



## Cambridge International AS & A Level

---

COMPUTER SCIENCE

9608/23

Paper 2 Written Paper

October/November 2020

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 2020 series for most Cambridge IGCSE™, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

---

This document consists of **27** printed pages.

**PUBLISHED****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)	One mark per bullet point  The purpose is: <ul style="list-style-type: none"> <li>• to express the algorithm in a level of sufficient detail // to split a large task into (smaller) sub-tasks</li> <li>• so that it can be programmed // so that individual tasks are easier to solve // to make the problem more manageable / understandable</li> </ul>	<b>2</b>										
1(b)	Many acceptable answers, must be four different data types together with appropriate values One mark per row  For example: <table border="1" data-bbox="338 691 981 1018" style="margin-left: 20px;"> <thead> <tr> <th data-bbox="338 691 658 754">Data type</th> <th data-bbox="658 691 981 754">Example data value</th> </tr> </thead> <tbody> <tr> <td data-bbox="338 754 658 820">BOOLEAN</td> <td data-bbox="658 754 981 820">FALSE</td> </tr> <tr> <td data-bbox="338 820 658 885">CHAR</td> <td data-bbox="658 820 981 885">'!'</td> </tr> <tr> <td data-bbox="338 885 658 951">DATE</td> <td data-bbox="658 885 981 951">01/01/01</td> </tr> <tr> <td data-bbox="338 951 658 1018">INTEGER</td> <td data-bbox="658 951 981 1018">27</td> </tr> </tbody> </table> <p>Note: STRING and REAL are excluded as these are given in the question.</p>	Data type	Example data value	BOOLEAN	FALSE	CHAR	'!'	DATE	01/01/01	INTEGER	27	<b>4</b>
Data type	Example data value											
BOOLEAN	FALSE											
CHAR	'!'											
DATE	01/01/01											
INTEGER	27											
1(c)(i)	Max 1 mark, features include: <ul style="list-style-type: none"> <li>• Control Structures / selection statements / iteration statements / IO statements</li> <li>• Modular structure (functions, procedures)</li> <li>• Parameters to / from subroutines</li> <li>• Variable declaration / assignment / data structures / OOP ref</li> </ul>	<b>1</b>										

**PUBLISHED**

Question	Answer	Marks
1(c)(ii)	<ul style="list-style-type: none"> <li>Transferable skill</li> </ul>	<b>1</b>
1(d)	<p>Max 3 marks, methods include:</p> <ul style="list-style-type: none"> <li>IDE features: breakpoints / single stepping / watch window</li> <li>Manually check program code / reading error report</li> <li>Trace table / dry run / White-box testing</li> <li>Use of appropriate test data</li> <li>Addition of output statement to follow changes to variables</li> </ul>	<b>3</b>

Question	Answer	Marks
2(a)	<p>One mark per step (or equivalent):</p> <ol style="list-style-type: none"> <li>Set Total to 0</li> <li>Set AGradeCount to 0</li> <li>Input Mark</li> <li>Add Mark to Total</li> <li>If Mark &gt; 75 then increment AGradeCount</li> <li>Repeat from Step 3 for 30 times</li> <li>Output AGradeCount</li> <li>Output Total / 30</li> </ol>	<b>8</b>

**PUBLISHED**

Question	Answer	Marks												
2(b)	<p>One mark per row:</p> <table border="1"> <thead> <tr> <th data-bbox="338 272 1016 338">Statement</th> <th data-bbox="1016 272 1695 338">Error</th> </tr> </thead> <tbody> <tr> <td data-bbox="338 338 1016 403">Code ← LEFT(3, "Europe")</td> <td data-bbox="1016 338 1695 403">Parameters are reversed</td> </tr> <tr> <td data-bbox="338 403 1016 509">Hour ← MID("ALARM:12:02", 7, 6)</td> <td data-bbox="1016 403 1695 509">Third param too big (should be max 5) // string too short</td> </tr> <tr> <td data-bbox="338 509 1016 574">Size ← LENGTH(27.5)</td> <td data-bbox="1016 509 1695 574">Invalid type – param should be a string</td> </tr> <tr> <td data-bbox="338 574 1016 639">Num ← INT(27/ (Count + 3)</td> <td data-bbox="1016 574 1695 639">Missing closing bracket</td> </tr> <tr> <td data-bbox="338 639 1016 721">Result ← "Conditional" AND "Loop"</td> <td data-bbox="1016 639 1695 721">Wrong variable types / operator</td> </tr> </tbody> </table>	Statement	Error	Code ← LEFT(3, "Europe")	Parameters are reversed	Hour ← MID("ALARM:12:02", 7, 6)	Third param too big (should be max 5) // string too short	Size ← LENGTH(27.5)	Invalid type – param should be a string	Num ← INT(27/ (Count + 3)	Missing closing bracket	Result ← "Conditional" AND "Loop"	Wrong variable types / operator	<b>5</b>
Statement	Error													
Code ← LEFT(3, "Europe")	Parameters are reversed													
Hour ← MID("ALARM:12:02", 7, 6)	Third param too big (should be max 5) // string too short													
Size ← LENGTH(27.5)	Invalid type – param should be a string													
Num ← INT(27/ (Count + 3)	Missing closing bracket													
Result ← "Conditional" AND "Loop"	Wrong variable types / operator													

Question	Answer	Marks
2(c)	<p>'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix.</p> <pre> Index ← 0 Status ← FALSE WHILE Status &lt;&gt; TRUE     Status ← TopUp()     Index ← Index + 1 ENDWHILE  IF Index &gt; 100     THEN         SetLevel("Super") ENDIF </pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> <li>1 Set Index to 0 <b>and</b> Status to FALSE</li> <li>2 Pre-condition loop</li> <li>3 Assign value of TopUp() to Status <b>in a loop</b></li> <li>4 Increment Index <b>in a loop</b></li> <li>5 Test Index greater than 100 <b>after loop</b></li> <li>6 If TRUE then Call to SetLevel with param "Super"</li> </ol>	<b>6</b>

Question	Answer	Marks
3(a)	<p>'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix.</p> <pre> PROCEDURE BubbleSort()   DECLARE Temp : INTEGER   DECLARE NoSwaps : BOOLEAN   DECLARE Boundary, J : INTEGER    Boundary ← 4999   REPEAT     NoSwaps ← TRUE     FOR J ← 1 TO Boundary       IF ProdNum[J] &gt; ProdNum[J+1]         THEN           Temp ← ProdNum[J]           ProdNum[J] ← ProdNum[J+1]           ProdNum[J+1] ← Temp           NoSwaps ← FALSE         ENDIF       ENDFOR       Boundary ← Boundary - 1     UNTIL NoSwaps = TRUE    ENDPROCEDURE </pre>	7



**PUBLISHED**

Question	Answer	Marks
3(a)	<p>Mark as follows, max 7 marks from 8 possible marks:</p> <ol style="list-style-type: none"> <li>1 Procedure heading and ending</li> <li>2 Conditional outer loop (may be count-controlled but if so must be <math>\geq 4999</math> iterations)</li> <li>3 An inner loop</li> <li>4 Correct range for inner loop</li> <li>5 Comparison (element <math>n</math> with <math>n + 1</math>) <b>in a loop</b></li> <li>6 Swap array element <b>in a loop</b></li> <li>7 'No-Swap' mechanism: (<b>both</b> needed for mark):               <ul style="list-style-type: none"> <li>○ Conditional outer loop including flag reset</li> <li>○ Flag set in inner loop to indicate swap</li> </ul> </li> <li>8 Reducing Boundary <b>in the <u>outer</u> loop</b></li> </ol>	

Question	Answer	Marks
4(a)	<pre> FUNCTION Search(SearchString : STRING) RETURNS INTEGER    DECLARE RetVal : INTEGER   DECLARE Index : INTEGER    RetVal ← -1   Index ← 1    WHILE Index &lt;= 100 AND RetVal = -1     IF NameList[Index] = SearchString       THEN         RetVal ← Index       ENDIF     Index ← Index + 1   ENDWHILE    RETURN RetVal  ENDFUNCTION </pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> <li>1 Function heading and ending including parameter</li> <li>2 Declaration of integer for <code>Index</code></li> <li>3 Initialisation <b>and</b> increment of <code>Index</code> (implied in FOR loop)</li> <li>4 Conditional loop // FOR loop with immediate RETURN if <code>SearchString</code> found</li> <li>5 Comparison of array element with <code>SearchString</code> <b>AND</b> assigning <b>just the first</b> occurrence to <code>RetVal</code> <b>OR</b> setting the termination condition</li> <li>6 Return <code>RetVal</code> (correctly in both cases)</li> </ol>	<b>6</b>

**PUBLISHED**

Question	Answer	Marks						
4(b)	<ul style="list-style-type: none"> <li>Adaptive maintenance</li> </ul>	<b>1</b>						
4(c)	Ma 1 mark, reasons include: <ul style="list-style-type: none"> <li>Program doesn't perform as expected / does not meet the <u>original</u> specification</li> <li>Program contains errors / bugs</li> <li>Performance / efficiency needs improving</li> <li>New hardware has been introduced</li> </ul>	<b>1</b>						
4(d)	One mark for each value One mark for each explanation <table border="1" data-bbox="338 627 1433 823"> <thead> <tr> <th data-bbox="338 627 472 692">Output</th> <th data-bbox="472 627 1433 692">Explanation</th> </tr> </thead> <tbody> <tr> <td data-bbox="338 692 472 758">20</td> <td data-bbox="472 692 1433 758">A <u>copy of</u> the variable itself is passed</td> </tr> <tr> <td data-bbox="338 758 472 823">25</td> <td data-bbox="472 758 1433 823">A <u>pointer to</u> / <u>the address of</u> the variable is passed</td> </tr> </tbody> </table>	Output	Explanation	20	A <u>copy of</u> the variable itself is passed	25	A <u>pointer to</u> / <u>the address of</u> the variable is passed	<b>4</b>
Output	Explanation							
20	A <u>copy of</u> the variable itself is passed							
25	A <u>pointer to</u> / <u>the address of</u> the variable is passed							
4(e)	Max 2 marks, example answers: <ul style="list-style-type: none"> <li>Allows the module to be called from many / multiple places // re-used</li> <li>Module code can be (independently) tested and debugged once and can then be used repeatedly</li> <li>If the module task changes the change needs to be made only once</li> <li>Reduces unnecessary code duplication</li> <li>Allows modules to be shared among many programmers / given to programmers with specific skills</li> <li>Makes the program easier to work on / debug / test / etc</li> </ul>	<b>2</b>						

Question	Answer	Marks
5(a)	<pre> FUNCTION AddHashtag (HashTag : STRING) RETURNS BOOLEAN   DECLARE Index : INTEGER   DECLARE Added : BOOLEAN   CONSTANT EMPTY = ""    Added ← FALSE   Index ← 1 // first element    REPEAT     IF TagString[Index] = EMPTY       THEN         TagString[Index] ← HashTag         TagCount[Index] ← 1         Added ← TRUE       ELSE         Index ← Index + 1     ENDIF    UNTIL Index &gt; 10000 OR Added = TRUE    RETURN Added ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Declaration of two local variables: Integer for <code>index</code> &amp; Boolean for return value (unless immediate Return used)</li> <li>2 Conditional loop through all elements until empty element found OR end of array</li> <li>3 Test if <code>TagString</code> element is empty <b>in a loop</b></li> <li>4 If so then assign <code>HashTag</code> to <code>TagString[]</code> and 1 to <code>TagCount[]</code></li> <li>5 Set loop termination</li> <li>6 Return Boolean (for both cases)</li> </ol>	<b>6</b>

Question	Answer	Marks
5(b)	<p>'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix.</p> <pre> FUNCTION CountHashtag (Message : STRING) RETURNS INTEGER   DECLARE TagNum, StartPos : INTEGER   DECLARE Found : BOOLEAN    TagNum ← 0   Found ← TRUE    REPEAT     StartPos ← GetStart (Message, TagNum + 1)     IF StartPos = -1       THEN         Found ← FALSE       ELSE         TagNum ← TagNum + 1       ENDIF   UNTIL NOT Found    RETURN TagNum  ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Function heading and ending including parameter</li> <li>2 Declaration <b>and</b> initialisation of local integer for count (TagNum)</li> <li>3 Conditional loop through message</li> <li>4 Use of GetStart () <b>in a loop</b></li> <li>5 Test GetStart () return value for -1 and increment count accordingly <b>in a loop</b></li> <li>6 Return integer value</li> </ol>	<b>6</b>

Question	Answer	Marks
5(c)	<p>'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix.</p> <pre> FUNCTION IncrementHashtag (HashTag : STRING) RETURNS BOOLEAN   DECLARE Index : INTEGER   DECLARE Found : BOOLEAN    Found ← FALSE   Index ← 1 // first element    REPEAT     IF TagString[Index] = HashTag       THEN         TagCount[Index] ← TagCount[Index] + 1         Found ← TRUE       ELSE         Index ← Index + 1       ENDIF   UNTIL Index &gt; 10000 OR Found = TRUE   RETURN Found ENDFUNCTION </pre> <p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Conditional loop until hashtag found <b>or</b> end of array</li> <li>2 Compare element value to parameter <b>in a loop</b></li> <li>3 If found, increment corresponding TagCount element</li> <li>4 Return Boolean correctly in both cases</li> </ol>	<b>4</b>

Question	Answer	Marks
5(d)	<pre> PROCEDURE OutputMostPop()   DECLARE Index : INTEGER   DECLARE MostPopTag : STRING   DECLARE Max : INTEGER      // the integer value of the biggest number   DECLARE Count : INTEGER    CONSTANT EMPTY = ""    Max ← -1    FOR Index ← 1 To 10000     IF TagCount[Index] &gt; Max       THEN         Max ← TagCount[Index]         Count ← 1      // there is only one max value         MostPopTag ← TagString[Index]       ELSE         IF TagCount[Index] = Max           THEN             Count ← Count + 1      // another max value           ENDIF         ENDIF       ENDFOR    IF Count = 1     THEN       OUTPUT "The most popular hashtag is: ", MostPopTag, "It occurs: ", Max, " times."     ELSE       OUTPUT "The maximum hashtag count is: ", Max, __               "The number of hashtags with this count is: ", Count     ENDIF   ENDPROCEDURE </pre>	8

**PUBLISHED**

Question	Answer	Marks
5(d)	<p>1 mark for each of the following:</p> <ol style="list-style-type: none"> <li>1 Initialise Max to a value less than 1 or to TagCount [1]</li> <li>2 Loop through all elements</li> <li>3 Test if TagCount value &gt; Max <b>in a loop</b></li> <li>4 and if so set Max to TagCount value</li> <li>5 and save TagString element (or array index) <b>and</b> set Count to 1 (unless counting is separate)</li> <li>6 ELSE If TagCount value = Max, increment Count (or via separate loop)</li> <li>7 Output for single max <b>after the loop</b></li> <li>8 Or Output for multiple max <b>after the loop</b></li> </ol> <p>Alternative "two-loop" solution:</p> <pre> PROCEDURE OutputMostPop ()   DECLARE Index : INTEGER   DECLARE MostPopTag : STRING   DECLARE Max : INTEGER //The integer value of the biggest number   DECLARE MaxCount : INTEGER    CONSTANT EMPTY = ""    Max ← -1    FOR Index ← 1 To 10000     IF TagCount[Index] &gt; Max       THEN         Max ← TagCount[Index]         MostPopTag ← TagString[Index]       ENDIF   ENDFOR </pre>	



Question	Answer	Marks
5(d)	<pre> MaxCount ← 0 FOR Index ← 1 To 10000   IF TagCount[Index] = Max     THEN       MaxCount ← MaxCount + 1     ENDIF   ENDFOR  IF MaxCount = 1   THEN     OUTPUT "The most popular hashtag is: ", MostPopTag, ". It occurs: ", Max, " times."   ELSE     OUTPUT "The mamimum value is: ",Max, ". It occurred", MaxCount, " times."   ENDIF ENDPROCEDURE </pre>	

\*\*\* End of Mark Scheme – example program code solutions follow \*\*\*

**Program Code Example Solutions****Q2 (c): Visual Basic**

```
Index = 0
Status = FALSE
Do While Status <> TRUE
    Status = TopUp()
    Index = Index + 1
Loop

If Index > 100 Then
    SetLevel("Super")
End If
```

**Q2 (c): Pascal**

```
Index := 0;
Status := FALSE;

while Status <> TRUE do
begin
    Status := TopUp();
    Index := Index + 1;
end;

if Index > 100 then
    SetLevel("Super");
```

**Q2 (c): Python**

```
Index = 0
Status = FALSE
while Status <> TRUE:
    Status = TopUp()
    Index = Index + 1

if Index > 100:
    SetLevel("Super")
```

**Q3: Visual Basic**

```
Sub BubbleSort()
    Dim Temp As Integer
    Dim NoSwaps As Boolean
    Dim Boundary, J As Integer

    Boundary = 4998
    Do
        NoSwaps = TRUE
        For J = 0 To Boundary
            If ProdNum(J) > ProdNum(J+1) Then
                Temp = ProdNum(J)
                ProdNum(J) = ProdNum(J+1)
                ProdNum(J+1) = Temp
                NoSwaps = FALSE
            End If
        Next
        Boundary = Boundary - 1
    Loop Until NoSwaps = TRUE
End Sub
```

**Q3: Pascal**

```
Peocedure BubbleSort ();
var
  Temp: Integer;
  NoSwaps : Boolean;
  Boundary, J : Integer;

begin
  Boundary := 4999;
  repeat
    NoSwaps := TRUE;
    for J := 1 To Boundary do
      begin
        if ProdNum[J] > ProdNum[J+1] then
          begin
            Temp := ProdNum[J];
            ProdNum[J] := ProdNum[J+1];
            ProdNum[J+1] := Temp;
            NoSwaps := FALSE;
          end;
        end;
    Boundary := Boundary - 1;
  until NoSwaps = TRUE;

end;
```

**Q3: Python**

```
def BubbleSort():
    # Temp As Integer
    # NoSwaps As Boolean
    # Boundary, J As Integer

    NoSwaps = False
    Boundary = 4999

    while not NoSwaps:
        NoSwaps = True
        for J in range(Boundary):
            if ProdNum[J] > ProdNum[J+1]:
                Temp = ProdNum[J]
                ProdNum[J] = ProdNum[J+1]
                ProdNum[J+1] = Temp
                NoSwaps = FALSE

        Boundary = Boundary - 1
```

**Q5 (b): Visual Basic**

```
Function CountHashtag (Message As STRING) As INTEGER
    Dim TagNum As INTEGER
    Dim StartPos As INTEGER
    Dim Found As BOOLEAN

    TagNum = 0
    Found = TRUE

    Do
        StartPos = GetStart(Message, TagNum + 1)
        If StartPos = -1 Then
            Found = FALSE
        Else
            TagNum = TagNum + 1
        End If
    Loop Until No Found

    Return TagNum

End Function
```

**Q5 (b): Pascal**

```
Function CountHashtag (Message : STRING) : INTEGER;
var
  TagNum : Integer;
  StartPos : Integer;
  Found : Boolean;

begin
  TagNum := 0;
  Found := TRUE;

  repeat
    StartPos := GetStart(Message, TagNum + 1);
    if StartPos = -1 then
      Found := FALSE
    else
      TagNum := TagNum + 1;

until Not Found;

CountHashtag := TagNum;

end;
```

**Q5 (b): Python**

```
def CountHashtag (Message)
    # TagNum, StartPos As INTEGER
    # Found As BOOLEAN

    TagNum = 0
    Found = TRUE

    while Found:
        StartPos = GetStart (Message, TagNum + 1)
        if StartPos == -1:
            Found = FALSE
        else:
            TagNum = TagNum + 1

    return TagNum
```



**Q 5 (c): Visual Basic**

```
Function IncrementHashtag (Hashtag As String) As Boolean
    Dim Index As Integer
    Dim Found As Boolean

    Found = False
    Index = 1 'First element

    Do
        If TagString(Index) = Hashtag Then
            TagCount(Index) = TagCount(Index) + 1
            Found = True
        Else
            Index = Index + 1
        End If
    Loop Until Index > 10000 Or Found = True

    Return Found
End Function
```

**Q 5 (c): Pascal**

```
Function IncrementHashtag (Hashtag : String) : Boolean;
var
  Index : Integer;
  Found : Boolean

begin
  Found := FALSE;
  Index := 1; //First element

  repeat
    If TagString[Index] = Hashtag then
      begin
        TagCount[Index] := TagCount[Index] + 1;
        Found := TRUE;
      end
    else
      Index := Index + 1;

until Index > 10000 OR Found = TRUE;

IncrementHashtag := Found;

end;
```

**Q 5 (c): Python**

```
def IncrementHashtag (HashTag):
    # Index As Integer
    # Found As Boolean

    Found = FALSE
    Index = 0 #First element

    while not Found and Index < 10000:
        if TagString[Index] == HashTag:
            TagCount[Index] = TagCount[Index] + 1
            Found = TRUE
        else:
            Index = Index + 1

    Return Found
```