Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/42**

Paper 4 Written Paper **May/June 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 May/June 2019 series for most Cambridge IGCSE™, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

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:

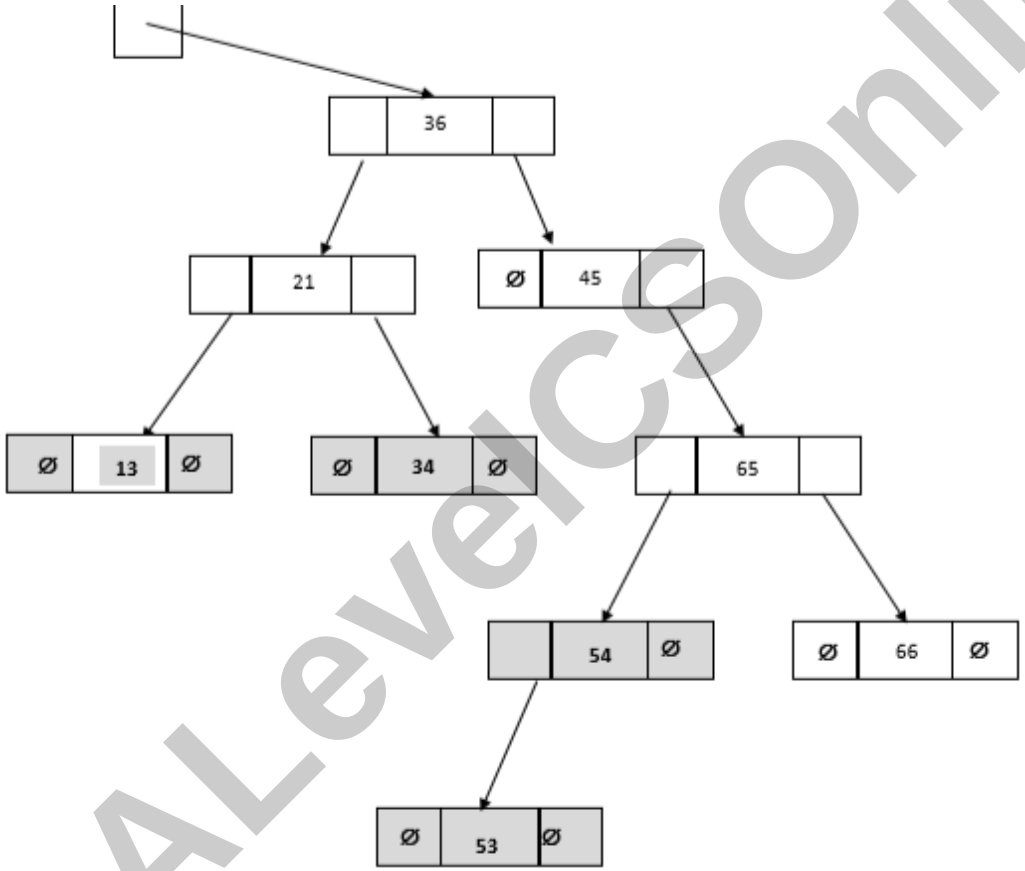
Rules must be applied consistently e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

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)(i)	1 mark per bullet point • 13 to left of 21 • 34 to right of 21 • 54 to left of 65 • 53 to left of 54 • Null and other pointers on all boxes written **and** no other pointers filled in  	**5**

Question	Answer	Marks
1(a)(ii)	1 mark per bullet point • FreePointer • 36 • 45 • 21 and 65 • 66 and 13 • 54, 53 and 34	**6**

RootPointer
0

FreePointer
9

Index	LeftPointer	Data	RightPointer
[0]	2	36	1
[1]	null	45	3
[2]	5	21	8
[3]	6	65	4
[4]	null	66	null
[5]	null	13	null
[6]	7	54	null
[7]	null	53	null
[8]	null	34	null
[9]			

Question	Answer	Marks

1(b)(i)	1 mark per bullet point
	
	• Language specific constructor header and close (where appropriate)
	• …with two parameters (variables)
	• Initialise `FinalMark` to `0`
	• Initialise `Grade` to `"Fail"`
	• Initialise `PaperID` to `CentreNumber + CandidateNumber`
	
	PYTHON
	
	`def__init__(self, CentreNumber,CandidateNumber):`
	`  self.__FinalMark = 0`
	`  self.__Grade = "Fail"`
	`  self.__PaperID = CentreNumber + CandidateNumber`
	
	PASCAL
	
	`Contructor NewExaminationPaper.Create(CentreNumber  : String, CandidateNumber : String);`
	`begin`
	`  FinalMark := 0;`
	`  Grade := 'Fail';`
	`  PaperID := CentreNumber + CandidateNumber;`
	`end;`
	
	VB
	
	`Public Sub New (ByVal CentreNumber As String, ByVal CandidateNumber As String)`
	`  FinalMark = 0`
	`  Grade = "Fail"`
	`  PaperID = CentreNumber & CandidateNumber`
	`End Sub`
		5

Question	Answer	Marks
1(b)(ii)	1 mark per bullet point to max 3 • Used to access/change the properties/attributes • …only using the get/set methods • …that are set to private • Provide encapsulation • Prevents accidental change • To make sure data is valid // act as validation • Hides data • The get methods allow the data to be accessed/returned • The set methods allow the data to be changed/written to	**3**

Question	Answer	Marks

1(b)(iii)	1 mark per bullet point
	
	• 1 get method header without parameter (returning string where appropriate)
	• …returning the property
	• A second working Get
	• A third working Get
	
	PYTHON
	
	`def GetFinalMark():`
	`  return (FinalMark)`
	`def GetGrade():`
	`  return (Grade)`
	`def GetPaperID():`
	`  return (PaperID)`
	
	PASCAL
	
	`function GetFinalMark():Integer;`
	`begin`
	`  GetFinalMark:= FinalMark;`
	`end;`
	`function GetGrade():String;`
	`begin`
	`  GetGrade:= Grade;`
	`end;`
	`function GetPaperID():string;`
	`begin`
	`  GetPaperID:= PaperID;`
	`end;`
		4

Question	Answer	Marks

1(b)(iii)	**VB**
	
	`Public Function GetFinalMark() As Integer`
	`  Return FinalMark`
	`End Function`
	`Public Function GetGrade() As String`
	`  Return Grade`
	`End Function`
	`Public Function GetPaperID() As String`
	`  Return PaperID`
	`End Function`	

Question	Answer	Marks

1(b)(iv)	1 mark per bullet point
	
	• Set header taking parameter and close (where appropriate
	• check if **parameter** >= 0 and <= 90
	• If valid (>=0 **and** <=90) return TRUE
	• …and set FinalMark to parameter
	• If not valid, return FALSE **and** do not set FinalMark
	
	PYTHON
	
	`def SetFinalMark (Mark):`
	`  if Mark >=0 and Mark <=90:`
	`  IsValid = True`
	`  FinalMark = Mark`
	`  else:`
	`  IsValid = False`
	`  return(IsValid)`
	
	PASCAL
	
	`function SetFinalMark (Mark: Integer) : Boolean;`
	`var IsValid : Boolean;`
	`begin`
	`  If (Mark >=0) AND (Mark <=90) Then`
	`  FinalMark := Mark;`
	`  IsValid := True;`
	`  Else`
	`  IsValid :=False;`
	`end;`
		5

Question	Answer	Marks

1(b)(iv)	**VB**
	
	`Public Function SetFinalMark(ByVal Mark As Integer) As Boolean`
	`  If (Mark >=0) And (Mark <=90) Then`
	`  FinalMark = Mark`
	`  Return True`
	`  Else`
	`  Return False`
	`  End If`
	`End Function`	

Question	Answer	Marks

1(b)(v)	1 mark per bullet point
	
	• Procedure header with **three** parameters (and close where appropriate)
	• Check if `FinalMark` is >= `DistMark` and `Grade` set to `Distinction`
	• Check if `FinalMark` is >= `MeritMark` and <`DistMark` and `Grade` set to `Merit`
	Check if `FinalMark` is >= `PassMark` and <`MeritMark` and `Grade` set to `Pass`
	• Otherwise `Grade` set to `Fail`
	
	PYTHON
	
	`def SetGrade(DistMark, MeritMark, PassMark):`
	`  if FinalMark >= DistMark:`
	`  Grade = "Distinction"`
	`  elif FinalMark >= MeritMark:`
	`  Grade = "Merit"`
	`  elif FinalMark >= PassMark:`
	`  Grade = "Pass"`
	`  else:`
	`  Grade = "Fail"`
	
	PASCAL
	
	`procedure SetGrade(DistMark, MeritMark, PassMark)`
	`begin`
	`  If FinalMark >= DistMark Then`
	`  Grade := "Distinction";`
	`  Else If FinalMark >= MeritMark Then`
	`  Grade := "Merit";`
	`  Else If FinalMark >= PassMark Then`
	`  Grade:= "Pass";`
	`  Else`
	`  Grade:= "Fail";`
	`end;`
		4

Question	Answer	Marks

1(b)(v)	**VB**
	
	`Public Sub SetGrade(DistMark, MeritMark, PassMark)`
	`  If FinalMark >= DistMark Then`
	`  Grade = "Distinction"`
	`  ElseIf FinalMark >= MeritMark Then`
	`  Grade = "Merit"`
	`  ElseIf FinalMark >= PassMark Then`
	`  Grade = "Pass"`
	`  Else`
	`  Grade = "Fail"`
	`  End If`
	`End Sub`	

Question	Answer	Marks

1(b)(vi)	1 mark per bullet point to max 8
	
	• Procedure `Main` header and close (where appropriate)
	• Input candidate number, centre number and mark with suitable prompt(s)
	• Create instance of `ExaminationPaper` named `ThisPaper`...
	• … with input parameters candidate number and centre number
	• Call `SetFinalMark` for `ThisPaper` with mark input as parameter
	• …storing/using return value
	• …outputting a message if this is valid or invalid
	• Call `SetGrade` for `ThisPaper` with correct thresholds
	• Output grade for `ThisPaper` ...
	• …using `.GetGrade`
	
	PYTHON
	
	`def main():`
	`  candidateNumber = input("Please enter the candidate number")`
	`  centreNumber = input("Please enter the centre number")`
	`  mark = input("Please enter the mark")`
	`  ThisPaper = ExaminationPaper(centreNumber, candidateNumber)`
	`  if ThisPaper.SetFinalMark(mark) == FALSE:`
	`  print("Invalid mark")`
	`  else:`
	`  ThisPaper.SetGrade(80, 70, 55)`
	`  print(ThisPaper.GetGrade())`
		8

Question	Answer	Marks

1(b)(vi)	**PASCAL**
	
	`procedure Main();`
	`  var`
	`  candidateNumber, centreNumber : String;`
	`  isValid : boolean;`
	`  thisPaper : ExaminationPaper;`
	`  mark : integer;`
	`  begin`
	`  Writeln(Enter candidate number: ');`
	`  Readln(candidateNumber);`
	`  Writeln('Enter centre number: ');`
	`  Readln(centreNumber);`
	`  ThisPaper := ExaminationPaper.Create(centreNumber, candidateNumber);`
	`  Writeln('Enter mark: ');`
	`  Readln(mark);`
	`  isValid := ThisPaper.SetFinalMark(mark);`
	`  if isValid = true:`
	`  thisPaper.SetGrade(80, 70, 55);`
	`  Writeln(ThisPaper.GetGrade());`
	`  else:`
	`  Writeln("Invalid mark")`
	`end;`	

Question	Answer	Marks

1(b)(vi)	**VB**
	
	`Sub main()`
	`  Dim candidateNumber As String`
	`  Dim centreNumber As String`
	`  Console.WriteLine("Please enter the candidate number")`
	`  candidateNumber = Console.ReadLine()`
	`  Console.WriteLine("Please enter the centre number")`
	`  centreNumber = Console.ReadLine()`
	`  Dim ThisPaper As New ExaminationPaper(centreNumber, candidateNumber)`
	`  Dim IsValid As Boolean`
	`  Console.WriteLine("Please enter the mark")`
	`  Dim mark As Integer`
	`  mark = Console.ReadLine()`
	`  IsValid = ThisPaper.SetFinalMark(mark)`
	`  if IsValid = True then`
	`  ThisPaper.SetGrade(80, 70, 55)`
	`  Console.WriteLine(ThisPaper.GetGrade())`
	`  else`
	`  Console.WriteLine("Invalid mark")`
	`  endif`
	`End Sub`	

1(c)	1 mark per bullet point to max 4 • There are a large number of objects/records to store // hash is better for a large number of objects/records • A hashing **algorithm/hash** performed on **key field/record** (to form the address) • …to allow direct access to the object • …so it is likely to be faster in finding the object // linked list is slower in finding the object • In a linked list, each object needs to be checked until found // sequentially/linear accessed • …the left/right/next pointer is followed // have to trace pointers	**4**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>2(a)</td><td>1 mark for correct tick

<table><tr><th>Statement</th><th>Tick (✓)</th></tr><tr><td>Last in first out</td><td>✓</td></tr><tr><td>First in first out</td><td></td></tr><tr><td>Last in last out</td><td></td></tr></table></td><td>1</td></tr>
<tr><td>2(b)(i)</td><td>1 mark for correct stack

<table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td>10</td></tr><tr><td>35</td></tr><tr><td>20</td></tr></table></td><td>1</td></tr>
<tr><td>2(b)(ii)</td><td>1 mark for correct stack

<table><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td></td></tr><tr><td>65</td></tr><tr><td>50</td></tr><tr><td>35</td></tr><tr><td>20</td></tr></table></td><td>1</td></tr>
</table>

Question	Answer	Marks

2(b)(iii)	1 mark for each bullet
	
	• Function `Push` …
	• …taking parameter (returning Boolean)
	• Checking if `Top` = 7 …
	• …returning `FALSE` if full
	• …returning `TRUE` otherwise
	• if not full, increment `Top`
	• ... add parameter to `Top` of `ArrayStack`
	
	`FUNCTION Push (BYVALUE DataItem : Integer) (RETURNS Boolean)`
	`  IF Top = 7`
	`  THEN`
	`  RETURN FALSE`
	`  ELSE`
	`  Top ← Top + 1`
	`  ArrayStack[Top] ← DataItem`
	`  RETURN TRUE`
	`  ENDIF`
	`ENDFUNCTION`
		7

Question	Answer	Marks
3(a)	1 mark for name of feature; 1 mark for description e.g. • Colouring code//Pretty printing • This is how the code is presented in the IDE e.g. colour coding and indentation • Context-sensitive prompts • Displays keywords or hints at the point of insertion e.g. drop-down list of commands • Auto-indent • Automatically indent your code for selection/iteration/procedures/methods • Auto-complete • Avoid typing errors // speeds up process of typing • Expand/collapse subroutines/code • To make it easier to view code currently working on	**4**
3(b)	1 mark per each correct bullet point • Normal • Abnormal / erroneous / invalid • Boundary / extreme	**3**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(a)(i)</td><td>1 mark for error and correction

Error 1 – <code>IF List[LowerBound] = SearchValue</code>
Correction – <code>IF List[MidPoint] = SearchValue</code>

Error 2 – <code>UpperBound ← MidPoint + 1</code>
Correction – <code>LowerBound ← MidPoint + 1</code>

Error 3 – <code>IF LowerBound > MidPoint</code>
Correction - <code>IF LowerBound > UpperBound</code>

Error 4 – <code>IF ValueFound = FALSE</code>
Correction – <code>IF ValueFound = TRUE</code></td><td>4</td></tr>
<tr><td>4(a)(ii)</td><td>Linear search</td><td>1</td></tr>
<tr><td>4(b)(i)</td><td>1 mark per shaded section

<table>
<tr><th rowspan="2">Count</th><th rowspan="2">TempValue</th><th rowspan="2">Sorted</th><th colspan="6">ArrayData</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>0</td><td>""</td><td>TRUE</td><td>5</td><td>20</td><td>12</td><td>25</td><td>32</td><td>29</td></tr>
<tr><td>1</td><td>12</td><td>FALSE</td><td></td><td>12</td><td>20</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>29</td><td>(FALSE)</td><td></td><td></td><td></td><td></td><td>29</td><td>32</td></tr>
<tr><td>0</td><td></td><td>TRUE</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>4</td></tr>
</table>

Question	Answer	Marks

4(b)(ii)	1 mark per bullet point
	
	• Initialising a counter variable to 0 **and** must be the same variable used to access array elements
	• While loop checking counter is < 5 or <= 4
	• Incrementing counter inside the loop and outside IF, **and** remainder of algorithm completed (The IF to ENDIF)
	
	e.g.
	
	`Count ← 0`
	`WHILE Count < 5`
	`  IF ArrayData[Count] > ArrayData[Count + 1]`
	`  THEN`
	`  TempValue ← ArrayData[Count + 1]`
	`  ArrayData[Count + 1] ← ArrayData[Count]`
	`  ArrayData[Count] ← TempValue`
	`  Sorted ← False`
	`  ENDIF`
	`  Count ← Count + 1`
	`ENDWHILE`
		3

4(b)(iii)	Bubble sort	**1**
4(b)(iv)	One from: • Insertion sort • Merge sort • Quick sort	**1**