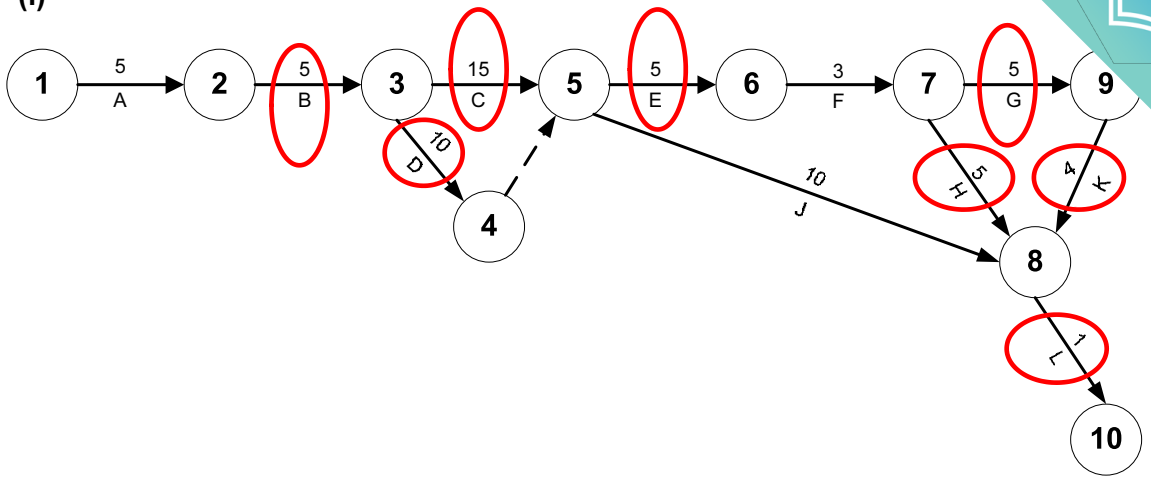
QUESTION 1.

1 (a) (i)



[max. 7]

(ii) 1 – 2 – 3 – 5 – 6 – 7 – 9 – 8 – 10
1–5 scores 1
6–10 scores 1 [2]

(iii) 43 weeks [1]

(b) (i) week number 25 [1]

(ii) week number 32 [1]

(c) To see what activities can be done in parallel // show dependencies
To record changes to project timings [max. 1]

QUESTION 2.

4 (a)

```
FUNCTION Hash(Key : STRING) RETURNS INTEGER
DECLARE Number : INTEGER
Number ← ASCII(LEFTSTRING(Key,1))
// Number ← ASCII(Key[1])
Number ← Number - 64
RETURN Number
// Result ← Number  // Hash ← Number
ENDFUNCTION
```

Accept `ASC` instead of `ASCII`
Accept `LEFT` instead of `LEFTSTRING`
Key can be a different identifier but must be the same in both places [5]

(b) (i)

Index	Dictionary Key	Value
1		
2		
3	Computer	Rechner
4	Disk	Platte
5	Error	Fehler
6	File	Datei
7		
8		
:	:	
:	:	
1999		
2000		

Ignore spelling mistakes
1 mark for 2 correct pairs entered in correct slots [2]

(ii) Collision / synonym / space already occupied / same index in array
Overwrites previous key-value pair

reject error [Max 2]

(iii) Create an overflow area
The `'home'` record has a pointer to others with the same key // linked list
OR
Store the overflow record at the next available address …
in sequence (= next available)
OR
Re-design the hash function …. // write a different/another algorithm
to generate a wider range of indexes // enlarging storage space // to create fewer collisions [2]

(iv) Mark as follows:

Check whether slot is empty:

```
IF Dictionary[Index,1] <>"" //  != '' //  > NULL  // 
NONE
```

If not: update index: `THEN Index ← <some value>`

...to find an empty slot (loop / follow pointer / go to overflow area) reject FOR loop

Insert code between lines 20 and 30

```
21 WHILE Dictionary[Index,1] > ""
22   Index ← Index + 1
23   IF Index > 2000
24     THEN
25       Index ← 1
26   ENDIF
```

QUESTION 3.

4

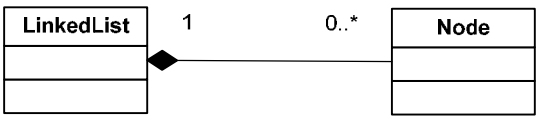
	Acceptance testing	Integration testing	
Who	The end user // user of the software	The programmer / in-house testers	**[1] + [1]**
When	When the software is finished/ when it is installed	When the separate modules have been written and tested	**[1] + [1]**
Purpose	To ensure the software is what the customer ordered // to check that the software meets the user requirements	To ensure the modules work together as expected	**[1] + [1]**

QUESTION 4.

2 [6]

	(i) Alpha testing	**(ii) Beta testing**
Who	In house testers / developers / programmers	(potential) (end) user(s)/client(s)
When	Near the end of development // program is nearly fully-usable // after integration and before beta	Before general release of software // passed Alpha testing
Purpose	To find errors not found in earlier testing // ensure ready for beta testing	For constructive comments/ feedback // to test in real-life scenarios/situations/ environments // ensure it is ready for release // ensure it meets users' needs

QUESTION 5.

Question	Answer	Marks
4(a)(i)	containment/aggregation	**1**
4(a)(ii)	 1 mark for the two classes (in boxes) and connection with correct end point 1 mark for 0 ..* 0	**Max 2**

Question	Answer	

4(b)	**mark as follows:**
	• Class heading and ending
	• Constructor heading and ending
	• Parameters in constructor heading
	• Declaration of (private) attributes : Pointer, Data
	• Assignment of parameters to Pointer and Data
	
	Python Example
	`class Node:` 1
	`def __init__(self, D, P):` 1 + 1
	`self.__Data = D` 1
	`self.__Pointer = P` 1
	`return`
	
	Example Pascal
	`type` } `Node = class` 1
	`private` 1
	`Data : String;` } `Pointer : Integer;`
	`public`
	`constructor Create(D : string; P : integer);`
	`procedure SetPointer(P : Integer);`
	`procedure SetData(D : String);`
	`function GetData() : String;` } ignore
	`function GetPointer() : Integer;`
	`end;`
	`constructor Node.Create(D : string; P : integer);` 1+1
	`begin`
	`Data := D;` } `Pointer := P;` 1
	`end;`
	
	Example VB.NET
	`Class Node` 1
	`Private Data As String` } `Private Pointer As Integer` 1
	`Public Sub New(ByVal D As String, ByVal P As Integer)` 1+1
	`Data = D` } `Pointer = P` 1
	`End Sub`
	`End Class`	

4(c)(i)	A pointer that doesn't point to any data/node/address	**1**

Question	Answer	
4(c)(ii)	–1 (accept NULL) The array only goes from 0 to 7 // the value is not an array index	
4(c)(iii)	**mark as follows:** • Class and constructor heading and ending • Declare private attributes (HeadPointer, FreeListPointer, NodeArray) • Initialise HeadPointer to null • Initialise FreeListPointer to 0 • Looping 8 times … • Creating empty node in NodeArray • Use .SetPointer method to point each new node to next node • Set last node pointer to null pointer	**Max 7**

Python Example

```
class LinkedList:                                          1
   def __init__(self):                                    1
      self.__HeadPointer = - 1                             1
      self.__FreeListPointer = 0                           1
      self.__NodeArray = []
      for i in range(8):                                   1
         ThisNode = Node("", (i + 1))                      1
         self.__NodeArray.append(ThisNode)
      self.__NodeArray[7].SetPointer(- 1)                  1
```

Example Pascal

```
type
LinkedList = class                                         1
      private
         HeadPointer : Integer;
         FreeList : Integer;
         NodeArray : Array[0..7] of Node;
      public
         constructor Create();
         procedure FindInsertionPoint(NewData : string; var
                     PreviousPointer, NextPointer : integer);
         procedure AddToList(NewData : string);
         procedure OutputListToConsole();
   end;
constructor LinkedList.Create();                           1
   var i : integer;
   begin
      HeadPointer := -1;                                   1
      FreeList := 0;                                       1
      for i := 0 To 7 do                                   1
         NodeArray[i] := Node.Create('', (i + 1));         1
      NodeArray[7].SetPointer(-1);                         1
   end;
```

Question	Answer	

	Example VB.NET
	`Class LinkedList` 1
	`  Private HeadPointer As Integer`
	`  Private FreeList As Integer`
	`  Private NodeArray(7) As Node`
	
	`  Public Sub New()` 1
	`  HeadPointer = -1` 1
	`  FreeList = 0` 1
	`  For i = 0 To 7` 1
	`  NodeArray(i) = New Node("", (i + 1))` 1
	`  Next`
	`  NodeArray(7).SetPointer(-1)` 1
	`  End Sub`
	`End Class`	

4(c)(iv)	• Creating instance of LinkedList assigned to contacts
	
	Python Example
	`contacts = LinkedList()`
	
	Pascal Example
	`var contacts : LinkedList;`
	`  contacts := LinkedList.Create;`
	
	VB.NET Example
	`Dim contacts As New LinkedList`
		1

Question	Answer	

4(c)(v)	**mark as follows:**
	• Start with HeadPointer
	• Output node data
	• Loop until null pointer
	• Following pointer to next node
	• Use of getter (ie GetData/GetPointer)
	
	Python Example
	`def OutputListToConsole(self) :`
	`  Pointer = self.__HeadPointer` 1
	`  while Pointer != -1 :` 1
	`  print(self.__NodeArray[Pointer].GetData())` 1+1
	`  Pointer = self.__NodeArray[Pointer].GetPointer()` 1
	`  print()`
	`  return`
	
	Pascal Example
	`procedure LinkedList.OutputListToConsole();`
	`  var Pointer : integer;`
	`  begin`
	`  Pointer := HeadPointer;` 1
	`  while Pointer <> -1 do` 1
	`  begin`
	`  WriteLn(NodeArray[Pointer].GetData);` 1+1
	`  Pointer := NodeArray[Pointer].GetPointer;` 1
	`  end;`
	`  end;`
	
	VB.NET Example
	
	`Public Sub OutputListToConsole()`
	`  Dim Pointer As Integer`
	`  Pointer = HeadPointer` 1
	`  Do While Pointer <> -1` 1
	`  Console.WriteLine(NodeArray(Pointer).GetData)` 1+1
	`  Pointer = NodeArray(Pointer).GetPointer` 1
	`  Loop`
	`End Sub`	

Question	Answer	

4(c)(vi)	**mark as follows:**
	• Store free list pointer as NewNodePointer
	• Store new data item in free node
	• Adjust free pointer
	• F list is currently empty
	• Make the node the first node
	• Set pointer of this node to Null Pointer
	• Find insertion point
	• If previous pointer is Null pointer
	• Link this node to front of list
	• Link new node between Previous node and next node
	
	Python Example
	`def AddToList(self, NewData):`
	
	`  NewNodePointer = self.__FreeListPointer`
	
	`  self.__NodeArray[NewNodePointer].SetData(NewData)`
	
	`  self.__FreeListPointer = self.__NodeArray[self.__FreeListPointer].GetPointer()`
	
	`  if self.__HeadPointer == -1:`
	
	`  self.__HeadPointer = NewNodePointer`
	`  self.__NodeArray[NewNodePointer ].SetPointer(-1)`
	`  else:`
	`  PreviousPointer, NextPointer = self.FindInsertionPoint(NewData)`
	`  if PreviousPointer == -1 :`
	
	`  self.__NodeArray[NewNodePointer ].SetPointer (self.__HeadPointer)`
	`  self.__HeadPointer = NewNodePointer`
	`  else:`
	
	`  self.__NodeArray[NewNodePointer ].SetPointer(NextPointer)`
	`  self.__NodeArray[PreviousPointer].SetPointer(NewNodePointer)`	

Question	Answer

	Pascal Example
	`procedure LinkedList.AddToList(NewData : string);`
	`var NewNodePointer , PreviousPointer,`
	`NextPointer : integer;`
	`begin`
	`// make a copy of free list pointer`
	`NewNodePointer := FreeListPointer;`
	`// store new data item in free node`
	`NodeArray[NewNodePointer].SetData(NewData);`
	`// adjust free pointer`
	`FreeListPointer :=`
	`NodeArray[FreeListPointer].GetPointer;`
	`// if list is currently empty`
	`if HeadPointer = -1`
	`then`
	`// make the node the first node`
	`begin`
	`HeadPointer := NewNodePointer;`
	`// set pointer to Null pointer`
	`NodeArray[NewNodePointer].SetPointer(-1);`
	`end`
	`else`
	`// find insertion point`
	`begin`
	`FindInsertionPoint(NewData, PreviousPointer,`
	`NextPointer);`
	`// if previous pointer is Null pointer`
	`if PreviousPointer = -1`
	`then`
	`// link node to front of list`
	`begin`
	`NodeArray[NewNodePointer]`
	`.SetPointer(HeadPointer);`
	`HeadPointer := NewNodePointer ;`
	`end`
	`else`
	`// link new node between`
	`Previous node and next node`
	`begin`
	`NodeArray[NewNodePointer ]`
	`.SetPointer(NextPointer);`
	`NodeArray[PreviousPointer]`
	`.SetPointer(NewNodePointer);`
	`end;`
	`end;`
	`end;`

Question	Answer

	VB.NET Example
	`Public Sub AddToList(ByVal NewData As String)`
	`  Dim NewNodePointer, PreviousPointer, NextPointer As Integer`
	`  ' make copy of free list pointer`
	`  NewNodePointer= FreeListPointer`
	`  ' store new data item in free node`
	`  NodeArray(NewNodePointer).SetData(NewData)`
	`  ' adjust free pointer`
	`  FreeListPointer = NodeArray(FreeListPointer).GetPointer`
	`  ' if list iscurrently empty`
	`  If HeadPointer = -1 Then`
	`  ' make the node the first node`
	`  HeadPointer = NewNodePointer`
	`  ' set pointer to Null pointer`
	`  NodeArray(NewNodePointer).SetPointer(-1)`
	`  Else`
	`  ' find insertion point`
	`  FindInsertionPoint(NewData, PreviousPointer,`
	`NextPointer)`
	`  ' if previous pointer is Null pointer`
	`  If PreviousPointer = -1 Then`
	`  ' link to front of list`
	`  NodeArray(NewNodePointer).SetPointer(HeadPointer)`
	`  HeadPointer = NewNodePointer`
	`  Else`
	`  ' link new node between Previous node and next node`
	`  NodeArray(NewNodePointer).SetPointer(NextPointer)`
	`  NodeArray(PreviousPointer).SetPointer(NewNodePointer)`
	`  End If`
	`  End If`
	`End Sub`

Question	Answer

Pseudocode for reference:

```
PROCEDURE AddToList(NewData)
   // remember value of free list pointer
   NewNodePointer ← FreeListPointer
   // add new data item to free node pointed to by free list
   NodeArray[NewNodePointer].Data ← NewData
   // adjust free pointer to point to next free node
   FreeListPointer  ← NodeArray[FreeList].Pointer
   // is list currently empty?
   IF HeadPointer = NullPointer
      THEN
         // make the node the first node
         HeadPointer ← NewnodePointer
         // set pointer of new node to Null pointer
         NodeArray[NewNodePointer].Pointer ← NullPointer
      ELSE
         // find insertion point
         CALL FindInsertionPoint(NewData, PreviousPPointer, NextPointer)
         // if previous pointer is Null pointer
         IF PreviousPointer = NullPointer
            THEN
               // link new node to front of list
               NodeArray[NewNodePointer].Pointer ← HeadPointer
               HeadPointer ← NewNodePointer
         ELSE
            // link new node between previous node and next node
            NodeArray[NewNodePointer].Pointer ← NextPOinter
            NodeArray[PreviousPointer].Pointer ← NewNodePointer
         END IF
   ENDIF
END PROCEDURE
```

QUESTION 6.

Question	Answer	Marks
2(a)	• A pointer that doesn't point to another node/other data/address // indicates the end of the branch	1
2(b)	one mark per bullet • node with 'Athens' linked to left pointer of Berlin (ignore null pointer) • null pointers in left and right pointers of Athens	2
2(c)(i)	RootPointer: 0 FreePointer: 7 **1 mark** [0] 2 | Dublin | 1 [1] -1/∅ | London | 3 [2] 6 | Berlin | 5 [3] 4 | Paris | -1/∅ [4] -1/∅ | Madrid | -1/∅ [5] -1/∅ | Copenhagen | -1/∅ [6] -1/∅ | Athens | -1/∅ [7] 8 | | -1/∅ [8] 9 | | -1/∅ [9] -1/∅ | | -1/∅	5
2(c)(ii)	• –1 • It is not the number for any node.	2

Question	Answer	Marks
2(d)(i)	TYPE Node **LeftPointer : INTEGER** **RightPointer : INTEGER** 1 **Data : STRING** ENDTYPE DECLARE Tree : ARRAY[0 : 9] **OF Node** 1 DECLARE FreePointer : INTEGER DECLARE RootPointer : INTEGER PROCEDURE CreateTree() DECLARE Index : INTEGER **RootPointer ← -1** 1 **FreePointer ← 0** 1 FOR Index ← 0 TO 9 // link nodes **Tree[Index].LeftPointer ← Index + 1** 1 **Tree[Index].RightPointer ← -1** 1 ENDFOR **Tree[9].LeftPointer ← -1** 1 ENDPROCEDURE	**7**

Question	Answer	Marks

2(d)(ii)	PROCEDURE AddToTree(ByVal NewDataItem : STRING)
	// if no free node report an error
	IF FreePointer **= -1** 1
	THEN
	ERROR("No free space left")
	ELSE // add new data item to first node in the free list
	NewNodePointer ← FreePointer
	Tree[NewNodePointer].Data ← NewDataItem 1
	// adjust free pointer
	FreePointer ← **Tree[FreePointer].LeftPointer** 1
	// clear left pointer
	Tree[NewNodePointer].LeftPointer ← **-1** 1
	// is tree currently empty ?
	IF **RootPointer = -1** 1
	THEN // make new node the root node
	RootPointer ← NewNodePointer 1
	ELSE // find position where new node is to be added
	Index ← RootPointer
	CALL FindInsertionPoint(NewDataItem, Index, Direction)
		8

Question	Answer	Marks
	IF Direction = "Left" THEN // add new node on left **Tree[Index].LeftPointer ← NewNodePointer** 1 ELSE // add new node on right **Tree[Index].RightPointer ← NewNodePointer** **1** ENDIF ENDIF ENDIF ENDPROCEDURE	
2(e)	**1 mark per bullet** • test for base case (null/-1) • recursive call for left pointer • output data • recursive call for right pointer • order, visit left, output, visit right IF Pointer <> NULL 1 THEN TraverseTree(Tree[Pointer].LeftPointer) 1 OUTPUT Tree[Pointer].Data 1 + 1 TraverseTree(Tree[Pointer].RightPointer) 1 ENDIF ENDPROCEDURE	**5**

QUESTION 7.

Question	Answer	Marks
5(a)(i)	PERT / GANTT	1
5(a)(ii)	1 mark per bullet to max 3 For example: • Calculate total minimum time required for project • Identify milestones • Task dependencies • Provides the critical path analysis • Identify which tasks need to be prioritised • Determine when to begin specific tasks/stages • Identify slack time • Identify when resources need allocating • Identify tasks that can be completed in parallel	3
5(b)(i)	Integration	1
5(b)(ii)	Beta / acceptance	1

Question	Answer	Marks
6(a)	1 mark per bullet to max 6 • Declaring a class with the name animal • Declaring variables for across, down and score (all Integers) • …as private/protected • Correct constructor header and ending • Randomly generating an across between 0–39 inc. in constructor • Randomly generating a down between 0–39 inc. in constructor • Initialising Score to zero in constructor • Correct get for `Across` • Correct set for `Across`	6

Question	Answer	Marks
6(a)	Example: VB Class Animal Private Across As Integer Private Down As Integer Private Score As Integer Function GetAcross() Return Across End Function Sub SetAcross(Value As Integer) Across = Value End Sub Sub New() Randomize() Across = randomnumber.Next(0, 40) Down = randomnumber.Next(0, 40) Score = 0 End Sub End Class	

Question	Answer	Marks
6(a)	or Class Animal Private Across As Integer Property _Across As Integer Get Return _Across End Get Set(Value As Integer) Across = Value End Set End Property Private Down As Integer Private _Score As Integer Sub New() Randomize() Across = randomnumber.Next(0, 40) Down = randomnumber.Next(0, 40) _Score = 0 End Sub End Class Example: Python class Animal : def __init__ (self) : x = random.randint(0,39) y = random.randint(0,39) self.Across = x self.Down = y self.Score = 0 def SetAcross(A) : self.Across = A def GetAcross() : return self.Across	

Question	Answer	Marks

6(a)	Example: Pascal
	type
	Animal = class
	private
	Across: integer;
	Down: integer;
	score: integer;
	public
	constructor init;
	
	procedure SetAcross(AcrossV: integer);
	function GetAcross(): integer;
	end;
	
	constructor Animal.init();
	SetAcross(random(40));
	SetDown (random(40));
	SetScore (0);
	end;
	
	procedure Animal.SetAcross(AcrossV: integer);
	begin
	Across := AcrossV;
	end;
	
	function Animal.GetAcross(): integer;
	begin
	GetAcross := Across;
	end;	

Question	Answer	Marks

6(b)	1 mark per bullet to max 5
	• constructor method heading and ending
	• Initialise all 40 by 40 elements of Grid as '' or equivalent
	• Loop 5 times…
	• …Creates a new instance of animal inside loop...
	• ...and adds it to array `AnimalList`
	
	• Call generate food **and** initialise `StepCounter` to 0
	
	Example Python
	
	`def __init__ (self) :`
	`  self.grid = [[' ' for i in range(40)] for j in range(40)]`
	`  self.AnimalList = []`
	`  self.StepCounter = 0`
	`  for i in range(5) :`
	`  newAnimal = Animal ()`
	`  self.AnimalList.append(newAnimal)`
	`  self.GenerateFood()`
	
	Example VB
	
	`Sub New()`
	`  For x = 0 To 39`
	`  For y = 0 To 39`
	`  grid(x, y) = ""`
	`  Next`
	`  Next`
	
	`  For z = 0 To 4`
	`  AnimalList(z) = New Animal`
	`  Next`
	
	`  Call GenerateFood()`
	`End Sub`
		5

Question	Answer	Marks

6(b)	Example Pascal
	
	`constructor Desert.init();`
	`  for x := 0 to 39 do`
	`  begin`
	`  for y := 0 to 39 do`
	`  begin`
	`  grid(x,y) = "";`
	`  end`
	`  end`
	
	`  for x := 0 to 4 do`
	`  begin`
	`  AnimalList(x) = object (Animal);`
	`  end`
	
	`  GenerateFood();`
	`end;`	

6(c)(i)	1 mark per bullet: • Function header and ending taking one value as parameter • Check if coordinate = 0 (on lower bound) • …generate random number (0 or 1) • Check if coordinate = 39 (on upper bound) • …generate random number (–1 or 0) • Generate random number (e.g. –1, 0, 1) • Return the generated value	**max 4**

Question	Answer	Marks

6(c)(i)	Example VB
	
	`Function GenerateDirection(ByRef coord As Integer)`
	`  Dim lowerbound As Integer = -1`
	`  Dim upperbound As Integer = 1`
	
	`  If coord = 0 Then`
	`  lowerbound = 0`
	`  ElseIf coord = 39 Then`
	`  upperbound = 0`
	`  End If`
	
	`  GenerateDirection = randomnumber.Next(lowerbound, upperbound)`
	
	`End Function`
	
	Example Python
	
	`def GenerateDirection(Coord) :`
	`  lowerBound = -1`
	`  upperBound = 1`
	`  if Coord == 0 :`
	`  lowerBound = 0`
	`  elif Coord == 39 :`
	`  upperBound = 0`
	`  return random.randint(lowerBound, upperBound)`	

Question	Answer	Marks

6(c)(i)	Example Pascal
	
	`function GenerateDirection(coord : Integer): Integer;`
	`begin`
	`  lowerbound = -1;`
	`  upperbound = 1;`
	`  if coord = 0 then`
	`  lowerbound = 0;`
	`  else if coord = 39 then`
	`  upperbound = 0;`
	`  GenerateDirection = random(39);`
	`end;`	

6(c)(ii)	1 mark per bullet to max 4
	• Procedure move header, no parameters
	• Calling GenerateDirection **twice** sending across and down as separate parameters
	• Add return value to Across
	• Add return value to Down
	• Check if the grid, at the (new) coordinates == “F”
	• ..if true, Call EatFood
	
	Example python
	
	`def Move(self) :`
	`  self.Across +=  GenerateChangeInCoordinate(self.Across)`
	`  self.Down +=  GenerateChangeInCoordinate(self.Down)`
	`  if grid[self.Across][self.Down] == 'F' :`
	`  self.EatFood()`
	`  return`
		4

Question	Answer	Marks

6(c)(ii)	Example VB
	
	`Sub Move(ByRef thisAnimal As Animal)`
	`  thisAnimal.across += GenerateChangeInCoordinate (thisAnimal.across)`
	`  thisAnimal.down += GenerateChangeInCoordinate (thisAnimal.down)`
	`  If thegrid._grid(thisAnimal.across, thisAnimal.down) = "F"`
	`  Then`
	`  Call EatFood()`
	`  End If`
	`End Sub`
	
	Example Pascal
	
	`procedure Move(thisAnimal : Animal);`
	`begin`
	`  thisAnimal.across = this.Animal.across + GenerateChangeInCoordinate (thisAnimal.across);`
	`  thisAnimal.down = thisAnimal.down + GenerateChangeInCoordinate (thisAnimal.down);`
	`  if (thisgrid.grid(thisAnimal.across, thisAnimal.down) = "F") then`
	`  EatFood();`
	`End;`	

6(d)	1 mark per bullet to max 3 • Pre-compiled • Collection of Code/modules/routines • Each module performs a specific purpose/task • Each module can be linked/imported into the program	**2**

QUESTION 8.

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

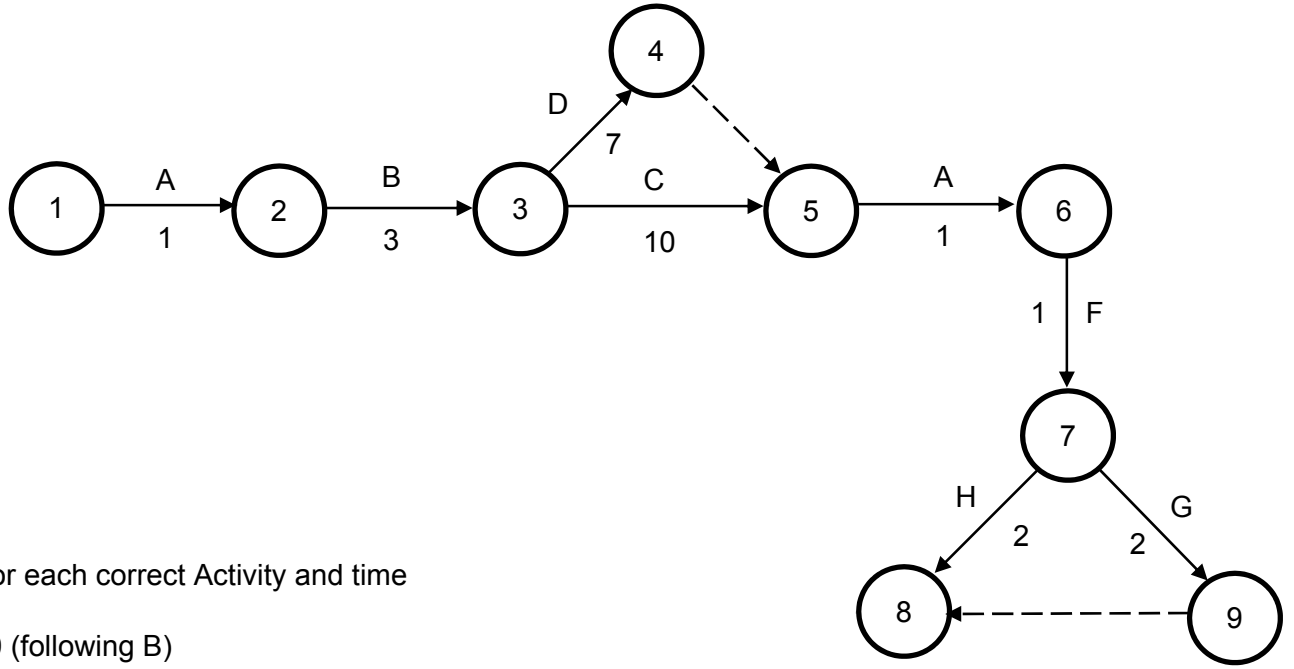
• 3 correct Normal test data
• 3 correct Abnormal test data
• 3 correct Boundary test data

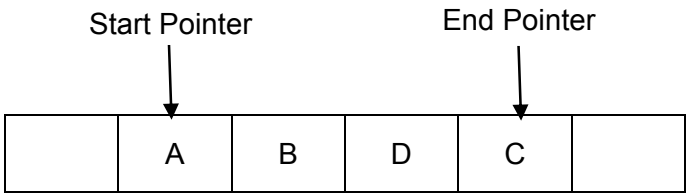
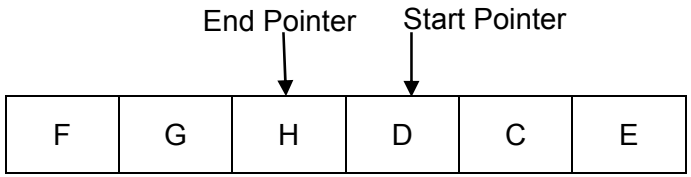
<table>
<tr><th>Category</th><th>Type of test data</th><th>Example test data</th></tr>
<tr><td rowspan="3">Beginner</td><td>Normal</td><td>e.g. 75</td></tr>
<tr><td>Abnormal</td><td>e.g. 85 / bob</td></tr>
<tr><td>Boundary</td><td>80, 50</td></tr>
<tr><td rowspan="3">Intermediate</td><td>Normal</td><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		

QUESTION 9.

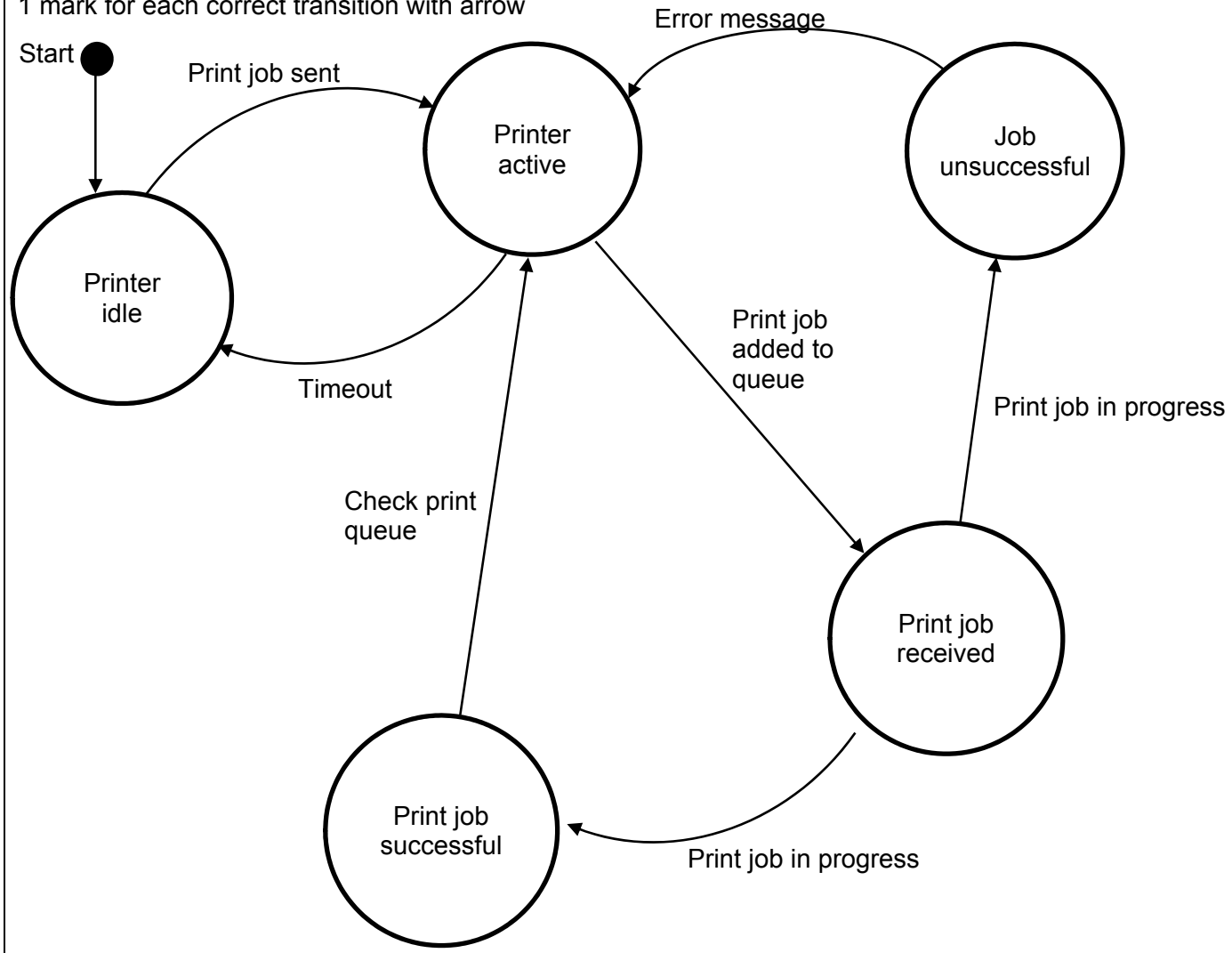
Question	Answer	Marks
1(a)(i)	 1 mark for each correct Activity and time • B 3 • C 10 (following B) • D 7 (following B) • E 3 (following C and D) • G 2 (following F) • H 2 (following F)	**6**
1(a)(ii)	The shortest time to complete the project // the sequence of activities that must be completed to avoid delaying the project	**1**
1(a)(iii)	1 mark for identify, max 1 for description • GANTT • A table that has time across the top and activities on the left, boxes are coloured to show dependencies and find critical path // colour in the boxes to show the length of time for each task	**2**

Question	Answer	Marks
1(b)(i)	A, B, C and D in correct places with no alteration to start and end pointer 	**1**
1(b)(ii)	1 mark per bullet point • correct jobs in correct order… • … correct location of start pointer • … correction location of new end pointer 	**3**
1(b)(iii)	1 mark from: • An error **message** would be generated	**1**

Question	Answer	Marks

1(b)(iv)	1 mark for each correct line
	
	`FUNCTION Remove RETURNS STRING`
	`  DECLARE PrintJob : STRING`
	`  IF StartPointer = EndPointer`
	`   THEN`
	`     RETURN "Empty"`
	`   ELSE`
	`     PrintJob ← Queue[StartPointer]`
	`     IF StartPointer = 5`
	`       THEN`
	`        StartPointer ← 0`
	`       ELSE`
	`        StartPointer ← StartPointer + 1`
	`      ENDIF`
	`     RETURN PrintJob`
	`   ENDIF`
	`ENDFUNCTION`
		4

1(b)(v)	1 mark per bullet point • A stack is Last In First Out (LIFO) while a queue is First In First Out (FIFO) • The queue removes and returns the element at **start pointer** // item is removed from the **start/head //** • A stack would remove and return the element at **end pointer** // item is removed from the **end**	**2**

Question	Answer	Marks
1(c)	1 mark for each correct transition with arrow 	5

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>1(d)(i)</td><td>1 mark per bullet
• Way of modelling logic
• Show all possible outputs // shows every possible outcome // all outcomes …
• … based on the inputs
• Determine which action to take in specific conditions // how different conditions affect the actions/outcomes</td><td>2</td></tr>
<tr><td>1(d)(ii)</td><td>1 mark for each row

Accept Y/X/ticks as long as clear which are Y. Accept N/X/– for empty spaces

<table>
<tr><th></th><th></th><th colspan="8">Rules</th></tr>
<tr><td rowspan="3">Conditions</td><td>Document printed, but quality is poor</td><td>Y</td><td>Y</td><td>Y</td><td>Y</td><td>N</td><td>N</td><td>N</td><td>N</td></tr>
<tr><td>Error light is flashing on printer</td><td>Y</td><td>Y</td><td>N</td><td>N</td><td>Y</td><td>Y</td><td>N</td><td>N</td></tr>
<tr><td>Document printed, but paper size is incorrect</td><td>Y</td><td>N</td><td>Y</td><td>N</td><td>Y</td><td>N</td><td>Y</td><td>N</td></tr>
<tr><td rowspan="4">Actions</td><td>Check connection from computer to printer</td><td></td><td></td><td></td><td></td><td></td><td>X</td><td></td><td></td></tr>
<tr><td>Check ink status</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Check if there is a paper jam</td><td></td><td></td><td></td><td></td><td></td><td>X</td><td></td><td></td></tr>
<tr><td>Check paper size selected</td><td>X</td><td></td><td>X</td><td></td><td>X</td><td></td><td>X</td><td></td></tr>
</table>
</td><td>4</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>1(d)(iii)</td><td>1 mark each for each correct column

Accept –/X for empty spaces. Accept Y/X/Ticks as long as clear which are used

<table>
<tr><th></th><th></th><th colspan="8">Rules</th></tr>
<tr><td rowspan="3">Conditions</td><td>Document printed, but quality is poor</td><td>Y</td><td>Y</td><td>N</td><td>N</td><td>N</td><td></td><td></td><td></td></tr>
<tr><td>Error light is flashing on printer</td><td></td><td></td><td></td><td>Y</td><td>N</td><td></td><td></td><td></td></tr>
<tr><td>Document printed, but paper size is incorrect</td><td>Y</td><td>N</td><td>Y</td><td>N</td><td>N</td><td></td><td></td><td></td></tr>
<tr><td rowspan="4">Actions</td><td>Check connection from computer to printer</td><td></td><td></td><td></td><td>X</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Check ink status</td><td>X</td><td>X</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Check if there is a paper jam</td><td></td><td></td><td></td><td>X</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Check paper size selected</td><td>X</td><td></td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>5</td></tr>
</table>

Question	Answer	Marks

1(e)(i)	1 mark per bullet point to **max 4**
	
	• Method header and close (where necessary) with three parameters
	• Initialised `PrintID`, `FirstName`, `LastName` and `Credits` …
	• … to the parameters
	• Initialised `Credits` to 50
	
	PYTHON
	`def__init__(self, NewFN, NewLN, NewPrintID):`
	`  self.__PrintID = NewPrintID`
	`  self.__FirstName = NewFN`
	`  self.__LastName = NewLN`
	`  self.__Credits = 50`
	
	PASCAL
	`Constructor NewPrintAccount.Create(NewFN, NewLN, NewPrintID);`
	`begin`
	`  PrintID := NewPrintID;`
	`  FirstName = NewFN;`
	`  LastName = NewLN;`
	`  Credits := 50;`
	`end;`
	
	VB
	`Public Sub New(NewFN, NewLN, NewPrintID As String)`
	`  PrintID = NewPrintID`
	`  FirstName = NewFN`
	`  LastName = NewLN`
	`  Credits = 50`
	`End Sub`
		4

Question	Answer	Marks

1(e)(ii)	1 mark per bullet point
	
	• method/procedure header (and close where appropriate) taking a parameter
	• `FirstName` is set to **parameter**
	
	PYTHON
	`def __SetFirstName(self, NewFirstName):`
	`  self.__FirstName = NewFirstName`
	
	PASCAL
	`procedure SetFirstName(newFirstName : String);`
	`begin`
	`  FirstName := newFirstName;`
	`end;`
	
	VB
	`public sub SetFirstName(NewFirstName As String)`
	`  FirstName = NewFirstName`
	`End Sub`
		2

Question	Answer	Marks

1(e)(iii)	1 mark per bullet point
	
	• concatenates `FirstName`, space and `LastName` …
	• … function/method header without parameter **and** returns (generated) value
	
	PYTHON
	`def __GetName(self):`
	`  return(self.__FirstName + " " + self.__LastName)`
	
	PASCAL
	`function GetName();`
	`begin`
	`  result := FirstName + " " + LastName`
	`end;`
	
	VB
	`public function GetName() As String`
	`  return(FirstName & " " & LastName)`
	`End Function`
		2

Question	Answer	Marks

1(e)(iv)	1 mark per each correct bullet point from:
	• **Procedure**/method header and close (where necessary) passing MoneyInput
	• At least **3** constants (e.g. `freecredit10`, `freecredit20`, `twenty`, `10`, `creditperdollar`)
	• If `MoneyInput<10` calculate `MoneyInput * 25`
	• If `MoneyInput >9` and `MoneyInput <20` then calculate `MoneyInput * 25 + 25`
	• If `MoneyInput >19` then calculate `MoneyInput * 25 + 50`
	• All **three** correct calculations add to `Credits`, not overwrite
	• Efficient IF (i.e. elseif)
	
	PYTHON
	`def __AddCredits(self, MoneyInput):`
	`  CreditPerDollar = 25`
	`  FreeCredit10 = 25`
	`  FreeCredit20 = 50`
	`  Twenty = 20`
	`  Ten = 10`
	`  if MoneyInput >= Twenty:`
	`  Credits = Credits + (MoneyInput * CreditPerDollar) + FreeCredit20`
	`  elif MoneyInput >= Ten:`
	`  Credits = Credits + (MoneyInput * CreditPerDollar) + FreeCredit10`
	`  else:`
	`  Credits = Credits + (MoneyInput * CreditPerDollar)`
		6

Question	Answer	Marks

1(e)(iv)	PASCAL
	`procedure AddCredits(MoneyInput : Real);`
	`  const CreditPerDollar = 25`
	`  const FreeCredit10 = 25`
	`  const FreeCredit20 = 50`
	`  const Twenty = 20`
	`  const Ten = 10`
	`begin`
	`  If MoneyInput > = Twenty Then`
	`  Credits := Credits + (MoneyInput * CreditPerDollar) + FreeCredit20;`
	`  Else If MoneyInput > = Ten Then`
	`  Credits := Credits + (MoneyInput * CreditPerDollar) + FreeCredit10;`
	`  Else`
	`  Credits := Credits + (MoneyInput * CreditPerDollar);`
	`end;`
	
	VB.NET
	`Public Sub AddCredits(MoneyInput As Integer)`
	`  Const CreditPerDollar As Integer = 25`
	`  Const FreeCredit10 As Integer = 25`
	`  Const FreeCredit20 AS integer = 50`
	`  Const Twenty As Integer = 20`
	`  Const Ten AS Integer = 10`
	`  If MoneyInput > = Twenty Then`
	`  Credits = Credits + (MoneyInput * CreditPerDollar) + FreeCredit20`
	`  Else If MoneyInput > = Ten Then`
	`  Credits = Credits + (MoneyInput * CreditPerDollar) + FreeCredit10`
	`  Else`
	`  Credits = Credits + (MoneyInput * CreditPerDollar)`
	`  End If`
	`End Sub`	

Question	Answer	Marks
1(e)(v)	1 mark per bullet • Declaring `StudentAccounts` as array of 1000 elements… • …of type `PrintAccount` `DECLARE StudentAccounts ARRAY[0:999] OF PrintAccount`	**2**

Question	Answer	Marks

1(e)(vi)	1 mark per bullet point to **max 8**
	
	• Generating ID with '1' at the end all in **lowercase** from parameters
	
	• Loop through array to last occupied element …
	• … Check if the `PrintID` already exists
	• … using `GetPrintID()`
	• … increment number at end of `PrintID`
	
	• Create a new instance of `PrintAccount` …
	• … sending `FirstName`, `LastName`, `PrintID` as parameters
	• adding new account to `StudentAccounts` at position `NumberStudents`
	• Increment `NumberStudents`
	
	VB.NET
	`Sub CreateId(firstName, lastName)`
	`  Dim count As Integer`
	`  Dim PrintID = Left(firstname, 3).ToLower & Left(lastname, 3).ToLower & "1"`
	`  Dim studentAdd As Integer = 0`
	`  If numberStudents <> 0 Then`
	`    For x = 0 To numberStudents - 1`
	`      If studentAccounts(x).getPrintID() = username Then`
	`        PrintID = PrintID + 1`
	`        username = Left(firstname, 3).ToLower & Left(lastname, 3).ToLower & PrintID.ToString`
	`       End If`
	`     Next`
	`     studentAdd = numberStudents`
	`   End If`
	`   studentAccounts(studentAdd) = New printAccount(firstname, lastname, username)`
	`   numberStudents = numberStudents + 1`
		8

Question	Answer	Marks

1(e)(vi)	**Python**
	`def CreateID(firstname, lastname):`
	`  count = 0`
	`  PrintID = firstname[0:3].lower() + lastname[-3].lower() + "1"`
	`  StudentAdd = 0`
	`  if numberStudents != 0:`
	`  for x in range(0, numberStudents):`
	`  if studentAccounts[x].getPrintID() == username:`
	`  PrintID = PrintID + 1`
	`  username = firstname[0:3].lower() + lastname[0:3].lower + str(PrintID)`
	`  studentAdd = numberStudents`
	`  studentAccounts[studentAdd] = printAccount(firstname, lastname, username)`
	`  numberStudents = numberStudents + 1`
	
	Pascal
	`procedure CreateID(firstname : String, lastname: String);`
	`  var`
	`  count : Integer;`
	`  studentAdd : Integer;`
	`  PrintID : String;`
	
	`begin`
	`  studentAdd := 0;`
	`  PrintID := LowerCase(substr(firstname,0,3)) + LowerCase(substr(lastname,0,3))+ "1";`
	`  if numberStudents <> 0:`
	`  ror x := 0 To numberStudents - 1;`
	`  if studentAccounts[x].getPrintID() = username:`
	`  PrintID := PrintID + 1;`
	`  username := LowerCase(substr(firstname, 3) +`
	`LowerCase(substr(lastname,0,3))+str(PrintID);`
	
	`  studentAdd := numberStudents;`
	`studentAccounts[studentAdd] := printAccount.Create(firstname, lastname, username);`
	`numberStudents := numberStudents + 1`	