QUESTION 1.

Question	Answer	Marks
1(a)(i)	1 mark for each correct statement: • bird(lays_egg). • bird(has_wings).	**2**
1(a)(ii)	1 mark for each correct line: • feature(eagle, lays_eggs). • feature(eagle, has_wings).	**2**
1(b)(i)	1 mark for each animal: tuna, crab	**2**
1(b)(ii)	1 mark per bullet point: • feature() • tuna, C feature(tuna, C)	**2**

1(c)	1 mark per bullet point to max 3:
	
	• feature(X,Y) AND bird(Y) // feature(X, has_wings)
	• AND
	• feature(X,Z) AND bird(Z) // feature(X, lays_eggs)
	
	(feature(X, Y) AND bird(Y)) AND (feature(X, Z) AND bird(Z))
		3

1(d)(i)	A programming style/classification // characteristics/features that programming language has/uses	**1**
1(d)(ii)	1 mark for each: • Low-level • Imperative // Procedural	**2**

QUESTION 2.

2(b)	1 mark for each completed section
	
	`FUNCTION FindElement(Item : INTEGER) RETURNS INTEGER`
	`  CurrentPointer ← RootPointer`
	`  WHILE CurrentPointer <> NullPointer`
	`  IF List[CurrentPointer].Data <> Item`
	`  THEN`
	`  CurrentPointer ← List[CurrentPointer].Pointer`
	`  ELSE`
	`  RETURN CurrentPointer`
	`  ENDIF`
	`  ENDWHILE`
	`  CurrentPointer ← NullPointer`
	`  RETURN CurrentPointer`
	`ENDFUNCTION`
		6

2(c)(i)	1 mark per bullet point to max 3 e.g. • A sequence of steps that change the state of the program • The steps are in the order they should be carried out • e.g. procedural programming/language • Groups code into self-contained blocks // split the program into modules • … which are subroutines // by example	**3**

Question	Answer	
2(c)(ii)	1 mark per bullet point to max 3 e.g. • Creates classes • …as a blueprint for an object // objects are instances of classes • …that have properties/attributes **and** methods • … that can be private to the class // properties can **only** be accessed by the class's methods // encapsulation • **Subclasses** can inherit from **superclasses** (child and parent) • A **subclass** can inherit the methods and properties from the **superclass** • A **subclass** can change the methods from the **superclass** // **subclass** can use polymorphism • Objects can interact with each other	

2(d)(i)	1 mark per bullet point
	
	• Method header and close (where appropriate)
	• …with `InputPlayerID` parameter
	• Initialise `Score` to 0
	• Initialise Category to `"Not Qualified"`
	• Initialise `PlayerID` to parameter
	
	PYTHON
	`def__init__(self, InputPlayerID):`
	`  self.__Score = 0`
	`  self.__Category = "Not Qualified"`
	`  self.__PlayerID = InputPlayerID`
	PASCAL
	`Constructor Player.Create(InputPlayerID);`
	`begin`
	`  Score := 0 ;`
	`  Category := 'Not Qualified' ;`
	`  PlayerID := InputPlayerID;`
	`end;`
	VB
	`Public Sub New (InputPlayerID)`
	`  Score = 0`
	`  Category = "Not Qualified"`
	`  PlayerID = InputPlayerID`
	`End Sub`
		5

Question	Answer

2(d)(ii)	1 mark per bullet point
	
	• 1 get Method header without parameter (returning correct data type if given)
	• …returning the property
	• A second working Get
	• A third working Get
	
	PYTHON
	`def GetScore():`
	`  return (Score)`
	`def GetCategory():`
	`  return (Category)`
	`def GetPlayerID():`
	`  return (PlayerID)`
	
	PASCAL
	`function GetScore():Integer;`
	`begin`
	`  GetScore:= Score;`
	`end;`
	`function GetCategory():String;`
	`begin`
	`  GetCategory:= Category;`
	`end;`
	`function GetPlayerID():String;`
	`begin`
	`  GetPlayerID:= PlayerID;`
	`end;`
	
	VB
	`Public Function GetScore() As Integer`
	`  Return Score`
	`End Function`
	`Public Function GetCategory() As String`
	`  Return Category`
	`End Function`
	`Public Function GetPlayerID() As String`
	`  Return PlayerID`
	`End Function`

Question	Answer	

2(d)(iii)	1 mark per bullet point
	
	• Set method header and close (where appropriate)
	• Input value
	• Looping until input value is correct length …
	• … storing **valid** input value in `PlayerID`
	
	PYTHON
	`def SetPlayerID(self)`
	`  PlayerID = input("Enter your player ID")`
	`  while len(PlayerID) > 15 and len(PlayerId) < 4`
	`  PlayerID = input("Must be <=15 AND >=4 characters long. Enter your player ID")`
	
	PASCAL
	`Procedure SetPlayerID ()`
	`  WriteLn ('Enter your player ID');`
	`  ReadLn(PlayerID);`
	`  while length(PlayerID) > 15 and length(PlayerID) < 4 do`
	`  begin`
	`  WriteLn('Must be <=15 AND >=4 characters long. Enter your player ID');`
	`  ReadLn(PlayerID);`
	`  end;`
	
	VB
	`Public Sub SetPlayerID()`
	`  Console.WriteLine ("Enter your player ID")`
	`  PlayerID = Console.ReadLine()`
	`  While Len(PlayerID) > 15 and Len(PlayerID) < 4`
	`  Console.WriteLine ("Must be <=15 AND >=4 characters long. Enter your player ID")`
	`  PlayerID = Console.ReadLine()`
	`  End While`
	`End Sub`	

Question	Answer

2(d)(iv)	1 mark per bullet point
	
	• Function header and close (where appropriate) **and** takes `ScoreInput` as parameter
	• Check if 0 <= `ScoreInput` <= 150
	• …if valid, set `Score` to parameter
	• …if not valid, output error
	• Returns TRUE if valid **and** returns FALSE if not valid
	
	PYTHON
	`def __SetScore(ScoreInput):`
	`  if ScoreInput >=0 and ScoreInput <=150:`
	`  IsValid = True`
	`  self__Score = ScoreInput`
	`  else:`
	`  print("Error")`
	`  IsValid = False`
	`Return(IsValid)`
	
	PASCAL
	`function Player.SetScore(ScoreInput: Integer) : Boolean;`
	`begin`
	`  If (ScoreInput >=0) AND (ScoreInput <=150) Then`
	`  IsValid := True;`
	`  result := ScoreInput;`
	`  Else`
	`  WriteLn('Error')`
	`  result := False;`
	`end;`
	
	VB
	`Public Function SetScore(ByVal ScoreInput As Integer) As Boolean`
	`  If (ScoreInput >=0) And (ScoreInput <=150) Then`
	`  Return True`
	`  Score = ScoreInput`
	`  Else`
	`  Console.Writeline("Error")`
	`  Return False`
	`  End If`
	`End Function`

Question	Answer

2(d)(v)	1 mark per bullet point
	
	• Procedure header and close (where appropriate)
	• Accessing Score attribute
	• Correct selection to assign each category
	• … storing in Category attribute
	
	PYTHON
	`def SetCategory()`
	`  if self.__Score >120:`
	`  self.__Category = "Advanced"`
	`  elif self.__Score >80:`
	`  self.__Category = "Intermediate"`
	`  elif self.__Score>=50:`
	`  self.__Category = "Beginner"`
	`  else:`
	`  self.__Category = "Not Qualified"`
	
	PASCAL
	`procedure player.SetCategory()`
	`begin`
	`  If Score >120 Then`
	`  Category := "Advanced";`
	`  Else If Score >80 Then`
	`  Category := "Intermediate";`
	`  Else If Score >= 50 Then`
	`  Category := "Beginner";`
	`  Else`
	`  Category := "Not Qualified";`
	`end;`
	
	VB
	`Public Sub SetCategory()`
	`  If Score >120 Then`
	`  Category = "Advanced"`
	`  ElseIf Score >80 Then`
	`  Category = "Intermediate"`
	`  ElseIf Score >=50 Then`
	`  Category = "Beginner"`
	`  Else`
	`  Category = "Not Qualified"`
	`  End If`
	`End Sub`

Question	Answer

2(d)(vi)	1 mark per bullet point
	
	• `CreatePlayer()` header and close (where appropriate)
	• Input of `score` and `PlayerID` with suitable prompts
	• Create instance of `Player` named `JoannePlayer` …
	• …with `PlayerID` as parameter
	• Call method `SetScore` for `JoannePlayer` with parameter `Score`
	• …storing return value
	• …outputting appropriate message for not valid
	• Call `SetCategory` for `JoannePlayer`
	• Output Category for `JoannePlayer` …
	• … using `GetCategory` for object Joanne
	
	PYTHON
	`def CreatePlayer():`
	`  InputPlayerID = input("Enter your chosen ID")`
	`  Score = int(input("Please enter the score"))`
	`  JoannePlayer = Player(InputPlayerID)`
	`  if JoannePlayer.SetScore(Score) == false:`
	`  print("Invalid score")`
	`  else:`
	`  JoannePlayer.SetCategory()`
	`  print(JoannePlayer.GetCategory)`
	
	PASCAL
	`procedure CreatePlayer();`
	`  var`
	`  playerID : String;`
	`  isValid : boolean;`
	`  JoannePlayer : Player;`
	`  score : integer;`
	`  begin`
	`  Writeln(Enter Player ID: ');`
	`  Readln(playerID);`
	`  Writeln('Enter score: ');`
	`  Readln(score);`
	`  JoannePlayer := Player.Create(PlayerID);`
	`  isValid := JoannePlayer.SetScore(Score);`
	`  if isValid = true:`
	`  JoannePlayer.SetCategory();`
	`  Writeln(JoannePlayer.GetCategory());`
	`  else:`
	`  Writeln("Invalid score")`
	`end;`

<table>
<tr><th>Question</th><th>Answer</th><th></th></tr>
<tr><td>2(d)(vi)</td><td>VB
<pre>
Sub CreatePlayer()
    Dim Score As Integer, InputPlayerID As String
    Console.WriteLine("Please enter your chosen ID")
    InputPlayerID = Console.ReadLine()
    Console.WriteLine("Please enter the score")
    Score = Console.ReadLine()

    Dim JoannePlayer As New Player(InputPlayerID)
    if JoannePlayer.SetScore(Score) = True then
        JoannePlayer.SetCategory()
        Console.WriteLine(JoannePlayer.GetCategory())
    else
        Console.Writeline("Invalid score")
    endif
End Sub
</pre></td><td></td></tr>
<tr><td>2(e)</td><td>1 mark per bullet point

• 3 correct Normal test data
• 3 correct Abnormal test data
• 3 correct Boundary test data

<table>
<tr><th>Category</th><th>Type of test data</th><th>Example test data</th></tr>
<tr><td rowspan="3">Beginner</td><td>Normal</td><td>e.g. 75</td></tr>
<tr><td>Abnormal</td><td>e.g. 85 / bob</td></tr>
<tr><td>Boundary</td><td>80, 50</td></tr>
<tr><td rowspan="3">Intermediate</td><td>Normal</td><td>e.g. 95</td></tr>
<tr><td>Abnormal</td><td>e.g. 70 / bob</td></tr>
<tr><td>Boundary</td><td>81, 120</td></tr>
<tr><td rowspan="3">Advanced</td><td>Normal</td><td>e.g. 125</td></tr>
<tr><td>Abnormal</td><td>e.g. 115 / bob</td></tr>
<tr><td>Boundary</td><td>121, 150</td></tr>
</table></td><td>3</td></tr>
<tr><td>2(f)(i)</td><td>Insertion sort</td><td>1</td></tr>
<tr><td>2(f)(ii)</td><td>One from:
• Bubble sort
• Merge sort</td><td>1</td></tr>
</table>

Question	Answer
2(f)(iii)	1 mark per shaded section

Item	NumberOfScores	InsertScore	Index	ArrayData				
				0	1	2	3	4
				99	125	121	109	115
1	5	125	0		(125)			
2		121	1			125		
			0		121			
3		109	2				125	
			1			121		
			0		109			
4		115	3					125
			2				121	
			1			115		