Cambridge Assessment
International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/02**

Paper 2 **For examination from 2021**

MARK SCHEME

Maximum Mark: 75

Specimen

This document has **12** pages. Blank pages are indicated.

[Turn over

Generic Marking Principles

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptors for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1:

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

GENERIC MARKING PRINCIPLE 2:

Marks awarded are always **whole marks**(not half marks, or other fractions).

GENERIC MARKING PRINCIPLE 3:

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

GENERIC MARKING PRINCIPLE 4:

Rules must be applied consistently e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

GENERIC MARKING PRINCIPLE 5:

Marks should be awarded using the full range of marks defined in the mark scheme for the question(however; the use of the full mark range may be limited according to the quality of the candidate responses seen).

GENERIC MARKING PRINCIPLE 6:

Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

Question	Answer	Marks
1(a)(i)	(see table below)	5

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

One mark per row
Accept suitable alternatives for REAL

Question	Answer	Marks
1(a)(ii)	(see table below)	5

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**

One mark per row
Row 1 must have capital ‘C’ and quotes
Rows 2 to 6 must not have quotes

Question	Answer	Marks
1(b)	(see table below)	4

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

One mark per row

Question	Answer	Marks

1(c)	`MyCount ← 101`
	
	`REPEAT`
	    `OUTPUT MyCount`
	    `MyCount ← MyCount + 2`
	`UNTIL MyCount > 199`
	
	One mark for each of the following:
	Counter initialisation before loop
	`Repeat ... Until` loop
	Method for choosing (correct range of) odd numbers
	Output **all** odd numbers in the range
		4

Question	Answer	Marks
2(a)	• The identification of the modules // Checkout, Card payment, Account payment • The hierarchy of modules (allow 'relationship') • Parameters/data/variables passed between modules // The interface between the modules // or by example • The sequence • Module iteration/selection One mark per item Max 3	**3**
2(b)	`FUNCTION CardPayment(Amount : REAL, Name : STRING) RETURNS BOOLEAN` One mark per underlined part Parameter order not significant Function name and parameter names not important but must be present.	**3**

Question	Answer	Marks
3(a)	POP(): • The value 'E' is removed from the stack (and assigned to variable MyVar) • Top of Stack pointer is incremented to 102 PUSH(): • Top of Stack pointer is decremented to 101 • 'Z' is loaded into address 101 Allow follow through for PUSH()	**4**
3(b)	• The received string will be <u>reversed</u> • because the stack operates as a <u>FILO</u> structure	**2**

Name of parameter passing method	**Value output**	**Explanation**
(Call) by reference	5	• A <u>pointer to address of</u> the variable is passed. • <u>Original variable is changed</u> when parameter changed in called module.
(Call) by value	4	• A <u>copy of</u> the variable itself is passed. • <u>Original variable not changed</u> when parameter changed in called module.

Question	Answer	Marks
4(a)	Mark as follows: • One mark for each name **and** corresponding value • One mark per bullet in explanation Max 4 if explanations do not match answers in columns 1 and 2	**6**
4(b)	• Procedures • <u>Local</u> variable One mark per item	**2**

Question	Answer	Marks

5(a)	Pseudocode:
	
	`TYPE StockItem`
	`    DECLARE ProductCode : STRING`
	`    DECLARE Price : REAL`
	`    DECLARE NumberInStock : INTEGER`
	`ENDTYPE`
	(allow `END`)
	
	Mark as follows:
	• One mark for `TYPE` and `ENDTYPE`
	• One mark for `Productcode`
	• One mark for `Price` **and** `NumberInStock`
		3

5(b)	`DECLARE Stock : ARRAY` `[1:1000]` `OF StockItem` One mark per underlined phrase	**3**

5(c)	`Stock[20].Price ← 105.99`
	`Stock[20].NumberInStock ← Stock[20].NumberInStock + 12`
	
	One mark per statement
		2

Question	Answer	Marks
5(d)	Pseudocode:	4

```
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
```

One mark for each of:
- Loop through all elements of the array
- Check Price > 99.99
- `OUTPUT` of 2 fields ...
- ... with suitable supporting text text

(Or could ask for tabular form with column headers)

Question	Answer	Marks
6(a)	Pseudocode solution:	9

```
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 ← 0


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
   ENDIF
   n ← n + 1
ENDWHILE
```

Question	Answer	Marks

6(a)	`IF LCaseChar > 1 AND UCaseChar > 1 AND NumChar > 2 AND ReturnFlag`
	`   THEN`
	`      ReturnFlag ← TRUE`
	`ENDIF`
	`RETURN (ReturnFlag)`
	`ENDFUNCTION`
	
	1 mark for each of the following:
	1 Correct Function heading (including parameter) and ending
	2 Declaration and initialisation of local counter integer variables
	3 Correct FOR loop
	4 Picking up `NextChar` from `InString`
	5 Correct check and increment for lower case
	6 Correct check and increment for upper case
	7 Correct check and increment for numeric
	8 Correct check for invalid character
	9 Correct final format check and returning correct Boolean value	

6(b)(i)	Password1: Any valid string consisting of: • at least 2 uppercase alphabetic • at least 2 lowercase alphabetic • at least 3 numeric characters • No other characters e.g.: ‘ABcd123’	**1**

Question	Answer	Marks
6(b)(ii)	Modify Password1 for each rule: Test string: • Invalid passwords – Lower case characters (e.g. 'ABc123') – Upper case characters (e.g. 'Acd123') – Numeric characters (e.g. 'ABcd12') • Containing an invalid character (e.g. 'ABcd12+3') Mark as follows: One mark for correct invalid string + reason (testing *different* rules of the function); no half marks Each test string must only break a single rule	**4**
6(b)(iii)	White-box	**1**
6(b)(iv)	One mark per bullet: • Testing may be carried out before the modules are developed // not ready for full testing • Module stubs contain simple code to provide a known response // temporary replacement for a called module/return a fixed value/output a message to confirm the module has been called	**2**

Question	Answer	Marks
7	Pseudocode :	**8**

```
PROCEDURE LogEvents()

   DECLARE FileData : STRING
   DECLARE ArrayIndex : INTEGER
   OPENFILE "LoginFile.txt" FOR APPEND
   FOR ArrayIndex ← 1 TO 500 // 0 TO 499
      IF LogArray[ArrayIndex]<> "Empty"
         THEN
            FileData ← LogArray[ArrayIndex]
            WRITEFILE "LoginFile.txt", FileData
      ENDIF
   NEXT

   CLOSEFILE "LoginFile.txt"

ENDPROCEDURE
```

1 mark for each of the following:
1 Procedure heading and ending (ignore any input parameters but don’t allow a return value)
2 Declare `ArrayIndex` (any name) 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**

Allow single write to file **outside** loop if complete string built **within** loop