Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/23**

Paper 2 Written Paper **October/November 2019**

MARK SCHEME

Maximum Mark: 75

Published

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the October/November 2019 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

This document consists of **13** printed pages.

© UCLES 2019

Cambridge Assessment International Education

[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.

© UCLES 2019

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>1(a)(i)</td><td>One mark for each (different) data type ...
... and one mark for a corresponding example value

Acceptable types: Integer, Real, String, Char, Boolean, Date,</td><td>6</td></tr>
<tr><td>1(a)(ii)</td><td>Declaration</td><td>1</td></tr>
<tr><td>1(b)</td><td>Two from (max 2):

• A list of identifier / variable names
• Explanations/descriptions (of what they are used for)
• Data types</td><td>2</td></tr>
<tr><td>1(c)(i)</td><td>One mark for each stage (Input, Output)
One mark for each correct example

<table>
<tr><th>Stage</th><th>Example statement</th></tr>
<tr><td>Input</td><td><code>Next = Console.Readline()</code></td></tr>
<tr><td>Process</td><td><code>x = INT(y/3)</code></td></tr>
<tr><td>Output</td><td><code>Console.Writeline("Goodbye")</code></td></tr>
</table></td><td>5</td></tr>
<tr><td>1(c)(ii)</td><td>One mark for statement in program code that includes (at least) two 'stages'

Example correct answers:

• <code>Next = LEN(Console.Input())</code>
• <code>Console.writeline(Name & Address)</code>
• <code>Console.WriteLine(Console.Readline() & " is what you entered")</code></td><td>1</td></tr>
<tr><td>1(d)</td><td>Three from the following (max 3):

• Blank lines
• Capitalisation of Keywords
• Sensible variable names
• Use of (library/built-in) functions
• Comments
• PrettyPrint / keywords coloured</td><td>3</td></tr>
<tr><td>1(e)</td><td>White-box</td><td>1</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>2(a)(i)</td><td>• Count-controlled // FOR loop
• Used when the number of iterations is known / fixed</td><td>2</td></tr>
<tr><td>2(a)(ii)</td><td>

```
REPEAT
   CALL AlarmReset()
   Status1 ← GetStatus(Sys_A)
   Status2 ← GetStatus(Sys_B)
UNTIL (Status1 = TRUE AND Status2 = TRUE)
```

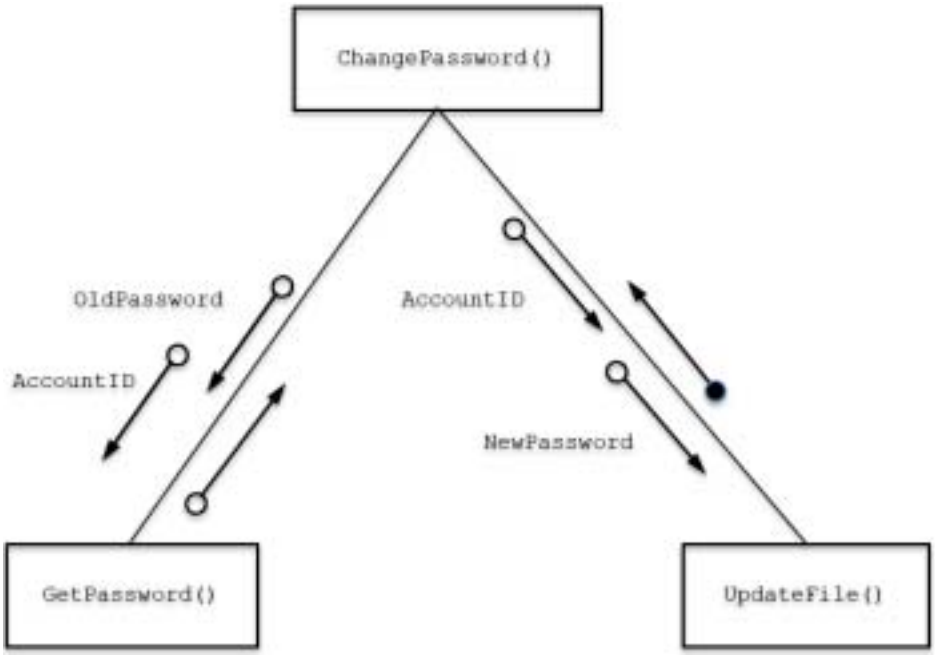
One mark for each of:

1 `REPEAT ... UNTIL`
2 Call to `AlarmReset()`
3 Asignment of `Status1` **and** `Status2`
4 correct logical test

</td><td>4</td></tr>
<tr><td>2(b)</td><td>

<table>
<tr><th>Feature</th><th>Answer</th></tr>
<tr><td>The symbol used to indicate an assignment</td><td>:=</td></tr>
<tr><td>The line numbers for the start and end of a count-controlled loop</td><td>180/190 and 230</td></tr>
<tr><td>The step value of the count-controlled loop</td><td>2</td></tr>
<tr><td>The character that indicates a comment</td><td>%</td></tr>
<tr><td>The name of a function</td><td>Mult // Read</td></tr>
</table>

</td><td>5</td></tr>
<tr><td>2(c)</td><td>Compiler / Interpreter</td><td>1</td></tr>
</table>

Question	Answer	Marks
3	 One mark for each of the following: • all three boxes correctly labelled • parameters in to `GetPassword()` • value back from `GetPassword()` • parameters in to `UpdateFile()` • BOOLEAN value back from `UpdateFile()`	**5**

Question	Answer	Marks
4(a)	One mark for each point. Valid string must contain: • at least one '.' characters • one '@' character • more than 5 other characters.	**3**

Question	Answer	Marks
4(b)(i)	One mark for each area as outlined:	**5**

Index	NextChar	NumDots	NumAts	NumOthers
		0	0	0
1	'J'			1
2	'i'			2
3	'm'			3
4	'.'	1		
5	'9'			4
6	'9'			5
7	'@'		1	
8	's'			6
9	'k'			7
10	'a'			8
11	'i'			9
12	'l'			10
13	'.'	2		
14	'c'			11
15	'o'			12
16	'm'			13

4(b)(ii)	TRUE	**1**

Question	Answer	Marks
4(c)	One mark for string and one mark for correct explanation. Same for second answer providing it results in a **different** path through the algorithm. Correct answers may be: • without the correct number of '.' • without the correct number of '@' • without the correct number of 'other characters'	**4**

© UCLES 2019

Question	Answer	Marks

5	`FUNCTION Abbreviate(Name : STRING) RETURNS STRING`
	
	`  DECLARE NewString : STRING`
	`  DECLARE NextChar : CHAR`
	`  DECLARE Index : INTEGER`
	`  DECLARE Space : BOOLEAN`
	`  CONSTANT SPACECHAR = ' '`
	
	`  Space ← TRUE`
	`  NewString ← ""`
	
	`  FOR Index ← 1 TO LENGTH(Name)`
	`  NextChar ← MID(Name,Index,1)`
	`  IF Space = TRUE`
	`  THEN`
	`  NewString ← NewString & NextChar // first char of next word`
	`  Space ← FALSE`
	`  ELSE`
	`  IF NextChar = SPACECHAR`
	`  THEN`
	`  Space ← TRUE`
	`  ENDIF`
	`  ENDIF`
	`  ENDFOR`
	
	`  RETURN NewString`
	
	`ENDFUNCTION`
	
	1 mark for each of the following (**max 8**):
	
	1 Function header, ending and return parameters
	2 Declare and Initialise `NewString` to either `""` or first character of name
	3 `FOR` loop picking out all characters from `Name`:
	4 extract an individual character **in a loop**
	5 check for space character **in a loop**
	6 concatenate the next character to `NewString` **in a loop**
	7 Return `NewString`
	8 Accommodate a string with trailing space
		8

Question	Answer	Marks
6(a)(i)	One mark per underlined section: `DECLARE Result : ARRAY [0:99, 0:1] OF STRING`	**3**

Question	Answer	Marks

6(a)(ii)	'Pseudocode' solution included here for development and clarification of mark scheme.
	Programming language example solutions appear in the Appendix.
	
	`FUNCTION FindBooksBy(SearchAuthor : STRING) RETURNS INTEGER`
	
	`  DECLARE Title : STRING`
	`  DECLARE Author : STRING`
	`  DECLARE Isbn : STRING`
	`  DECLARE Location : STRING`
	`  DECLARE Count : INTEGER`
	
	`  Count ← 0`
	
	`  OPENFILE "Library.txt" FOR READ`
	
	`  WHILE NOT EOF ("Library.txt")`
	
	`  READFILE "Library.txt", Title`
	`  READFILE "Library.txt", ThisAuthor`
	`  READFILE "Library.txt", ISBN`
	`  READFILE "Library.txt", Location`
	
	`  IF SearchAuthor = ThisAuthor`
	`  THEN`
	`  Result[Count, 0] ← Title`
	`  Result[Count, 1] ← Location`
	`  Count ← Count + 1`
	`  ENDIF`
	
	`  ENDWHILE`
	
	`  CLOSEFILE("Library.txt")`
	
	`  RETURN Count`
	
	`ENDFUNCTION`
	
	One mark for each of the following:
	
	1 Function heading (and ending) including parameters
	2 Declaration of variables used
	3 Open file for reading (Allow `Library` or `Library.txt`)
	4 `WHILE` loop checking for `EOF()`:
	5 Read **all** information 'fields', in the correct order, **in a loop**
	6 If the author matches write `Title` and `Location` to `Result` array **in a loop**
	7 And increment array index **in a loop** // number found
	8 Close file and RETURN `Count`
		8

Question	Answer	Marks

6(b)	<pre>PROCEDURE DisplayResults(Author:STRING, Count:INTEGER)
	  DECLARE Index, GLen : INTEGER
	  DECLARE Gap : STRING
	
	  Gap ← " " // 25 spaces
	
	  IF Count = 0
	  THEN
	  OUTPUT "Search found no books by: " & Author
	  ELSE
	  OUTPUT "Books written by: " & Author
	  OUTPUT "Title" & LEFT(Gap, 20) & "Location"
	  FOR Index ← 1 TO Count
	  GLen ← 25 - LENGTH(Result[Index, 0])
	  OUTPUT Result[Index, 0] & LEFT(Gap, GLen) &
	  Result[Index, 1]
	  ENDFOR
	  OUTPUT "Number of titles found: " &
	  NUM_TO_STRING(Count)
	  ENDIF
	ENDPROCEDURE</pre>
	One mark for each of the following (**max 7**):
	
	1 Procedure heading and ending including parameters
	2 Declaration of local INTEGER variable for use as index
	3 Test if count = 0 **and if so** output suitable message including `Author` for no books found **otherwise** output the two header strings (exact format not important)
	4 A `FOR` loop for `Count times`
	5 ... output two array elements from `Result` array **in a loop**
	6 Final output statement
	7 A reasonable attempt at calculating the number of spaces required to align 'Location' column:
	8 Alignment correct
		7

Program Code Example Solutions

Q6 (a) (ii): Visual Basic

```
FUNCTION FindBooksBy(ByVal SearchAuthor As String) As Integer

  Dim Title As String
  Dim Author As String
  Dim Isbn As String
  Dim Location As String
  Dim Count As Integer

  Count = 0

  FileOpen(1, "Library.txt", OpenMode.Input)

  While Not EOF(1)
    Title = LineInput(1)
    ThisAuthor = LineInput(1)
    Isbn = LineInput(1)
    Location = LineInput(1)

    If SearchAuthor = ThisAuthor Then
        Result(Count, 0) = Title
        Result(Count, 1) = Location
        Count = Count + 1
    End If
  End While

  FileClose(1)

  Return Count
END FUNCTION
```

Q6 (a) (ii): Pascal

```
function FindBooksBy(SearchAuthor : string) : integer;

  var
    Title : string;
    Author : string;
    Isbn : string;
    Location : string;
    Count : integer;
    MyFile : text;

  begin
     Count := 0;
     assign(MyFile, 'Library.txt');
     reset(MyFile);

     while not EOF(MyFile) do
     begin
        readln(MyFile, Title);
        readln(MyFile, ThisAuthor);
        readln(MyFile, Isbn);
        readln(MyFile, Location);

        if SearchAuthor = ThisAuthor then
        begin
           Result[Count, 0] := Title;
           Result[Count, 1] := Location;
           Count := Count + 1;
        end;

     end;

     close(MyFile);
     FindBooksBy := Count;

  end;
```

Q6 (a) (ii): Python

```
def FindBooksBy(SearchAuthor):

  ## Title : STRING
  ## Author : STRING
  ## Isbn : STRING
  ## Location : STRING
  ## Count : INTEGER

    Count = 0
    MyFile = open("Library.txt", 'r')
    Title = MyFile.readline()

    while Title != "":
        ThisAuthor = MyFile.readline()
        Isbn= MyFile.readline()
        Location = MyFile.readline()
        if SearchAuthor == ThisAuthor.strip():
            Result[Count][0] = Title
            Result[Count][1] = Location
            Count = Count + 1
        Title = MyFile.readline()

    MyFile.close()
    return(Count)
```