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 2018**
MARK SCHEME	
Maximum Mark: 75	

Published

This document consists of **21** printed pages.

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.

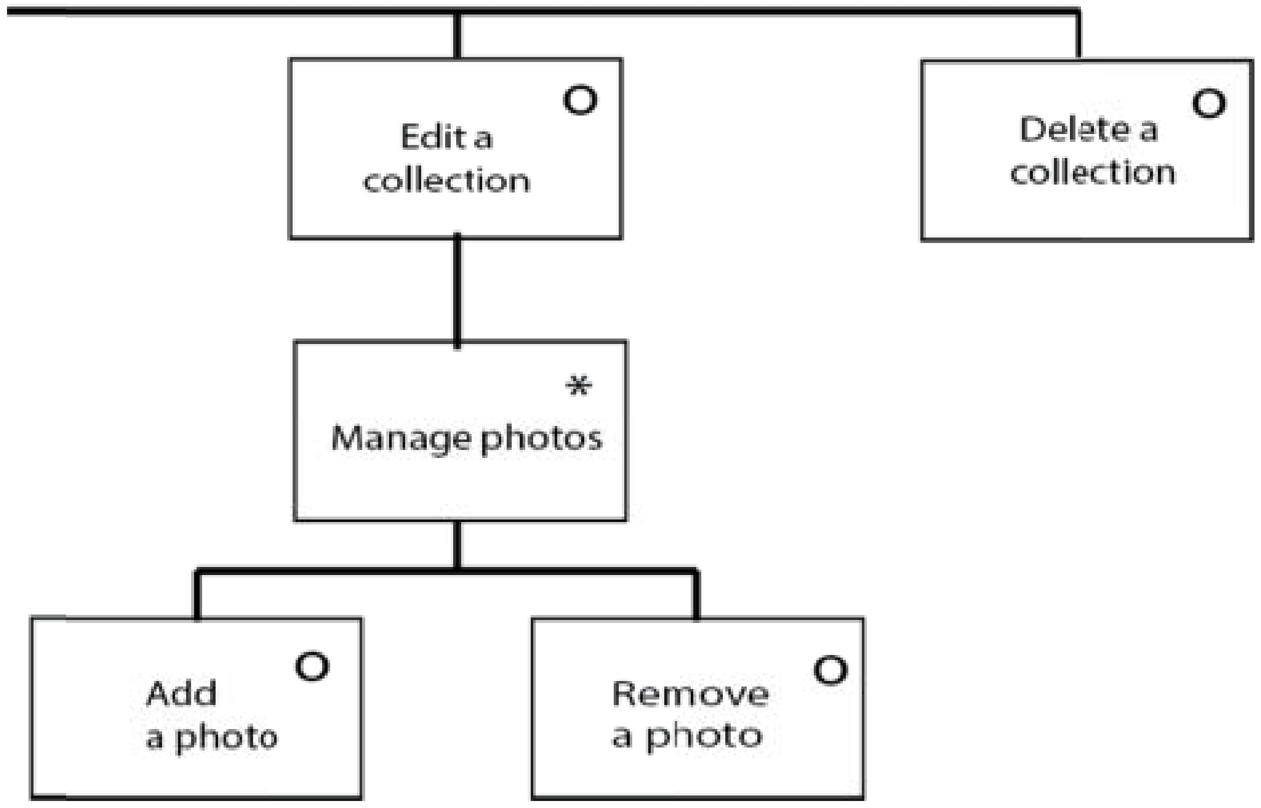
GENERIC MARKING PRINCIPLE 4:

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

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 for each correctly completed pseudocode line to max 4	**4**

```
01       REPEAT
02          CALL TakePhoto
03          CALL AddPhotoToCollection
04          OUTPUT "Do you want to take another photo?"
05          INPUT AddPhoto
06       UNTIL AddPhoto = "No"
07       REPEAT
08          CALL AddUser
09          OUTPUT "Do you want to add another user?"
10          INPUT NewUser
11       UNTIL NewUser = "No"
12       OUTPUT "Do you want to create another collection?"
13       INPUT NewCollection
14    UNTIL NewCollection = "No"
```

Question	Answer	Marks
1(b)	1 mark per bullet: • Edit a collection // Choose a collection // Select a collection • Manage photos • Delete a collection • both add and remove a photo // add to collection, delete to collection • appropriate selection and iteration in all boxes  Edit a collection O Delete a collection O Manage photos * Add a photo O Remove a photo O	**5**

Question	Answer	Marks

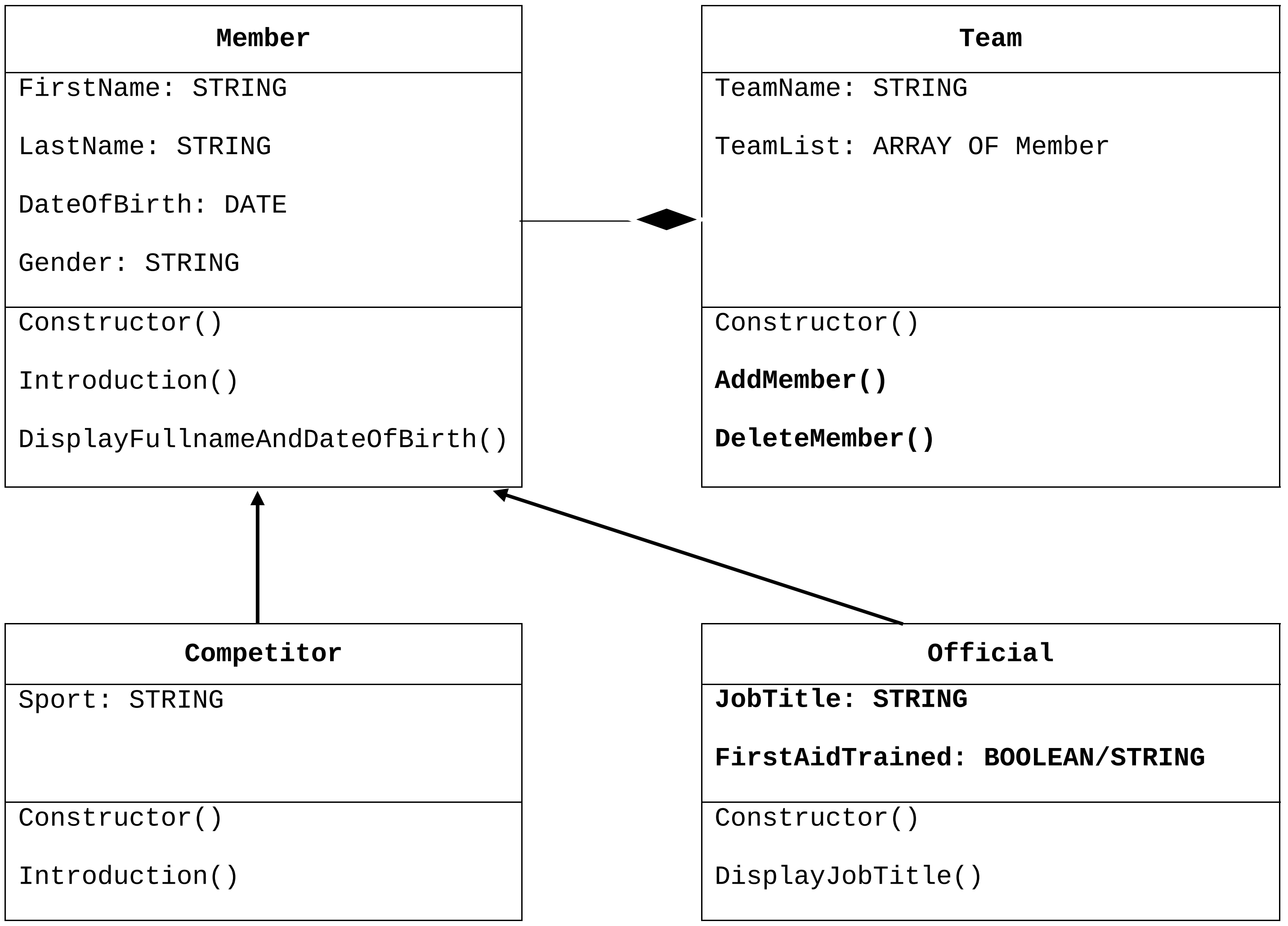
2(a)	1 mark for each statement
	
	`15   is_a(gecko, lizard).`
	`16   maxsize(gecko, 182).`
		2

2(b)	1 mark for 2 results 2 marks for 3 correct results `green_iguana, cayman, smooth_iguana`	**2**
2(c)	1 mark per bullet • `is_a` used with brackets () • `squamata, X` in correct order `is_a(squamata, X).`	**2**
2(d)	1 mark for each bullet to max 3 • `is_a(X, Z)` • `and // , has(Z, Y).` `is_a(X, Z) AND has(Z, Y).`	**3**
2(e)	YES	**1**

Question	Answer	Marks
3(a)	CardData is **partially** sorted/ordered // more items in order/sorted	**1**
3(b)	1 mark for each correct statement 01 ArraySize ← 10 02 FOR Pointer ← 2 TO **ArraySize // 10** 03 ValueToInsert ← CardData[Pointer] 04 HolePosition ← **Pointer** 05 WHILE(HolePosition>1 AND(**CardData[HolePosition - 1]** > **ValueToInsert**)) 06 CardData[HolePosition] ← CardData[**HolePosition – 1**] 07 HolePosition ← **HolePosition – 1** 08 ENDWHILE 09 CardData[HolePosition] ← **ValueToInsert** 10 ENDFOR	**7**
3(c)(i)	1 mark per bullet to max 2 • It doesn't check every value • The midpoint is the middle element, not the middle numerical value • When the higher/lower elements are discarded they will not be the higher/lower elements • It might discard the value you are looking for	**2**
3(c)(ii)	1 mark per bullet to max 4. Max 2 marks if no relation to CardData values. • Find mid-point **and** comparison // 25 is smaller than/compared to 52/56 • Discard/ignore greater // change upper bound to 33/52/midpoint - 1 //e.g. right hand side // only use array elements 1 - 4/5 • Find **and** compare to mid-point of new list e.g. 12/25 • Value is the midpoint // Continue until value found	**4**

Question	Answer	Marks
3(d)	1 mark for each complete statement	**4**

```
PROCEDURE BinarySearch(CardData, SearchValue)
   DECLARE Midpoint : INTEGER
   First ← 1
   Last ← ARRAYLENGTH(CardData)
   Found ← FALSE
   WHILE (First <= Last) AND NOT(Found)
      Midpoint ← (First + Last) \ 2
      IF CardData[Midpoint] = SearchValue
         THEN
            Found ← TRUE
         ELSE
            IF SearchValue < CardData[Midpoint]
               THEN
                  Last ← Midpoint - 1
               ELSE
                  First ← Midpoint + 1
            ENDIF
      ENDIF
   ENDWHILE
ENDPROCEDURE
```

Question	Answer	Marks
4(a)	1 mark per bullet: • Team methods • Official attributes • Two inheritance arrows **or** containment  **Member** FirstName: STRING LastName: STRING DateOfBirth: DATE Gender: STRING Constructor() Introduction() DisplayFullnameAndDateOfBirth() **Team** TeamName: STRING TeamList: ARRAY OF Member Constructor() **AddMember()** **DeleteMember()** **Competitor** Sport: STRING Constructor() Introduction() **Official** **JobTitle: STRING** **FirstAidTrained: BOOLEAN/STRING** Constructor() DisplayJobTitle()	**3**

Question	Answer	Marks

4(b)	1 mark per bullet to max 5
	
	• class declaration
	• `FirstName`, `LastName`, `DateOfBirth` and Gender all defined as private
	
	• constructor declaration
	• …all four attributes assigned values from parameters
	
	• (Public) method for `Introduction`
	• …outputs message with `FirstName` and `LastName` attributes // returns `FirstName` and `LastName`
	
	• Public method for `DisplayFullNameAndDateofbirth`
	• … outputs message with `FirstName`, `LastName` and `DateOfBirth` // returns `FirstName`, `LastName`, `DateOfBirth`
	
	Python example code:
	
	`class Member:`
	
	`  def __init__(self, Fname, Lname, DOB, GenderVal):`
	`  self.__FirstName = Fname`
	`  self.__LastName = Lname`
	`  self.__DateOfBirth = DOB`
	`  self.__Gender = GenderVal`
	
	`  def Introduction(self):`
	`    return "Hello, I am ", self.__FirstName, " ",`
	`                 self.LastName`
	
	`  def DisplayFullnameAndDateofbirth(self):`
	`    print self.__FirstName, self.__LastName,`
	`                 self.__DateOfBirth`
		5

Question	Answer	Marks

4(b)	Visual Basic example code:
	
	`Class Member`
	`  Private  Firstname As String`
	`  Private  Lastname As String`
	`  Private  DateOfBirth As Date`
	`  Private  Gender As String`
	
	`  Public Sub New(ByVal Fname As String,ByVal Lname As String,`
	`                ByVal DOB As Date, ByVal GenderVal As String)`
	`    Firstname = Fname`
	`    Lastname = Lname`
	`    DateOfBirth = DOB`
	`    Gender = GenderVal`
	`  End Sub`
	
	`  Public Function Introduction() As String`
	`    Dim Message As String`
	`    Message = "Hello, I am " + Firstname + " " + Lastname + " " +`
	`                DateOfBirth`
	`    Return Message`
	`  End Function`
	
	`  Public Function DisplayFullNameAndDateOfBirth As String`
	`    DisplayFullNameAndDateOfBirth = Firstname + " " + Lastname +`
	`                                    " " + DateOfBirth`
	`  End Function`
	`End Class`	

Question	Answer	Marks

4(b)	Pascal example code:
	
	`type`
	`Member = class`
	`private`
	`  Firstname : string;`
	`  Lastname : string;`
	`  DateOfBirth : date;`
	`  Gender : string;`
	
	`public`
	`  constructor Create(Fname, Lname, Gend, DBirth : string);`
	`  function Introduction() : string;`
	`  function DisplayFullNameAndDateOfBirth() : string;`
	
	`constructor Member.Create(Fname, Lname, Gend, DBirth : string);`
	`  begin`
	`    Firstname := Fname`
	`    LastName := Lname`
	`    Gender := Gend`
	`    DateOfBirth := DBirth`
	`  end;`
	
	`function Member.Introduction() : String;`
	`  begin`
	`    Introduction :=  "Hello, I am " + Firstname + " " + Lastname`
	`  end;`
	
	`function Member.DisplayFullNameAndDateOfBirth As String;`
	`  begin;`
	`    DisplayFullNameAndDateOfBirth = Firstname + " " + Lastname +`
	`                                     " " + DateOfBirth`
	`  end;`	

Question	Answer	Marks

4(c)	1 mark per bullet to max 5
	
	• Class declaration that inherits from Member
	• Constructor declaration taking all five parameters
	• …that inherits from Member
	• Declaration of Sport as private String
	• …. and assigning to parameter
	• Introduction method declaration (with polymorphism)
	• …returning/outputting message with `FirstName`, `LastName` and `Sport` variables
	
	Python example code:
	
	`class Competitor(Member):`
	`  def __init__(self, Fname, Lname, DOB, GenderVal, MySport):`
	`    Member.__init__(self, Fname, Lname, DOB, GenderVal)`
	`    self.__Sport = MySport`
	
	`  def Introduction(self):`
	`    print "Hello, I am %s %s and my sport is %s" % (self.FirstName,`
	`           self.LastName, self.__Sport)`
		5

Question	Answer	Marks

4(c)	Visual Basic example code:
	
	`Class Competitor`
	`  Inherits Member`
	`  Private Sport As String`
	`  Public Sub New(ByVal Fname As String,ByVal Lname As String,`
	`               ByVal DOB As Date, ByVal GenderVal As String, ByVal`
	`               SportVal As String)`
	`    MyBase.New(Fname, Lname, DOB, GenderVal)`
	`    Sport = SportVal`
	`  End Sub`
	
	`  Public Overloads Function Introduction() As String`
	`    Dim Message As String`
	`    Message = "Hello, I am " + Firstname + " " + Lastname + " and my`
	`              sport is " + Sport`
	`    Return Message`
	`  End Function`
	`End Class`	

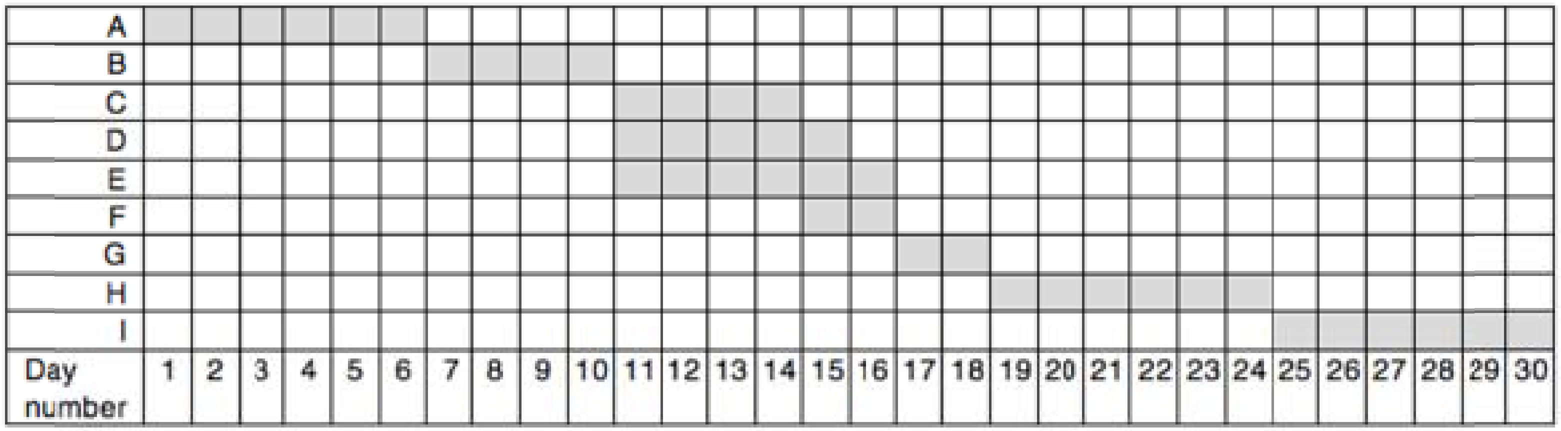
Question	Answer	Marks

4 (c)	Pascal example code:
	
	`type`
	`  Competitor = class(Member)`
	`  private`
	`    Sport : String;`
	`  public`
	`    Constructor init(Fname, Lname: String; DOB: Date; GenderVal,`
	`                   Sport:String);`
	`    Function Introduction() : String;`
	`end;`
	
	`Constructor Competitor.initFname, Lname: String; DOB: Date; GenderVal, SportVal:String);`
	`begin`
	`  inherited init(Fname, Lname, DOB, GenderVal);`
	`  Sport := SportVal;`
	`end;`
	
	`Function Competitor.Introduction();`
	`begin`
	`  Result:= "Hello, I am " + Firstname + " " + Lastname + " and my sport`
	`            is " + Sport`
	`end;`	

Question	Answer	Marks

4(d)	1 mark per bullet
	
	• variable BMXJudge assigned value
	• call Official
	• with all 6 parameters assigned correctly
	
	Python example code:
	
	`BMXJudge = Official("Omar", "Ellaboudy", "17/03/1993", "Male", true, "Judge")`
	
	Visual Basic example code:
	
	`BMXJudge = New Official("Omar", "Ellaboudy", "17/03/1993", "Male, true, "Judge")`
	
	Pascal example code:
	
	`BMXJudge := Official("Omar", "Ellaboudy", "17/03/1993", "Male", true, "Judge")`
		3

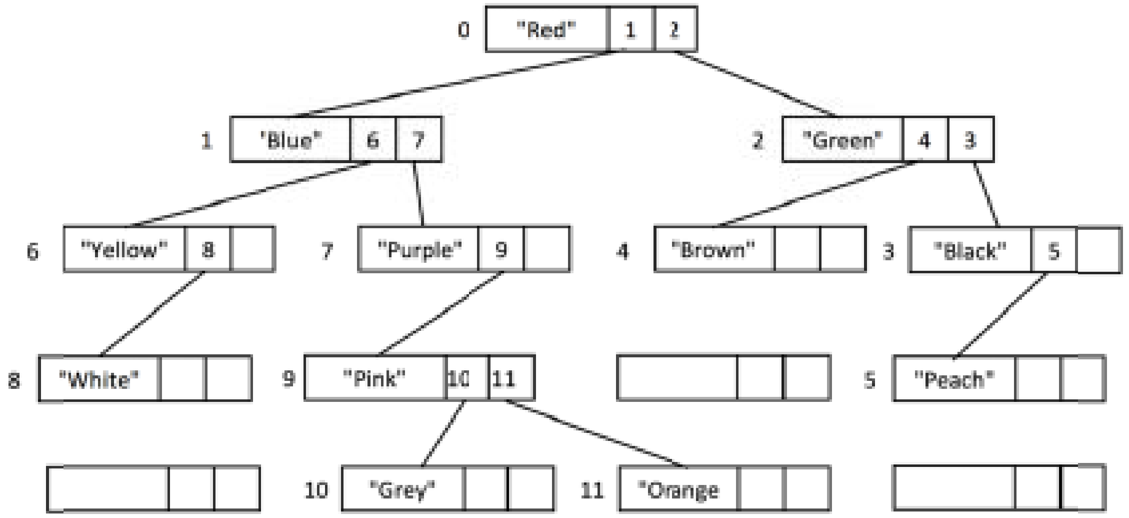
Question	Answer	Marks
5(a)	1 mark per bullet • **C**/**D**/**E** in parallel starting after **B**, with correct durations. • **F** with dependency on **C** and correct duration • **G** with dependency on **D** and **E** with correct duration • **H** with correct dependency on **F** and **G** • **I** with dependency on **H**	**5**



A	▨	▨	▨	▨	▨	▨																								
B							▨	▨	▨	▨																				
C											▨	▨	▨	▨																
D											▨	▨	▨	▨	▨															
E											▨	▨	▨	▨	▨	▨														
F															▨	▨														
G																	▨	▨												
H																			▨	▨	▨	▨	▨	▨						
I																									▨	▨	▨	▨	▨	▨
Day number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Question	Answer	Marks
5(b)(i)	**C, D, E**	**1**
5(b)(ii)	**E, F**	**1**

Question	Answer	Marks
5(c)	1 mark per bullet to max 2 For example: • Check if the project is on track • …so the project manager can intervene if behind • lets you identify slack time to • …reallocate resources to support the process • find critical path • …to ensure activities are given correct priority • see when tasks end • ...to plan the next tasks • see which tasks can run in parallel • set milestones/goals • check correct tasks are being carried out on current day • Calculate latest start time • Calculate earliest finish time • Calculate latest start time for a task	**2**

Question	Answer	Marks
6(a)	1 mark per bullet • Brown left and black right from node 2 • Yellow left and Purple right from node 1 • Peach left comes from 3 • White left from 6 **and** Pink left from 7 • Grey left from 9 and orange right from 9  0 "Red" 1 2 1 "Blue" 6 7 2 "Green" 4 3 6 "Yellow" 8 7 "Purple" 9 4 "Brown" 3 "Black" 5 8 "White" 9 "Pink" 10 11 5 "Peach" 10 "Grey" 11 "Orange"	**5**

Question	Answer	Marks

6(b)	1 mark for outputting all the leaf data values
	
	• Outputting `BinaryTree[CurrentNode].DataValue` **only** when both `LeftPointer` and `RightPointer` are -1
	
	1 mark per bullet to max 7
	
	• Function declaration
	• …taking `CurrentNode` or equivalent as parameter
	
	• Check if `BinaryTree[CurrentNode].LeftPointer` is not -1 …
	• … recursive call …
	• …with left pointer value as parameter
	
	• Check if `BinaryTree[CurrentNode].RightPointer` is not -1…
	• … recursive call…
	• … with right pointer value as parameter
	
	Python example code:
	
	`def FindLeaves(CurrentNode):`
	`  global BinaryTree`
	`  if(BinaryTree[CurrentNode].LeftPointer != -1):`
	`    FindLeaves(BinaryTree[CurrentNode].LeftPointer)`
	
	`  if(BinaryTree[CurrentNode].RightPointer != -1):`
	`    FindLeaves(BinaryTree[CurrentNode].RightPointer)`
	
	`  if((BinaryTree[CurrentNode].RightPointer == -1) and`
	`         (BinaryTree[CurrentNode].LeftPointer == -1)):`
	`    print BinaryTree[CurrentNode].DataValue`
	`  return`
		8

Question	Answer	Marks

6(b)	Visual Basic example code:
	
	`Procedure FindLeaves(CurrentNode):`
	
	`   if(BinaryTree[CurrentNode].LeftPointer <> -1) then`
	`    FindLeaves(BinaryTree[CurrentNode].LeftPointer)`
	`  End if`
	`  if(BinaryTree[CurrentNode].RightPointer <> -1) then`
	`    FindLeaves(BinaryTree[CurrentNode].RightPointer)`
	`  end if`
	`  if ((BinaryTree[CurrentNode].RightPointer = -1) and`
	`       (BinaryTree[CurrentNode].LeftPointer = -1)) then`
	`    Console.WriteLine(BinaryTree[CurrentNode].DataValue)`
	`  End if`
	`End Procedure`
	
	Pascal example code:
	
	`Procedure FindLeaves(CurrentNode);`
	`Begin`
	`  if(BinaryTree[CurrentNode].LeftPointer <> -1) then`
	`    FindLeaves(BinaryTree[CurrentNode].LeftPointer);`
	
	`  if(BinaryTree[CurrentNode].RightPointer <> -1):`
	`    FindLeaves(BinaryTree[CurrentNode].RightPointer);`
	
	`  if((BinaryTree[CurrentNode].RightPointer = -1) and`
	`      (BinaryTree[CurrentNode].LeftPointer = -1)) then`
	`    print (BinaryTree[CurrentNode].DataValue);`
	`End;`	