Empty"`. These can occur anywhere in the array and should **not** be added to the file.

Write pseudocode for the procedure `LogEvents`.

Refer to the **Insert** for the list of pseudocode functions

[8]

Specimen Paper Response

```
PROCEDURE LogEvents()

  DECLARE FileData : STRING
  DECLARE ArrayIndex : INTEGER
  OPENFILE "LoginFile.txt" FOR WRITE
  FOR ArrayIndex ← 1 TO 500
    IF LogArray[ArrayIndex] <> ""
      THEN
        FileData ← LogArray[ArrayIndex]
        WRITEFILE "LoginFile.txt", FileData
    ENDIF
  NEXT

  CLOSEFILE "LoginFile.txt"

ENDPROCEDURE
```

Examiner comment:

The scenario description is straightforward and although different solutions may include minor variations, all will require the following features:

- The pseudocode will be contained within a procedure named `LogEvents`, which has a header and an end statement
- A loop is required to search through the array. A loop variable (`ArrayIndex` above), of type `INTEGER` is necessary as this will be used to index individual element values within the array
- The file must be opened in `APPEND` mode in order to add data to the end of the existing file
- Within the loop, each array element needs to be checked to check whether it contains the string `"Empty"`
- If an array element does not contain the value `"Empty"` then the value is written to the file
- After the loop ends, loop the file is closed

The answer contains two errors as follows:

- The file is opened in the wrong mode
- The check for an unused elements uses an empty string rather than te value `"Empty"`

The available mark scheme points are shown below. For this answer, marks have been awarded for the points that have not been crossed through.

1. Procedure heading and ending
2. Declare `ArrayIndex` as integer
3. ~~Open file `LoginFile` for append~~

4. Correct loop
5. Extract data from array **in a loop**
6. ~~Check for unused element **in a loop**~~
7. Write data to file **in a loop**
8. Close the file **outside the loop**

Total mark awarded = 6 out of 8

Cambridge Assessment International Education
The Triangle Building, Shaftesbury Road, Cambridge, CB2 8EA, United Kingdom
t: +44 1223 553554
e: info@cambridgeinternational.org   www.cambridgeinternational.org

Copyright © UCLES July 2019