
COMPUTER SCIENCE

9608/41

Paper 4 Written Paper

May/June 2018

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 2018 series for most Cambridge IGCSE™, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

IGCSE™ is a registered trademark.

This document consists of **19** 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).

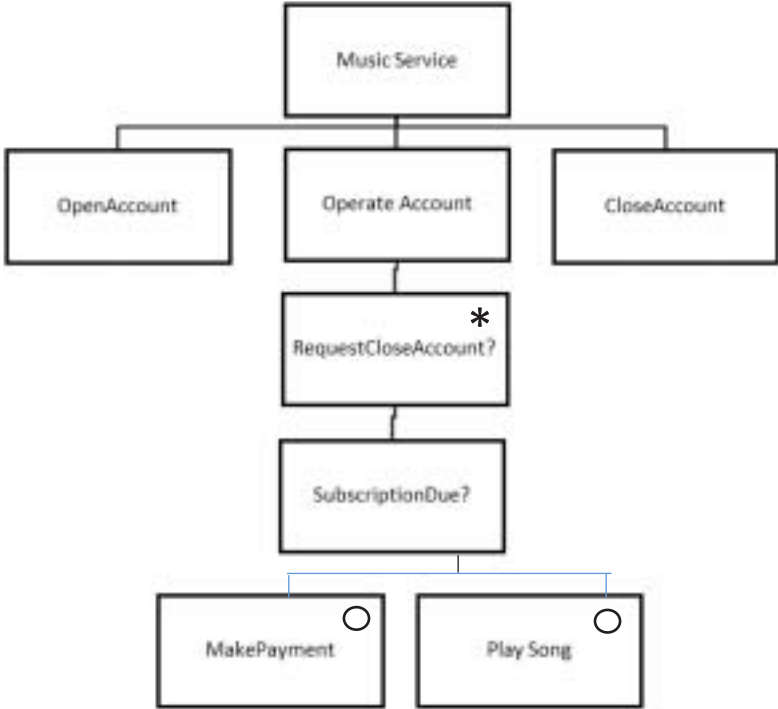
GENERIC MARKING PRINCIPLE 6:

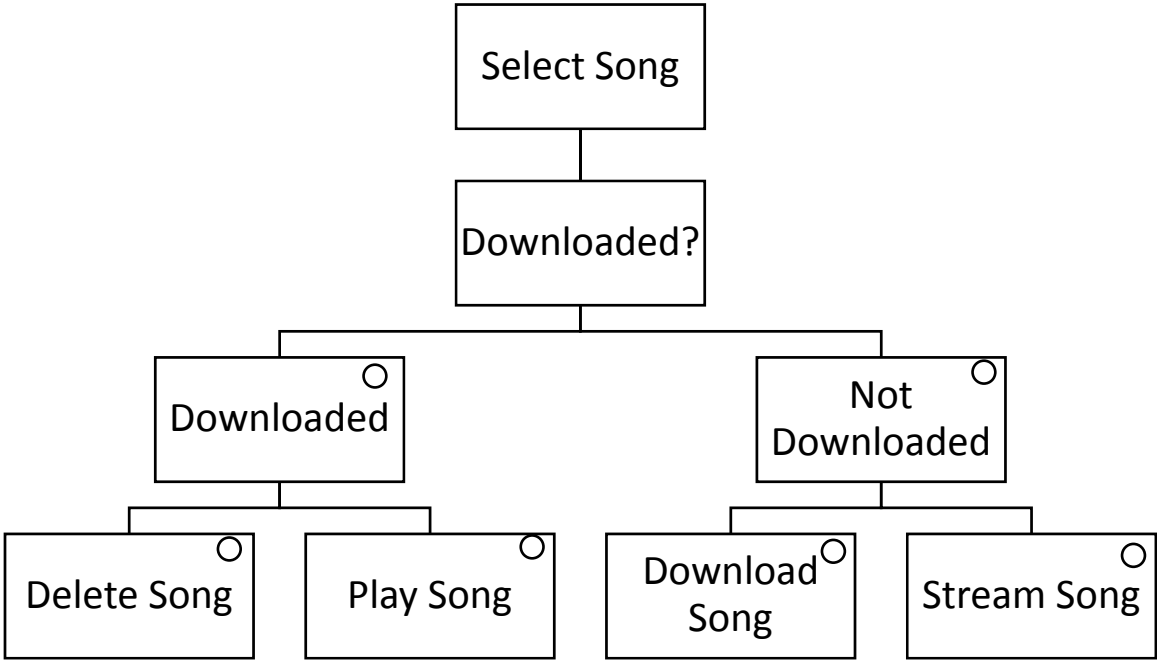
Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

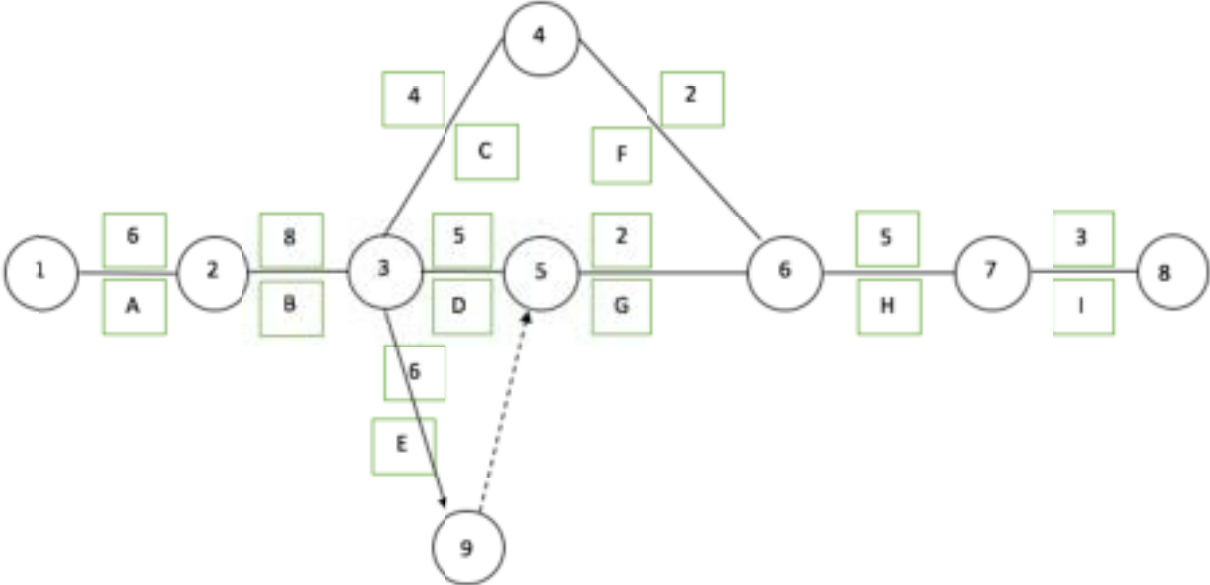
Question	Answer	Marks
1(a)	1 mark per fact 14 direct(london, rome). 15 flies(rome, british_air).	2
1(b)	1 mark per bullet: <ul style="list-style-type: none"> • palma • salzburg K = palma, salzburg	2
1(c)	1 mark per bullet: <ul style="list-style-type: none"> • direct • glasgow, M direct(glasgow, M) .	2
1(d)	1 mark per bullet: <ul style="list-style-type: none"> • flies(Z, X) • AND • direct(Z, Y) flies(Z, X) AND direct(Z, Y)	3
1(e)	YES	1

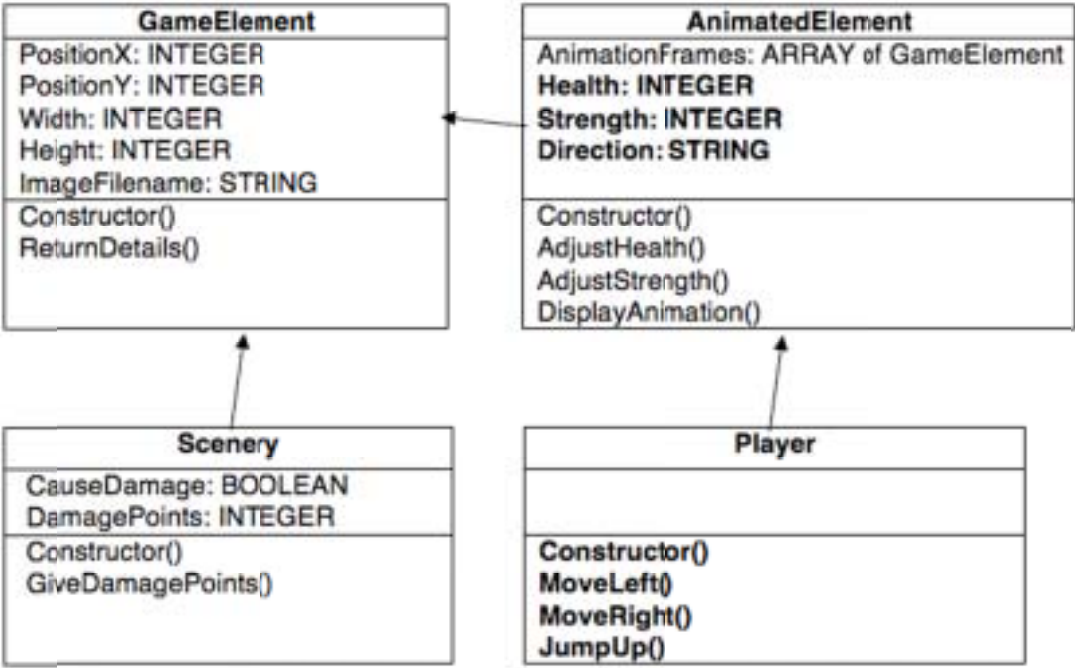
Question	Answer	Marks
2(a)	<p>1 mark for each completed statement</p> <pre> 01 MaxIndex ← 20 02 NumberItems ← MaxIndex - 1 // 19 03 FOR Outer ← 1 TO MaxIndex - 1 // 19 04 FOR Inner ← 1 to NumberItems 05 IF ItemList[Inner] > ItemList[Inner + 1] 06 THEN 07 Temp ← ItemList[Inner] 08 ItemList[Inner] ← ItemList[Inner + 1] 09 ItemList[Inner + 1] ← Temp 10 ENDIF 11 ENDFOR 12 NumberItems ← NumberItems - 1 13 ENDFOR </pre>	7
2(b)(i)	<p>1 mark per bullet</p> <ul style="list-style-type: none"> • Iterations continue // it continues doing comparisons • ...after the array is sorted 	2
2(b)(ii)	<p>1 mark per bullet to max 3</p> <ul style="list-style-type: none"> • Use of a flag to indicate if any swaps have taken place • If the inner loop has made all comparisons with no changes • ...flag/value set accordingly • A comparison checks the flag/value at the end of each inner loop • ...if it is sorted it breaks out/stops 	3

Question	Answer	Marks
2(c)	1 mark per bullet to max 4 e.g. <ul style="list-style-type: none">• When the list is almost sorted ...• ...because it will stop as soon as it is sorted • When there are a large number of data items ...• ...because it will perform fewer comparisons/loops	4

Question	Answer	Marks
<p>3(a)</p>	<p>1 mark per bullet</p> <ul style="list-style-type: none"> • OpenAccount, OperateAccount and Close Account on same level • RequestCloseAccount under OperateAccount • ...SubscriptionDue under RequestCloseAccount • ...Make Payment and PlaySong under SubscriptionDue • Correct selection and iteration throughout  <pre> classDiagram class MusicService class OpenAccount class OperateAccount class CloseAccount class RequestCloseAccount class SubscriptionDue class MakePayment class PlaySong MusicService -- OpenAccount MusicService -- OperateAccount MusicService -- CloseAccount OperateAccount -- RequestCloseAccount RequestCloseAccount -- SubscriptionDue SubscriptionDue -- MakePayment SubscriptionDue -- PlaySong </pre> <p>The diagram shows a class hierarchy for a Music Service. At the top is 'Music Service'. It has three subclasses: 'OpenAccount', 'Operate Account', and 'CloseAccount'. 'Operate Account' has a subclass 'RequestCloseAccount?' with an asterisk next to it. 'RequestCloseAccount?' has a subclass 'SubscriptionDue?'. 'SubscriptionDue?' has two subclasses: 'MakePayment' and 'Play Song', each with a small circle in its top right corner.</p>	<p>5</p>

Question	Answer	Marks
<p>3(b)</p>	<p>1 mark per bullet</p> <ul style="list-style-type: none"> • Downloaded? • Download and Not downloaded beneath downloaded • Delete song and play song beneath Downloaded (2) and Download song and stream song beneath not downloaded • All selections correct <div style="text-align: center;">  <pre> graph TD A[Select Song] --> B[Downloaded?] B --> C[Downloaded] B --> D[Not Downloaded] C --> E[Delete Song] C --> F[Play Song] D --> G[Download Song] D --> H[Stream Song] </pre> <p>The flowchart starts with a box labeled 'Select Song'. A vertical line connects it to a box labeled 'Downloaded?'. From 'Downloaded?', two lines branch out to 'Downloaded' (left) and 'Not Downloaded' (right). From 'Downloaded', two lines branch out to 'Delete Song' and 'Play Song'. From 'Not Downloaded', two lines branch out to 'Download Song' and 'Stream Song'. Each of the four bottom-level boxes has a small circle in its top right corner.</p> </div>	<p>4</p>

Question	Answer	Marks
4(a)	1 mark per bullet <ul style="list-style-type: none"> • C, D and E all coming from 3 • G following D and E • F following C • H from 6 to 7 • I from 7 to 8 	5
4(b)	1 mark per bullet <ul style="list-style-type: none"> • A→B→E→G→H→I • 30 days 	2
4(c)	1 mark per bullet <ul style="list-style-type: none"> • Earliest start time: 19 days • Latest finish time: 22 days 	2

Question	Answer	Marks
<p>5(a)</p>	<p>1 mark for each bullet:</p> <ul style="list-style-type: none"> • AnimatedElement attributes • Player methods • Inheritance arrows  <pre> classDiagram class GameElement { PositionX: INTEGER PositionY: INTEGER Width: INTEGER Height: INTEGER ImageFilename: STRING Constructor() ReturnDetails() } class AnimatedElement { AnimationFrames: ARRAY of GameElement Health: INTEGER Strength: INTEGER Direction: STRING Constructor() AdjustHealth() AdjustStrength() DisplayAnimation() } class Scenery { CauseDamage: BOOLEAN DamagePoints: INTEGER Constructor() GiveDamagePoints() } class Player { Constructor() MoveLeft() MoveRight() JumpUp() } GameElement < -- AnimatedElement Scenery < -- Player </pre>	<p>3</p>

Question	Answer	Marks
5(b)	<p>1 mark per bullet to max 6</p> <ul style="list-style-type: none"> • class declaration • private declaration of five attributes • constructor declaration • ...initialisation of attributes to the parameter values • declaration of <code>GetDetails</code> function • appropriate concatenation of string using attributes • return of all 5 values in one string <p>Python example code:</p> <pre> class GameElement: def __init__(self, PositionX, PositionY, Width, Height, ImageFilename): self.__PositionX = PositionX self.__PositionY = PositionY self.__Width = Width self.__Height = Height self.__ImageFilename = String def GetDetails(self): Message = "Position_x:", self.__PositionX, "Position_y:", self.__PositionY, "width:", self.__Width, "height:", self.__Height, "ImageFilename", self. __ImageFilename) return Message </pre>	6

Question	Answer	Marks
5(b)	<p>Visual Basic example code:</p> <pre> Class GameElement Private PositionX As Integer Private PositionY As Integer Private Width As Integer Private Height As Integer Private ImageFilename As String Public Sub New(ByVal X As Integer, ByVal Y As Integer, ByVal W As Integer, ByVal H As Integer, Filename As String) PositionX = X PositionY = Y Width = W Height = H ImageFilename = Filename End Sub Public Function GetDetails() Dim Message As String Message = "PositionX: " + PositionX + "PositionY: " + PositionY + ", width: " + Width + ", height: " + Height + ", ImageFilename:" + ImageFilename Return Message End Function End Class </pre>	

Question	Answer	Marks
5(b)	<p>Pascal example code:</p> <pre> type GameElement = class private PositionX : Integer; PositionY : Integer; Width : Integer; Height : Integer; ImageFilename : String; public Constructor init(X, Y, W, H:Integer; Filename: String); Function GetDetails() : String; end; Constructor GameElement.init(X, Y, W, H:Integer; Filename: String); begin PositionX := X; PositionY := Y; Width := W; Height := H; ImageFilename := Filename; end; Function GameElement.GetDetails() : String; var Message:String; begin Message = "PositionX: " + PositionX + "PositionY: " + PositionY + ", width: " + Width + ", height: " + Height + ", ImageFilename:" + ImageFilename; Result = Message end; </pre>	

Question	Answer	Marks
5(c)	<p>Max 4 from each section to max 6 overall</p> <p>1 mark per bullet to max 4</p> <ul style="list-style-type: none"> • class declaration with inheritance • constructor declaration • ...taking all 5 parameters and CauseDamage, DamagePoints parameters • ...with inheritance constructor call • Declaring CauseDamage, DamagePoints private and assigning parameters <p>1 mark per bullet to max 4</p> <ul style="list-style-type: none"> • Function declaration for GiveDamagePoints ... • ...checking if CauseDamage = True • ...returning DamagePoints if true • ...else returning appropriate value e.g. -1/null/blank <p>Python example code:</p> <pre> class Scenery(GameElement): def __init__(self, PositionX, PositionY, Width, Height, ImageFilename, CauseDamage, DamagePoints): Object.__init__(self, PositionX, PositionY, Width, Height, ImageFilename) self.__CauseDamage = CauseDamage self.__DamagePoints = DamagePoints def GiveDamagePoints(self): if(self.__CauseDamage): return self.__DamagePoints else: return 0 </pre>	6

Question	Answer	Marks
5(c)	<p>Visual Basic example code:</p> <pre> Class Scenery Inherits GameElement Private CauseDamage As Boolean Private DamagePoints As Integer Public Sub New(ByVal X As Integer, ByVal Y As Integer, ByVal W As Integer, ByVal H As Integer, Filename As String, ByVal CD As Boolean, ByVal DP As Integer) MyBase.New(X, Y, W, H, Filename) CauseDamage = CD DamagePoints = DP End Sub Public Function GiveDamagePoints() As Integer If (CauseDamage) Then Return DamagePoints Else Return 0 End if End Function End Class </pre>	

Question	Answer	Marks
5(c)	<p>Pascal example code:</p> <pre> Scenery = class(GameElement) private CauseDamage : Boolean; DamagePoints: Integer; public Constructor init(X, Y, W, H: Integer; Filename: String; CD:Boolean; DP: Integer); override; Function GiveDamagePoints() : Integer; end; constructor Scenery.init(X, Y, W, H: Integer; Filename: String; CD: Boolean; DP: Integer); begin inherited init(X, Y, W, H, Filename); CauseDamage := CD; DamagePoints := DP; end; Function Scenery.GiveDamagePoints() : Integer; begin if (CauseDamage): Result := DamagePoints else: Result := 0; end; </pre>	

Question	Answer	Marks
5(d)(i)	<p>1 mark per bullet</p> <ul style="list-style-type: none"> • Variable <code>GiftBox</code> assigned value • Call <code>Scenery</code> • With all 7 parameters assigned correctly <p>Python example code:</p> <pre>GiftBox = Scenery(150, 150, 50, 75, "box.png", True, 50)</pre> <p>Visual Basic example code:</p> <pre>GiftBox = Scenery(150, 150, 50, 75, "box.png", True, 50)</pre> <p>Pascal example code:</p> <pre>GiftBox := Scenery(150, 150, 50, 75, "box.png", True, 50)</pre>	3

Question	Answer	Marks
5(d)(ii)	<p>1 mark per bullet</p> <ul style="list-style-type: none"> • Function declaration with no parameters • Use inherited <code>GetDetails</code> method to get string • Return all values <pre>def GetScenery(self): Message = Object.GetDetails(self) Message = Message + " Causes Damage:", self.CauseDamage, "Damage Points:", self.DamagePoints return Message</pre> <p>Visual Basic example code:</p> <pre>Public Function GetScenery() As String Dim Message As String Message = MyBase.GetDetails() Message = Message + "CauseDamage: " + CauseDamage + " DamagePoints: " + DamagePoints Return Message End Function</pre> <p>Pascal example code:</p> <pre>Function Secenery.GetScenery(): String Var Message : String Begin Message := GetDetails(); Message := Message + "CauseDamage: " + CauseDamage + " DamagePoints: " + DamagePoints; Result:=Message; End;</pre>	3

Question	Answer	Marks
6(a)(i)	1 mark per bullet: <ul style="list-style-type: none"> • TYPE ListNode declaration and ENDTYPE • DECLARE Player : String • DECLARE Pointer : INTEGER <pre> TYPE ListNode DECLARE Player : STRING DECLARE Pointer : INTEGER ENDTYPE </pre>	3
6(a)(ii)	1 mark per bullet: <ul style="list-style-type: none"> • DECLARE Scorers : ARRAY[0:9] • OF ListNode <pre> DECLARE Scorers : ARRAY[0:9] OF ListNode </pre>	2
6(b)	1 mark for each completed statement <pre> FUNCTION SearchList(Find, Position) RETURNS INTEGER IF Scorer[Position].Player = Find THEN RETURN Position ELSE IF Scorer[Position].Player <> -1 THEN Position ← SearchList(Find, Scorer[Position].Pointer) RETURN Position ELSE RETURN 99 ENDIF ENDIF ENDPROCEDURE </pre>	5