

Cambridge International AS & A Level

COMPUTER SCIENCE**9618/42**

Paper 4 Practical

May/June 2025

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 2025 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

This document consists of **45** 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 descriptions 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.

PUBLISHED**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.

Annotations guidance for centres

Examiners use a system of annotations as a shorthand for communicating their marking decisions to one another. Examiners are trained during the standardisation process on how and when to use annotations. The purpose of annotations is to inform the standardisation and monitoring processes and guide the supervising examiners when they are checking the work of examiners within their team. The meaning of annotations and how they are used is specific to each component and is understood by all examiners who mark the component.

We publish annotations in our mark schemes to help centres understand the annotations they may see on copies of scripts. Note that there may not be a direct correlation between the number of annotations on a script and the mark awarded. Similarly, the use of an annotation may not be an indication of the quality of the response.

The annotations listed below were available to examiners marking this component in this series.

Annotations

Annotation	Meaning
	Benefit of the doubt
	To indicate where a key word/phrase/code is missing
	Incorrect
	Follow through
	Indicate a point in an answer
Highlighted text	To draw attention to a particular aspect or to indicate where parts of an answer have been combined
	Ignore
	Not answered question
	No examples or not enough

PUBLISHED

Annotation	Meaning
	Not relevant or used to separate parts of an answer
Off-page comment	Allows comments to be entered at the bottom of the RM marking window and then displayed when the associated question item is navigated to.
	Repetition
	Indicates that work or a page has been seen including blank answer spaces and blank pages.
	Correct
	Too vague

PUBLISHED

- **Bold** in mark scheme means that idea is required.
- / in mark scheme means alternative.
- // in mark scheme means alternative solution that gains the same mark point.
- ... at the end of one mark point without a ... at the start of the next just means the sentence follows on. There is no dependency.
- ... at the end of one mark point and at the start of the next, this means the second cannot be awarded without the first.
- () means what is in the brackets is not required, or it is not required in some languages but may be required in others.

PUBLISHED

Question	Answer	Marks
1(a)	1 mark each (Global) Stack as 1D array (of strings) with 20 elements initialised to string "-1" (Global) TopOfStack initialised to -1	2
Example program code: Java <pre>public static String[] Stack = new String[20]; public static Integer TopOfStack; public static void main(String args[]){ for(Integer X = 0; X < 20; X++){ Stack[X] = "-1"; } TopOfStack = -1; }</pre> VB.NET <pre>Dim Stack(19) As String Dim TopOfStack As Integer For x = 0 To 19 Stack(x) = "-1" Next TopOfStack = -1</pre> Python <pre>Stack = [] TopOfStack = -1 #main for x in range(20): Stack.append("-1")</pre>		

PUBLISHED

Question	Answer	Marks
1(b)	1 mark each • Push function header (and end where appropriate) taking one (string) parameter • Checking if stack is full and returning integer –1 • (Otherwise) Incrementing TopOfStack • Storing parameter in the incremented the stack at TopOfStack and returning integer 1	4
Example program code: <pre> Java public static Integer Push(String Data){ if (TopOfStack == 19){ return -1; }else{ TopOfStack++; Stack[TopOfStack] = Data; return 1; } } VB.NET Function Push(ByVal Data) If TopOfStack = 19 Then Return -1 Else TopOfStack = TopOfStack + 1 Stack(TopOfStack) = Data Return 1 End If End Function </pre>		

PUBLISHED

Question	Answer	Marks
	<pre>Python def Push(Data): global Stack global TopOfStack if TopOfStack == 19: return -1 else: TopOfStack += 1 Stack[TopOfStack] = Data return 1</pre>	

PUBLISHED

Question	Answer	Marks
1(c)	1 mark each • Function header (and end where appropriate) and returning a value in all cases. • Checking if stack is empty and returning string "-1" • (Otherwise) Decrementing TopOfStack • Returning element in stack at TopOfStack before TopOfStack is decremented	4
Example program code: <pre> Java public static String Pop(){ if (TopOfStack == -1){ return "-1"; }else{ String ReturnValue = Stack[TopOfStack]; TopOfStack--; return ReturnValue; } } VB.NET Function Pop() If TopOfStack = -1 Then Pop = "-1" Else Pop = Stack(TopOfStack) TopOfStack = TopOfStack - 1 End If End Function </pre>		

PUBLISHED

Question	Answer	Marks
	<pre>Python def Pop(): global Stack global TopOfStack if TopOfStack == -1: return "-1" else: ReturnValue = Stack[TopOfStack] TopOfStack -= 1 return ReturnValue</pre>	

Question	Answer	Marks
1(d)	1 mark each • Procedure header (and end where appropriate) taking one (string) parameter • Opening the file with the filename parameter and closing the file in an appropriate place • Looping through to end of file and reading in each line ... • ... calling Push() once with each read in value ... • ... if any return value from Push() is integer -1 outputting "Stack full" • Exception handling for opening and reading from file with appropriate catch and output	6
Example program code: <pre> Java public static void ReadData(String FileName){ Integer ReturnValue; try{ FileReader f = new FileReader(FileName); try{ BufferedReader Reader = new BufferedReader(f); String Line= Reader.readLine(); Line = Line.replace("\n",""); while (Line != null){ Line = Line.replace("\n",""); ReturnValue = Push(Line); if (ReturnValue == -1){ System.out.println("Stack full"); } Line = Reader.readLine(); } Reader.close(); }catch(IOException ex){} }catch(FileNotFoundException e){ System.out.println("File not found"); } } </pre>		

PUBLISHED

Question	Answer	Marks
	<pre> VB.NET Sub ReadData(ByVal FileName As String) Dim ReturnValue As String Try Dim FileReader As New System.IO.StreamReader(FileName) While Not FileReader.EndOfStream ReturnValue = Push(FileReader.ReadLine()) If ReturnValue = "-1" Then Console.WriteLine("Stack full") End If End While FileReader.Close() Catch ex As Exception Console.WriteLine("Cannot open file") End Try End Sub Python def ReadData(FileName): global Stack global TopOfStack try: File = open(FileName) for Line in File: ReturnValue = Push(Line.strip()) if ReturnValue == -1: print("Stack full") File.close() except: print("Cannot open file") </pre>	

PUBLISHED

Question	Answer	Marks
1(e)	1 mark for: • Calculate() function header (and end where appropriate) • Looping until the stack is empty • Calling Pop() repeatedly within loop and storing/using return value • ... working out if return value from Pop() call is an operator or a number / alternating between operator and number • Select to determine if the operator is +, -, /, * or ^ and attempt the matching calculation • ... performing correct calculation using operator, number • ... updating total from previous loops and returning this final value	7
Example program code: <pre> Java public static Double Calculate(){ Double Total = Double.parseDouble(Pop()); String ReturnValue = ""; String LastOperator = ""; Boolean OperatorFlag = true; Integer TheData = 0; while(ReturnValue != "-1"){ ReturnValue = Pop(); if(OperatorFlag == false){ TheData = Integer.parseInt(ReturnValue); if(LastOperator.compareTo("+")==0){ Total = Total + TheData; }else if(LastOperator.compareTo("-")==0){ Total = Total - TheData; }else if(LastOperator.compareTo("*")==0){ Total = Total * TheData; }else if(LastOperator.compareTo("/")==0){ Total = Total / TheData; }else if(LastOperator.compareTo("^")==0){ Total = Math.pow(Total, TheData); } OperatorFlag = true; } } } </pre>		

PUBLISHED

Question	Answer	Marks
	<pre> }else{ LastOperator = ReturnValue; OperatorFlag = false; } } return Total; } VB.NET Function Calculate() Dim Total As Integer = Pop() Dim ReturnValue As String = "" Dim LastOperator As String = "" Dim OperatorFlag As Boolean = True Dim TheData As Integer = 0 While (ReturnValue <> "-1") ReturnValue = Pop() Select Case OperatorFlag Case False TheData = ReturnValue Select Case LastOperator Case "+" Total = Total + TheData Case "-" Total = Total - TheData Case "*" Total = Total * TheData Case "/" Total = Total / TheData Case "^" Total = Total ^ TheData End Select OperatorFlag = True Case Else LastOperator = ReturnValue </pre>	

PUBLISHED

Question	Answer	Marks
	<pre> OperatorFlag = False End Select End While Return Total End Function Python def Calculate(): global Stack global TopOfStack Total = Pop() Total = int(Total) Return = 0 LastOperator = "" Operator = True while(Return != "-1"): Return = Pop() if Operator == False: Data = int(Return) if LastOperator == "+": Total = Total + Data elif LastOperator == "-": Total = Total - Data elif LastOperator == "*": Total = Total * Data elif LastOperator == "/": Total = Total / Data elif LastOperator == "^": Total = Total ** Data Operator = True else: LastOperator = Return Operator = False return Total </pre>	

PUBLISHED

Question	Answer	Marks
1(f)(i)	1 mark each • Taking a filename as input and calling ReadData() with input • Calling Calculate() and outputting the return value	2
Example program code: Java <pre>TopOfStack = -1; System.out.println("Enter the filename"); Scanner scanner = new Scanner(System.in); String FileName = scanner.nextLine(); ReadData(FileName); Double ReturnValue = Calculate(); System.out.println(ReturnValue);</pre> VB.NET <pre>Console.WriteLine("Enter the filename") Dim FileName As String = Console.ReadLine() ReadData(FileName) Dim ReturnValue As Single ReturnValue = Calculate() Console.WriteLine(ReturnValue)</pre> Python <pre>FileName = input("Enter the filename: ") ReadData(FileName) ReturnValue = Calculate() print(ReturnValue)</pre>		

PUBLISHED

Question	Answer	Marks
1(f)(ii)	1 mark for screenshot showing input of StackData . txt and output of 131 1 mark for screenshot showing input of SecondStack . txt and output of 320	2
e.g. Enter the filename StackData.txt 131.0 Enter the filename SecondStack.txt 320.0		

PUBLISHED

Question	Answer	Marks
2(a)	1 mark each - Record/class NewRecord declared 3 variables within structure (all integer)	2
Example program code: Java <pre> class NewRecord{ private Integer Key; private Integer Item1; private Integer Item2; public NewRecord(Integer pKey, Integer pItem1, Integer pItem2){ Key = pKey; Item1 = pItem1; Item2 = pItem2; } public Integer GetKey(){ return Key; } public Integer GetItem1(){ return Item1; } public Integer GetItem2(){ return Item2; } } </pre> VB.NET <pre> Structure NewRecord Dim Key As Integer Dim Item1 As Integer Dim Item2 As Integer End Structure </pre>		

PUBLISHED

Question	Answer	Marks
	<pre> Python class Record: def __init__(self, pKey, pItem1, pItem2): self.__Key = pKey #integer self.__Item1 = pItem1 #integer self.__Item2 = pItem2 #integer def GetKey(self): return self.__Key def GetItem1(self): return self.__Item1 def GetItem2(self): return self.__Item2 </pre>	

PUBLISHED

Question	Answer	Marks
2(b)(i)	1 mark for • HashTable (200 records) and Spare (100 records) declared as (global) arrays	1
Example program code: Java <pre>public static NewRecord[] HashTable = new NewRecord[200]; public static NewRecord[] Spare = new NewRecord[100];</pre> VB.NET <pre>Dim HashTable(199) As NewRecord Dim Spare(99) As NewRecord</pre> Python <pre>HashTable = [] Spare = []</pre>		

PUBLISHED

Question	Answer	Marks
2(b)(ii)	1 mark each • Procedure <code>Initialise()</code> header (and close where appropriate) that initialises all elements in both arrays ... • ... to an empty record with <code>-1</code> in each of the 3 fields/elements	2
Example program code: Java <pre>public static void Initialise(){ NewRecord EmptyRecord = new NewRecord(-1, -1, -1); for(Integer X = 0; X < 200; X++){ HashTable[X] = EmptyRecord; } for(Integer X = 0; X < 100; X++){ Spare[X] = EmptyRecord; } }</pre> VB.NET <pre>Sub Initialise() Dim EmptyRecord As NewRecord EmptyRecord.Key = -1 EmptyRecord.Item1 = -1 EmptyRecord.Item2 = -1 For X = 0 To 199 HashTable(X) = EmptyRecord Next For X = 0 To 99 Spare(X) = EmptyRecord Next End Sub</pre>		

PUBLISHED

Question	Answer	Marks
	<pre>Python def Initialise(): global HashTable global Spare for X in range(200): HashTable.append(Record(-1, -1, -1)) for X in range(100): Spare.append(Record(-1, -1, -1))</pre>	

PUBLISHED

Question	Answer	Marks
2(c)	1 mark each - Function header (and close where appropriate), taking one (integer) parameter and returning the calculated value - Calculation of parameter MOD 200	2
Example program code: Java <pre>public static Integer CalculateHash(Integer TheKey){ return(TheKey % 200); }</pre> VB.NET <pre>Function CalculateHash(Key) Return Key Mod 200 End Function</pre> Python <pre>def CalculateHash(Key): return Key % 200</pre>		

PUBLISHED

Question	Answer	Marks
2(d)	1 mark each: • Procedure header (and end where appropriate) taking one record as a parameter • Calling CalculateHash () with key from parameter record and storing/using return value • Checking if HashTable at return value from CalculateHash is empty record ... • ... if it is empty, store parameter in location • ... otherwise, locating next free space in Spare ... • ... and storing in that index only (i.e. not in all other free spaces)	6

PUBLISHED

Question	Answer	Marks
	Example program code: Java <pre> public static void InsertIntoHash(NewRecord TheRecord){ Integer HashValue = CalculateHash(TheRecord.GetKey()); if(HashTable[HashValue].GetKey().equals(-1)){ HashTable[HashValue] = TheRecord; }else{ for(Integer X = 0; X < 99; X++){ if(Spare[X].GetKey().equals(-1)){ Spare[X] = TheRecord; X = 99; } } } } </pre> VB.NET <pre> Sub InsertIntoHash(TheRecord) Dim HashValue As Integer = CalculateHash(TheRecord.Key) If HashTable(HashValue).Key = -1 Then HashTable(HashValue) = TheRecord Else For X = 0 To 99 If Spare(X).Key = -1 Then Spare(X) = TheRecord X = 100 End If Next End If End Sub </pre>	

PUBLISHED

Question	Answer	Marks
	<pre> Python def InsertIntoHash(TheRecord): global HashTable global Spare HashValue = CalculateHash(TheRecord.GetKey()) if HashTable[HashValue].GetKey() == -1: HashTable[HashValue] = TheRecord else: for x in range(0, 100): if Spare[x].GetKey() == -1: Spare[x] = TheRecord break </pre>	

PUBLISHED

Question	Answer	Marks
2(e)	1 mark each to max 5 • Procedure header (and end where appropriate), opening and closing file HashData.txt • Reading in all lines of data ... • ... splitting each line by commas • Creating record with correct values with each line read in from file • Calling InsertIntoHash() with each record they have created • Exception handling for opening and reading from file with appropriate catch and output.	5

Example program code:

Java

```
public static void CreateHashTable(){

    String[] Data = new String[3];
    Integer NewKey;
    Integer NewItem1;
    Integer NewItem2;
    try{
        FileReader File = new FileReader("HashData.txt");
        try{
            BufferedReader Reader = new BufferedReader(File);
            String Line= Reader.readLine();
            while (Line != null){
                Line = Line.replace("\n", "");
                Data = Line.split(",");
                NewKey = Integer.parseInt(Data[0]);
                NewItem1 = Integer.parseInt(Data[1]);
                NewItem2 = Integer.parseInt(Data[2]);
                NewRecord ReadData = new NewRecord(NewKey, NewItem1, NewItem2);
                InsertIntoHash(ReadData);
                Line= Reader.readLine();
            }
            Reader.close();
        }catch(IOException ex){}
    }catch(FileNotFoundException e){System.out.println("File not found");}

}
```

PUBLISHED

Question	Answer	Marks
	<pre> VB.NET Sub CreateHashTable() Dim Line As String Dim Data(3) As String Dim TheRecord As NewRecord Try Dim FileReader As New System.IO.StreamReader("HashData.txt") While Not FileReader.EndOfStream Line = FileReader.ReadLine() Data = Split(Line, ",") TheRecord.Key = Integer.Parse(Data(0)) TheRecord.Item1 = Integer.Parse(Data(1)) TheRecord.Item2 = Integer.Parse(Data(2)) InsertIntoHash(TheRecord) End While FileReader.Close() Catch ex As Exception Console.WriteLine("Cannot open file") End Try End Sub Python def CreateHashTable(): global HashTable global Spare try: File = open("HashData.txt") for Line in File: Data = Line.strip() Data = Line.split(",") InsertIntoHash(Record(int(Data[0]), int(Data[1]), int(Data[2]))) File.close() except: print("Cannot open file") </pre>	

PUBLISHED

Question	Answer	Marks
2(f)(i)	1 mark each • Procedure header (and end where appropriate) and looping through each element in Spare ... • ... checking if record is empty and outputting key field if not empty	2

Example program code:

Java

```
public static void PrintSpare(){
    Integer X = 0;
    while(Spare[X].GetKey() != -1){
        System.out.println(Spare[X].GetKey());
        X++;
    }
}
```

VB.NET

```
Sub PrintSpare()
    Dim X As Integer = 0
    While Spare(X).Key <> -1
        Console.WriteLine(Spare(X).Key)
        X = X + 1
    End While
End Sub
```

Python

```
def PrintSpare():
    global Spare
    X = 0
    while Spare[X].GetKey() != -1:
        print(Spare[X].GetKey())
        X +=1
```

PUBLISHED

Question	Answer	Marks
2(f)(ii)	1 mark for calling Initialise() then CreateHashTable() then PrintSpare()	1
Example program code: Java Initialise(); CreateHashTable(); PrintSpare(); VB.NET Initialise() CreateHashTable() PrintSpare() Python Initialise() CreateHashTable() PrintSpare()		

PUBLISHED

Question	Answer	Marks
2(f)(iii)	1 mark for output	1
For example: <pre> 915 136 415 55 349 775 846 469 246 752 8 889 695 292 417 181 810 972 781 46 120 95 378 433 994 946 586 365 615 580 128 944 88 2 362 689 629 962 986 704 31 948 823 435 527 195 141 981 905 320 965 658 697 831 15 408 24 947 977 830 88 </pre>		

PUBLISHED

Question	Answer	Marks
3(a)(i)	1 mark each • Class header (and end when appropriate) • Four attributes with appropriate data types • Constructor header (and end where appropriate) within class taking (min) 4 parameters ... • ... assigning each parameter to its attribute	4

PUBLISHED

Question	Answer	Marks
	Example program code: <pre> Java class Animal{ public String Name; public String Sound; public Integer Size; public Integer Intelligence; public Animal(String pName, String pSound, Integer pSize, Integer pIntelligence){ Name = pName; Sound = pSound; Size = pSize; Intelligence = pIntelligence; } } VB.NET Class Animal Public Name As String Public Sound As String Public Size As Integer Public Intelligence As Integer Sub New(pName, pSound, pSize, pIntelligence) Name = pName Sound = pSound Size = pSize Intelligence = pIntelligence End Sub End Class </pre>	

PUBLISHED

Question	Answer	Marks
	<pre>Python class Animal: def __init__(self, pName, pSound, pSize, pIntelligence): self.Name = pName #string self.Sound = pSound #string self.Size = pSize #integer self.Intelligence = pIntelligence #integer</pre>	

PUBLISHED

Question	Answer	Marks
3(a)(ii)	1 mark each • Description() method header (and end where appropriate) with no parameter • Concatenating the attributes with the given message ... • ... and returning the created message	3
Example program code: Java <pre>public String Description(){ String Message = "The animal's name is " + Name + ", it makes a " + Sound + ", its size is " + Size + " and its intelligence level is " + Intelligence; return Message; }</pre> VB.NET <pre>Function Description() Dim Message As String = "The animal's name is " & Name & ", it makes a " & Sound & ", its size is " & CStr(Size) & " and its intelligence level is " & CStr(Intelligence) Return Message End Function</pre> Python <pre>def Description(self): Message = "The animal's name is " + self.Name + ", it makes a " + self.Sound + ", its size is " + str(self.Size) + " and its intelligence level is " + str(self.Intelligence) return Message</pre>		

PUBLISHED

Question	Answer	Marks
3(b)(i)	1 mark each • Class header (and end where appropriate) inherits from Animal • Constructor header (and end where appropriate) taking 6 parameters within class and calling parent constructor with the four parameters ... • ... WingSpan and NumberWords attributes defined with data types and parameters assigned within constructor • ChangeNumberWords() method header (and end where appropriate) takes one parameter and adds parameter to attribute NumberWords	4
Example program code: Java <pre>class Parrot extends Animal{ public Integer WingSpan; public Integer NumberWords; public Parrot(String pName, String pSound, Integer pSize, Integer pIntelligence, Integer pWingSpan, Integer pNumberWords){ super(pName, pSound, pSize, pIntelligence); WingSpan = pWingSpan; NumberWords = pNumberWords; } public void ChangeNumberWords(Integer Change){ NumberWords = NumberWords + Change; } }</pre> VB.NET <pre>Class Parrot Inherits Animal Dim WingSpan As Integer Dim NumberWords As Integer Sub New(pName, pSound, pSize, pIntelligence, pWingSpan, pNumberWords) MyBase.New(pName, pSound, pSize, pIntelligence) WingSpan = pWingSpan NumberWords = pNumberWords End Sub</pre>		

PUBLISHED

Question	Answer	Marks
	<pre> Sub ChangeNumberWords(Change) NumberWords = NumberWords + Change End Sub End Class Python class Parrot(Animal): def __init__(self, pName, pSound, pSize, pIntelligence, pWingSpan, pNumberWords): super().__init__(pName, pSound, pSize, pIntelligence) self.WingSpan = pWingSpan #integer self.NumberWords = pNumberWords #integer def ChangeNumberWords(self, Change): self.NumberWords = self.NumberWords + Change </pre>	

PUBLISHED

Question	Answer	Marks
3(b)(ii)	1 mark each - Description() method header (and end where appropriate) taking no parameters and overriding/overloads/extending/using parent method - Concatenating and returning the correct string	2

Example program code:

Java

```
public String Description(){
    String Message = "The animal's name is " + Name + ", it makes a " + Sound + ", its size is " + Size
+ " and its intelligence level is " + Intelligence + ". It has a wingspan of " + WingSpan + "cm and can say
" + NumberWords + " words.";
    return Message;
}
```

VB.NET

Overloads Function Description()

```
Dim Message As String = "The animal's name is " & Name & ", it makes a " & Sound & ", its size is " &
CStr(Size) & " and its intelligence level is " & CStr(Intelligence) & ". It has a wingspan of " &
CStr(WingSpan) & "cm and can say " & CStr(NumberWords) & " words."
Return Message
End Function
```

Python

```
def Description(self):
    Message = "The animal's name is " + self.Name + ", it makes a " + self.Sound + ", its size is " +
str(self.Size) + " and its intelligence level is " + str(self.Intelligence) + ". It has a wingspan of " +
str(self.WingSpan) + "cm and can say " + str(self.NumberWords) + " words."
    return Message
```

PUBLISHED

Question	Answer	Marks
3(c)(i)	1 mark each • Class header (and end where appropriate) inherits from Animal • Constructor header (and end where appropriate) taking 5 parameters within class and calling parent constructor with parameters • Attribute Territory defined as int and parameter assigned within constructor • SetTerritory() method header (and end) takes 1 parameter and adds parameter to attribute TerritorySize	4
Example program code: Java <pre>class Wolf extends Animal{ public Integer TerritorySize; public Wolf(String pName, String pSound, Integer pSize, Integer pIntelligence, Integer pTerritorySize){ super(pName, pSound, pSize, pIntelligence); TerritorySize = pTerritorySize; } public void SetTerritorySize(Integer Change){ TerritorySize = TerritorySize + Change; } }</pre> VB.NET <pre>Class Wolf Inherits Animal Dim TerritorySize As Integer Sub New(pName, pSound, pSize, pIntelligence, pTerritory) MyBase.New(pName, pSound, pSize, pIntelligence) TerritorySize = pTerritory End Sub Sub SetTerritorySize(Change) TerritorySize = TerritorySize + Change End Sub End Class</pre>		

PUBLISHED

Question	Answer	Marks
	<pre> Python class Wolf(Animal): def __init__(self, pName, pSound, pSize, pIntelligence, pTerritorySize): super().__init__(pName, pSound, pSize, pIntelligence) self.TerritorySize = pTerritorySize #integer def SetTerritorySize(self, Change): self.TerritorySize = self.TerritorySize + Change </pre>	

PUBLISHED

Question	Answer	Marks
3(c)(ii)	1 mark each • Description() method header (and end where appropriate) taking no parameters and overriding/overloads/extending/using parent method • Concatenating and return correct message	2

Example program code:

Java

```
public String Description(){
    String Message = "The animal's name is " + Name + ", it makes a " + Sound + ", its size is " + Size
+ " and its intelligence level is " + Intelligence + ". Its territory is " + TerritorySize + " square
miles.";
    return Message;
}
```

VB.NET

```
Overloads Function Description()
    Dim Message As String = "The animal's name is " & Name & ", it makes a " & Sound & ", its size is " &
CStr(Size) & " and its intelligence level is " & CStr(Intelligence) + ". Its territory is " &
CStr(TerritorySize) & " square miles."
    Return Message
End Function
```

Python

```
def Description(self):
    Message = "The animal's name is " + self.Name + ", it makes a " + self.Sound + ", its size is " +
str(self.Size) + " and its intelligence level is " + str(self.Intelligence) + " it's territory is " +
str(self.TerritorySize) + " square miles."
    return Message
```

PUBLISHED

Question	Answer	Marks
3(d)(i)	1 mark each 1 correct instance created and stored in a suitable variable/structure 2nd and 3rd correct instances created and stored in a suitable variable/structure	2
Example program code: Java <pre>Parrot Animal1 = new Parrot("Chewie", "Squawk", 1, 10, 30, 29); Wolf Animal2 = new Wolf("Nighteyes", "Howl", 8, 7, 100); Animal Animal3 = new Animal("Copper", "Neigh", 10, 6);</pre> VB.NET <pre>Dim Animal1 As Parrot Animal1 = New Parrot("Chewie", "Squawk", 1, 10, 30, 29) Dim Animal2 As Wolf Animal2 = New Wolf("Nighteyes", "Howl", 8, 7, 100) Dim Animal3 As Animal Animal3 = New Animal("Copper", "Neigh", 10, 6)</pre> Python <pre>Animal1 = Parrot("Chewie","Squawk",1,10,30,29) Animal2 = Wolf("Nighteyes","Howl",8,7,100) Animal3 = Animal("Copper", "Neigh", 10, 6)</pre>		

PUBLISHED

Question	Answer	Marks
3(d)(ii)	1 mark each • Calling SetTerritorySize(-20) for instance of Nighteyes • Calling ChangeNumberWords(2) for instance of Chewie • Calling Description() for all 3 animals (after any updates) and outputting return values	3
Example program code: Java <pre>Animal2.SetTerritorySize(-20); Animal1.ChangeNumberWords(2); System.out.println(Animal1.Description()); System.out.println(Animal2.Description()); System.out.println(Animal3.Description());</pre> VB.NET <pre>Animal2.SetTerritorySize(-20) Animal1.ChangeNumberWords(2) Console.WriteLine(Animal1.Description()) Console.WriteLine(Animal2.Description()) Console.WriteLine(Animal3.Description())</pre> Python <pre>Animal2.SetTerritorySize(-20) Animal1.ChangeNumberWords(2) print(Animal1.Description()) print(Animal2.Description()) print(Animal3.Description())</pre>		

PUBLISHED

Question	Answer	Marks
3(d)(iii)	1 mark each: • All three messages accurate with all relevant values • ... showing correctly updated territory for Nighteyes and words for Chewie	2
e.g. The animal's name is Chewie, it makes a Squawk, its size is 1 and its intelligence level is 10. It has a wingspan of 30cm and can say 31 words. The animal's name is Nighteyes, it makes a Howl, its size is 8 and its intelligence level is 7 it's territory is 80 square miles. The animal's name is Copper, it makes a Neigh, its size is 10 and its intelligence level is 6		