Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

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.

© UCLES 2018

Cambridge Assessment International Education

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.

© UCLES 2018

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 2018

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: • palma • salzburg `K = palma, salzburg`	**2**
1(c)	1 mark per bullet: • direct • glasgow, M `direct(glasgow, M).`	**2**
1(d)	1 mark per bullet: • `flies(Z,X)` • `AND` • `direct(Z, Y)` `flies(Z, X) AND direct(Z, Y)`	**3**
1(e)	`YES`	**1**

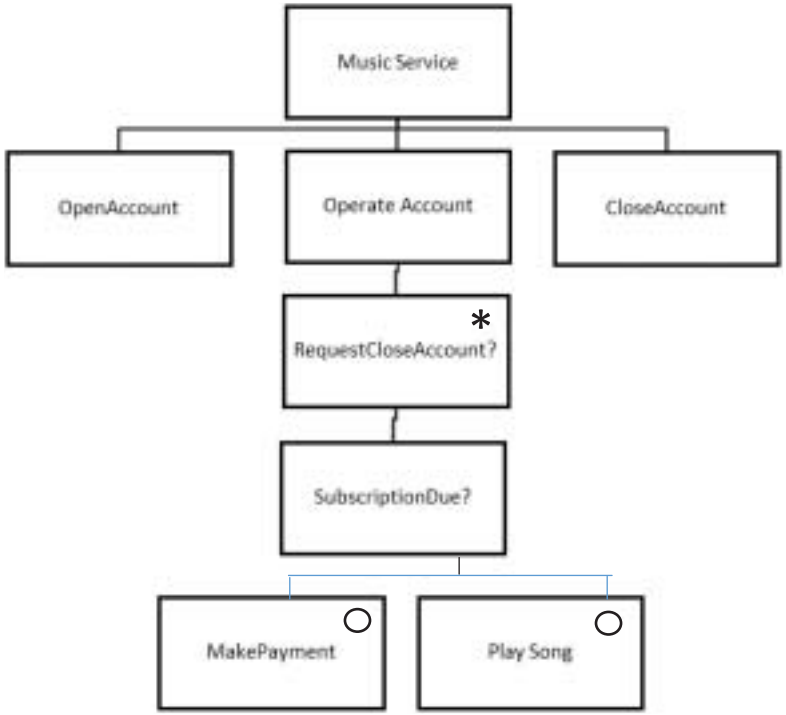
Question	Answer	Marks

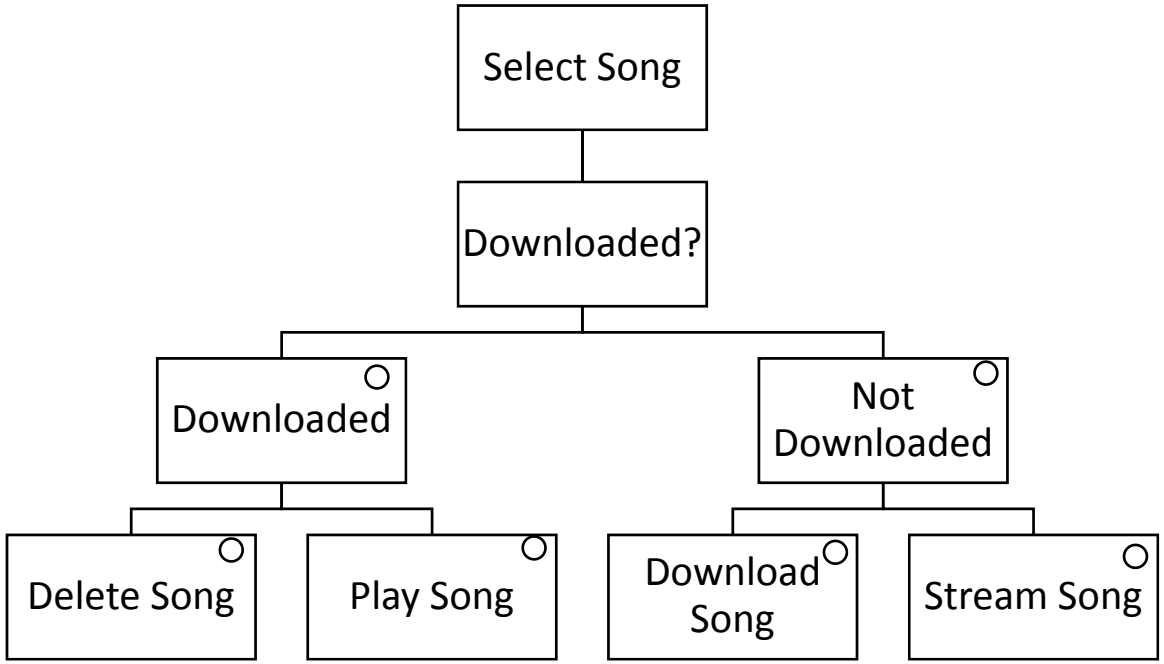
2(a)	1 mark for each completed statement
	`MaxIndex ← 20`
	`NumberItems ← MaxIndex - 1 // 19`
	`FOR Outer ← 1 TO MaxIndex - 1 // 19`
	`  FOR Inner ← 1 to NumberItems`
	`  IF ItemList[Inner] > ItemList[Inner + 1]`
	`  THEN`
	`  Temp ← ItemList[Inner]`
	`  ItemList[Inner] ← ItemList[Inner + 1]`
	`  ItemList[Inner + 1] ← Temp`
	`  ENDIF`
	`  ENDFOR`
	`  NumberItems ← NumberItems - 1`
	`ENDFOR`
		7

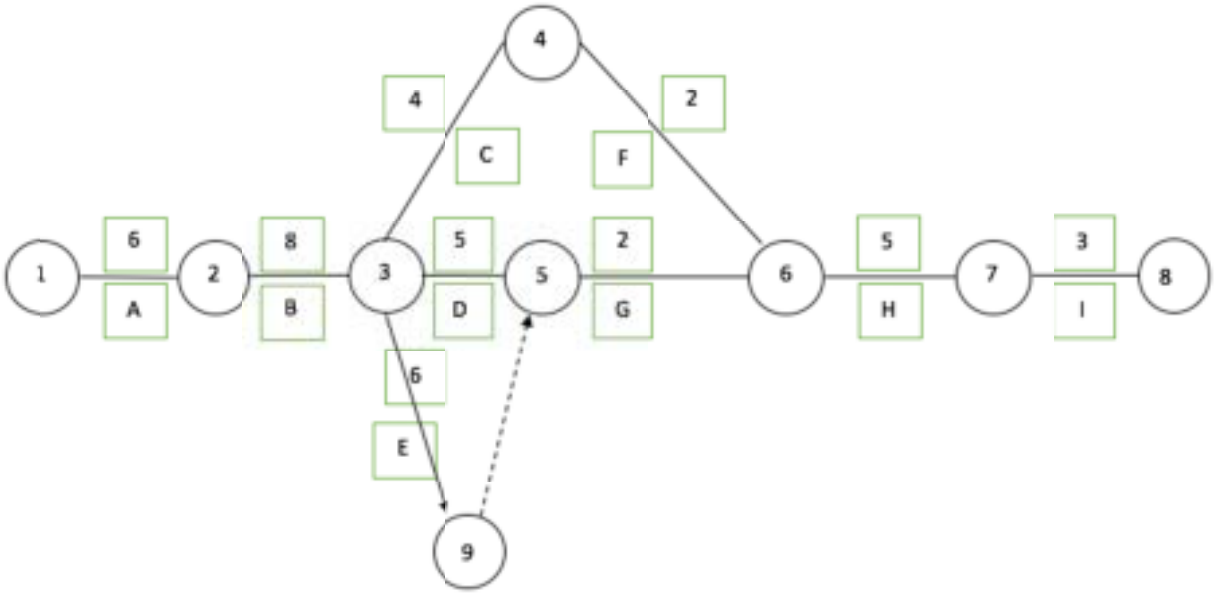
2(b)(i)	1 mark per bullet • Iterations continue // it continues doing comparisons • …after the array is sorted	**2**
2(b)(ii)	1 mark per bullet to max 3 • Use of a flag to indicate if any swaps have taken place • If the inner loop has made all comparisons with no changes • …flag/value set accordingly • A comparison checks the flag/value at the end of each inner loop • …if it is sorted it breaks out/stops	**3**

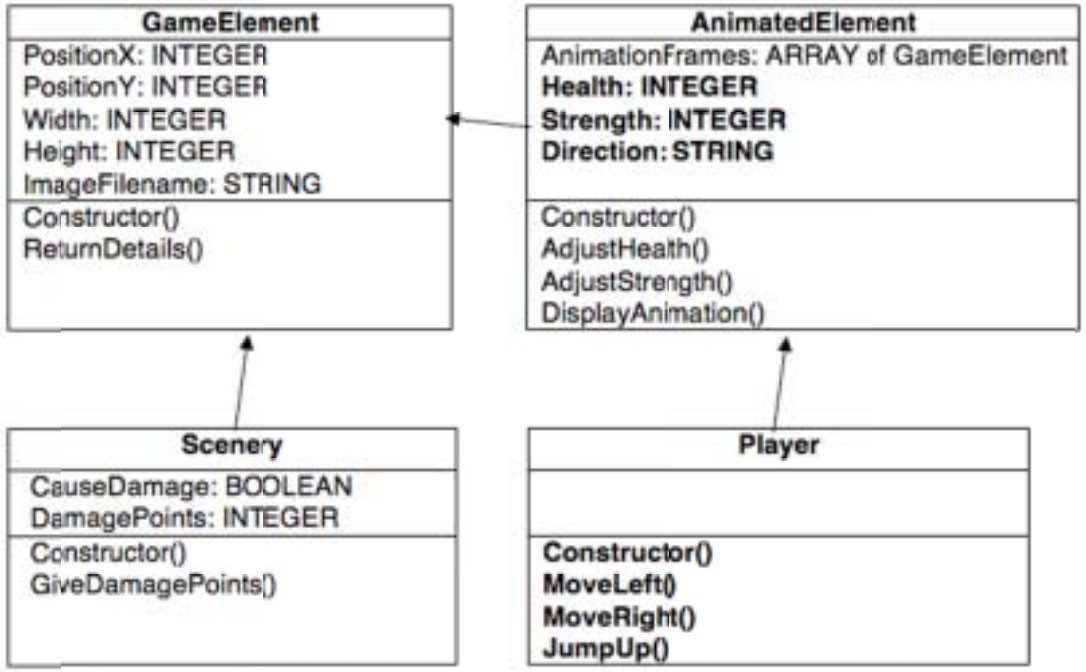
Question	Answer	Marks
2(c)	1 mark per bullet to max 4 e.g. • When the list is **almost** sorted … • …because it will stop as soon as it is sorted • When there are a large number of data items … • …because it will perform fewer comparisons/loops	**4**

© UCLES 2018

Question	Answer	Marks
3(a)	1 mark per bullet • OpenAccount, OperateAccount and Close Account on same level • RequestCloseAccount under OperateAccount • …SubscriptionDue under RequestCloseAcount • …Make Payment and PlaySong under SubscriptionDue • Correct selection and iteration throughout  	5

Question	Answer	Marks
3(b)	1 mark per bullet • 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  	**4**

Question	Answer	Marks
4(a)	1 mark per bullet • 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 • A→B→E→G→H→I • 30 days	2
4(c)	1 mark per bullet • Earliest start time: 19 days • Latest finish time: 22 days	2

Question	Answer	Marks
5(a)	1 mark for each bullet: • AnimatedElement attributes • Player methods • Inheritance arrows  	3

Question	Answer	Marks

5(b)	1 mark per bullet to max 6
	
	• class declaration
	• private declaration of five attributes
	• constructor declaration
	• …initialisation of attributes to the parameter values
	• declaration of `GetDetails` function
	• appropriate concatenation of string using attributes
	• return of all 5 values in one string
	
	Python example code:
	
	`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`
		6

Question	Answer	Marks
5(b)	Visual Basic example code:	

```
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
```

Question	Answer	Marks

5(b)	Pascal example code:
	
	`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;`	

Question	Answer	Marks

5(c)	Max 4 from each section to max 6 overall
	
	1 mark per bullet to max 4
	
	• 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
	
	1 mark per bullet to max 4
	
	• Function declaration for `GiveDamagePoints` …
	• …checking if `CauseDamage = True`
	• …returning `DamagePoints` if true
	• …else returning appropriate value e.g. -1/null/blank
	
	Python example code:
	
	`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`
		6

Question	Answer	Marks

5(c)	Visual Basic example code:
	
	`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`	

Question	Answer	Marks

5(c)	Pascal example code:
	
	`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;`	

Question	Answer	Marks

5(d)(i)	1 mark per bullet
	• Variable `GiftBox` assigned value
	• Call `Scenery`
	• With all 7 parameters assigned correctly
	
	Python example code:
	`GiftBox = Scenery(150, 150, 50, 75, "box.png", True, 50)`
	
	Visual Basic example code:
	`GiftBox = Scenery(150, 150, 50, 75, "box.png", True, 50)`
	
	Pascal example code:
	`GiftBox := Scenery(150, 150, 50, 75, "box.png", True, 50)`
		3

Question	Answer	Marks

5(d)(ii)	1 mark per bullet
	• Function declaration with no parameters
	• Use inherited `GetDetails` method to get string
	• Return all values
	
	`def GetScenery(self):`
	`  Message = Object.GetDetails(self)`
	`  Message = Message + " Causes Damage:", self.CauseDamage, "Damage`
	`  Points:", self.DamagePoints`
	`  return Message`
	
	Visual Basic example code:
	
	`Public Function GetScenery() As String`
	`  Dim Message As String`
	`  Message = MyBase.GetDetails()`
	`  Message = Message + "CauseDamage: " + CauseDamage +  "`
	`  DamagePoints: " + DamagePoints`
	`  Return Message`
	`End Function`
	
	Pascal example code:
	
	`Function Secenery.GetScenery(): String`
	`  Var Message : String`
	`  Begin`
	`  Message := GetDetails();`
	`  Message := Message + "CauseDamage: " + CauseDamage +  "`
	`  DamagePoints: " + DamagePoints;`
	`  Result:=Message;`
	`  End;`
		3

Question	Answer	Marks

6(a)(i)	1 mark per bullet:
	• `TYPE ListNode` declaration and `ENDTYPE`
	• `DECLARE Player : String`
	• `DECLARE Pointer : INTEGER`
	
	`TYPE ListNode`
	`  DECLARE Player : STRING`
	`  DECLARE Pointer : INTEGER`
	`ENDTYPE`
		3

6(a)(ii)	1 mark per bullet: • `DECLARE Scorers : ARRAY[0:9]` • `OF ListNode` `DECLARE Scorers : ARRAY[0:9] OF ListNode`	**2**

6(b)	1 mark for each completed statement
	
	`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`
		5