Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/42**

Paper 4 Practical **October/November 2023**

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 October/November 2023 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

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 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)(i)	**One** mark each
	• Two arrays with correct identifiers of type string/character
	• Each has 100 elements
	
	Example program code:
	
	Java
	
	`public static String[] StackVowel = new String[100];`
	`public static String[] StackConsonant = new String[100];`
	
	VB.NET
	
	`Dim StackVowel(0 To 99) As Char`
	`Dim StackConsonant(0 To 99) As Char`
	
	Python
	
	`StackVowel = [] #string 100`
	`StackConsonant = [] #string 100`
		2

Question	Answer	Marks

1(a)(ii)	**One** mark for
	• Declaring both variables as type integer global and initialised to 0
	
	Example program code:
	
	Java
	
	`public static Integer VowelTop = 0;`
	`public static Integer ConsonantTop = 0;`
	
	VB.NET
	
	`Dim VowelTop As Integer = 0`
	`Dim ConsonantTop As Integer = 0`
	
	Python
	
	`global VowelTop #integer`
	`global ConsonantTop #integer`
	`#main`
	`VowelTop = 0`
	`ConsonantTop = 0`
		1

Question	Answer	Marks

1(b)(i)	**One** mark each
	
	• Procedure `PushData()` heading (and end where appropriate) taking one parameter
	
	• Checking if parameter is a (lowercase) vowel …
	• … checking if `StackVowel` is full and outputting suitable message
	• … otherwise inserting parameter in next position
	• … incrementing `VowelTop`
	
	• Repeated for `Consonant`
	
	Example program code:
	
	Java
	
	`public static void PushData(String Letter){`
	`  if(Letter.equals("a") || Letter.equals("e") || Letter.equals("i") || Letter.equals("o") || Letter.equals("u")){`
	`  if(VowelTop == 100){`
	`  System.out.println("Vowel stack full");`
	`  }else{`
	`  StackVowel[VowelTop] = Letter;`
	`  VowelTop++;`
	`  }`
	`  }else{`
	`  if(ConsonantTop == 100){`
	`  System.out.println("Consonant stack full");`
	`  }else{`
	`  StackConsonant[ConsonantTop] = Letter;`
	`  ConsonantTop++;`
	`  }`
	`  }`
	`}`
		6

Question	Answer	Marks

1(b)(i)	VB.NET
	`Sub PushData(Letter As Char)`
	`  If Letter = "a" Or Letter = "e" Or Letter = "i" Or Letter = "o" Or Letter = "u" Then`
	`  If VowelTop = 100 Then`
	`  Console.WriteLine("Vowel stack full")`
	`  Else`
	`  StackVowel(VowelTop) = Letter`
	`  VowelTop += 1`
	`  End If`
	`  Else`
	`  If ConsonantTop = 100 Then`
	`  Console.WriteLine("Consonant stack full")`
	`  Else`
	`  StackConsonant(ConsonantTop) = Letter`
	`  ConsonantTop += 1`
	`  End If`
	`  End If`
	`End Sub`
	Python
	`def PushData(Letter):`
	`  global VowelTop`
	`  global ConsonantTop`
	`  if Letter == "a" or Letter == "e" or Letter == "i" or Letter == "o" or Letter == "u":`
	`  if VowelTop == 100:`
	`  print("Vowel stack full")`
	`  else:`
	`  StackVowel.append(Letter)`
	`  VowelTop = VowelTop + 1`
	`  else:`
	`  if ConsonantTop == 100:`
	`  print("Consonant stack full")`
	`  else:`
	`  StackConsonant.append(Letter)`
	`  ConsonantTop = ConsonantTop + 1`	

Question	Answer	Marks

1(b)(ii)	**One** mark each
	• Procedure header `ReadData()` with no parameter
	
	• Opening `StackData.txt` to read and closing file
	• Looping until EOF // Looping 100 times
	• Read each item from the file
	• Calling `PushData()` with each value as parameter
	
	• Appropriate exception handling with suitable output
	
	Example program code:
	
	Java
	
	`private static void ReadData(){`
	`  try{`
	`  Scanner Scanner1 = new Scanner(new File("StackData.txt"));`
	`  while(Scanner1.hasNextLine()){`
	`  PushData(Scanner1.next());`
	`  }`
	`  Scanner1.close();`
	`  }catch(FileNotFoundException ex){`
	`  System.out.println("No file found");`
	`  }`
	`}`
	
	VB.NET
	
	`Sub ReadData()`
	`  Try`
	`  Dim DataReader As New System.IO.StreamReader("StackData.txt")`
	
	`  Do Until DataReader.EndOfStream`
	`  PushData(DataReader.ReadLine())`
	`  Loop`
	`  DataReader.Close()`
		6

Question	Answer	Marks

1(b)(ii)	Catch ex As Exception
	Console.WriteLine("File not found")
	End Try
	
	End Sub
	
	Python
	
	def ReadData():
	try:
	DataFile = open("StackData.txt")
	for Line in DataFile:
	PushData(Line.strip())
	DataFile.close()
	except:
	print("File not found")	

Question	Answer	Marks

1(c)	**One** mark each
	• One function header with no parameter
	
	• Checking if stack is empty and **returning** `"No data"`
	• …otherwise, decrementing correct pointer
	• Returning value at top of stack
	
	• 2nd function fully correct
	
	Example program code:
	
	Java
	
	`public static String PopVowel(){`
	`  String DataToReturn = "";`
	`  if(VowelTop - 1 >= 0){`
	`  VowelTop --;`
	`  DataToReturn = StackVowel[VowelTop];`
	`  return DataToReturn;`
	`  }else{`
	`  return "No data";`
	`  }`
	`}`
	
	`public static String PopConsonant(){`
	`  String DataToReturn = "";`
	`  if(ConsonantTop - 1 >= 0){`
	`  ConsonantTop--;`
	`  DataToReturn = StackConsonant[ConsonantTop];`
	`  return DataToReturn;`
	`  }else{`
	`  return "No data";`
	`  }`
	`}`
		5

Question	Answer	Marks

1(c)	VB.NET
	
	`Function PopVowel()`
	`  If VowelTop - 1 >= 0 Then`
	`  VowelTop -= 1`
	`  Dim DataToReturn As Char = StackVowel(VowelTop)`
	`  Return DataToReturn`
	`  Else`
	`  Return "No data"`
	`  End If`
	`End Function`
	
	`Function PopConsonant()`
	`  If ConsonantTop - 1 >= 0 Then`
	`  ConsonantTop -= 1`
	`  Dim DataToReturn As Char = StackConsonant(ConsonantTop)`
	`  Return DataToReturn`
	`  Else`
	`  Return "No data"`
	`  End If`
	`End Function`
	
	Python
	
	`def PopVowel():`
	`  global VowelTop`
	`  global ConsonantTop`
	`  if VowelTop - 1 >= 0:`
	`  VowelTop = VowelTop - 1`
	`  DataToReturn = StackVowel[VowelTop]`
	`  del StackVowel[-1]`
	`  return DataToReturn`
	`  else:`
	`  return "No data"`	

Question	Answer	Marks

1(c)	`def PopConsonant():`
	`  global VowelTop`
	`  global ConsonantTop`
	`  if ConsonantTop - 1 >= 0:`
	`  ConsonantTop = ConsonantTop - 1`
	`  DataToReturn = StackConsonant[ConsonantTop]`
	`  del StackConsonant[-1]`
	`  return DataToReturn`
	`  else:`
	`  return "No data"`	

Question	Answer	Marks

1(d)(i)	**One** mark each to **max 6**
	• Calling `ReadData()`
	• Looping until 5 letters **successfully accessed**
	
	• Prompt and read in input of choice …
	• … if vowel is input calling `PopVowel()` and if consonant calling `PopConsonant()` …
	• … storing return values
	• Outputting appropriate message if no vowels and if no consonants (stacks full) **within loop**
	• Outputting the **five returned letters** on one line
	
	Example program code:
	
	Java
	
	`public static void main(String args[]){`
	`  VowelTop = 0;`
	`  ConsonantTop = 0;`
	`  ReadData();`
	`  String Letters = "";`
	`  Boolean Flag = false;`
	`  String Choice = "";`
	`  String DataAccessed = "";`
	`  for(Integer X = 0; X < 5; X++){`
	`  Flag = false;`
	`  while(Flag == false){`
	`  System.out.println("Vowel or Consonant");`
	`  Scanner scanner = new Scanner(System.in);`
	`  Choice = (scanner.nextLine()).toLowerCase();`
	`  if(Choice.equals("vowel")){`
	`  DataAccessed = PopVowel();`
	`  if(DataAccessed.equals("No data") == false){`
	`  Letters = Letters + DataAccessed;`
	`  Flag = true;`
	`  }else{`
	`  System.out.println("No vowels left");`
	`  }`
		6

Question	Answer	Marks
1(d)(i)		

```
            }else if(Choice.equals("consonant")){
               DataAccessed = PopConsonant();
               if(DataAccessed.equals("No data") == false){
                  Letters = Letters + DataAccessed;
                  Flag = true;
               }else{
                  System.out.println("No consonants left");
               }
            }
         }
      }
      System.out.println(Letters);
   }
```

VB.NET

```
Sub Main(args As String())
   VowelTop = 0
   ConsonantTop = 0
   ReadData()
   Dim Letters As String = ""
   Dim Flag As Boolean = False
   Dim Choice As String
   Dim DataAccessed As String
   For x = 0 To 4
      Flag = False
      While Flag = False
         Console.WriteLine("Vowel or Consonant?")
         Choice = Console.ReadLine().ToLower()
         If Choice = "vowel" Then
            DataAccessed = PopVowel()
            If DataAccessed <> "No data" Then
               Letters = Letters & DataAccessed
               Flag = True
            Else
               Console.WriteLine("No vowels left")
```

Question	Answer	Marks
1(d)(i)		

```
            End If
            ElseIf Choice = "consonant" Then
            DataAccessed = PopConsonant()
            If DataAccessed <> "No data" Then
               Letters = Letters & DataAccessed
               Flag = True
            Else
               Console.WriteLine("No consonants left")
            End If
         End If
      End While
   Next
   Console.WriteLine(Letters)
   End Sub
```

Python

```
#main
VowelTop = 0
ConsonantTop = 0
ReadData()
Letters = ""
Flag = False

for x in range(0, 5):
   Flag = False
   while Flag == False:
      Choice = input("Vowel or Consonant").lower()
      if Choice == "vowel":
         DataAccessed = PopVowel()
         if DataAccessed != "No data":
            Letters = Letters + DataAccessed
            Flag = True
         else:
            print("No vowels left")
```

Question	Answer	Marks

1(d)(i)	`elif Choice == "consonant":`
	`  DataAccessed = PopConsonant()`
	`  if DataAccessed != "No data":`
	`  Letters = Letters + DataAccessed`
	`  Flag = True`
	`else:`
	`  print("No consonants left")`
	
	`print(Letters)`	

1(d)(ii)	**One** mark showing input in order vowel, cons, cons, vowel, vowel. Output is then **utxoe**
	
	e.g.
	`Vowel or Consonantvowel`
	`Vowel or Consonantconsonant`
	`Vowel or Consonantconsonant`
	`Vowel or Consonantvowel`
	`Vowel or Consonantvowel`
	`utxoe`
		1

Question	Answer	Marks

2(a)(i)	**One** mark each
	• Function header with parameter
	• Correct loop
	• Modulus calculation
	• Return of correct value at correct place
	• Remainder of function correct
	
	Example program code:
	
	Java
	
	`public static Integer IterativeCalculate(Integer Number){`
	`  Integer ToFind = Number;`
	`  Integer Total = 0;`
	`  while(Number != 0){`
	`  if(ToFind % Number == 0){`
	`  Total += Number;`
	`  }`
	
	`  Number--;`
	`  }`
	`  return Total;`
	`}`
	
	VB.NET
	
	`Function IterativeCalculate(Number As Integer)`
	`  Dim total As Integer = 0`
	`  Dim ToFind As Integer = Number`
	`  While Number <> 0`
	`  If ToFind Mod Number = 0 Then`
	`  total = total + Number`
	`  End If`
	`  Number = Number - 1`
	`  End While`
	`  Return total`
	`End Function`
		5

Question	Answer	Marks

2(a)(i)	Python
	
	`def IterativeCalculate(Number):`
	`  Total = 0`
	`  ToFind = Number`
	`  while Number != 0:`
	`  if ToFind % Number == 0:`
	`  Total = Total + Number`
	`  Number = Number - 1`
	`  return Total`	

2(a)(ii)	**One** mark each
	• Calling `IterativeCalculate(10)`
	• Outputting return value
	
	Example program code:
	
	Java
	`System.out.println(IterativeCalculate(10));`
	
	VB.NET
	`Sub Main(args As String())`
	
	`  Console.WriteLine(IterativeCalculate(10))`
	`End Sub`
	
	Python
	`print(IterativeCalculate(10))`
		2

2(a)(iii)	**One** mark for screenshot showing **18**	**1**

Question	Answer	Marks

2(b)(i)	**One** mark for each gap (5)
	One mark for recursive calls both accurate and in correct places
	One mark for remainder of function with nothing superfluous
	
	`FUNCTION RecursiveValue(Number : Integer, ToFind : Integer) RETURNS INTEGER`
	`  IF Number = 0 THEN`
	`  RETURN 0`
	`  ELSE`
	`  IF ToFind MODULUS Number = 0 THEN`
	`  RETURN Number + RecursiveValue(Number - 1, ToFind)`
	`  ELSE`
	`  RETURN RecursiveValue(Number - 1, ToFind)`
	`  ENDIF`
	`  ENDIF`
	`ENDFUNCTION`
	
	Example program code:
	
	Java
	
	`  public static Integer RecursiveValue(Integer Number, Integer ToFind){`
	`  if(Number == 0){`
	`  return 0;`
	`  }else{`
	`  if(ToFind % Number == 0){`
	`  return Number + RecursiveValue(Number - 1, ToFind);`
	`  }else{`
	`  return RecursiveValue(Number - 1, ToFind);`
	`  }`
	`  }`
	`  }`
		7

Question	Answer	Marks

2(b)(i)	VB.NET
	
	`Function RecursiveValue(Number As Integer, ToFind As Integer)`
	`  If Number = 0 Then`
	`  Return 0`
	`  Else`
	`  If ToFind Mod Number = 0 Then`
	`  Return Number + RecursiveValue(Number - 1, ToFind)`
	`  Else`
	`  Return RecursiveValue(Number - 1, ToFind)`
	`  End If`
	`  End If`
	`End Function`
	
	Python
	
	`def RecursiveValue(Number, ToFind):`
	`  if Number == 0:`
	`  return 0`
	`  else:`
	`  if ToFind % Number == 0:`
	`  return Number + RecursiveValue(Number - 1, ToFind)`
	`  else:`
	`  return RecursiveValue(Number - 1, ToFind)`	

Question	Answer	Marks

2(b)(ii)	**One** mark for calling `RecursiveValue(50,50)` and outputting return value
	
	Example program code:
	
	Java
	
	`System.out.println(RecursiveValue(50,50));`
	
	VB.NET
	
	`Console.WriteLine(RecursiveValue(50,50))`
	
	Python
	
	`print(RecursiveValue(50,50))`
		1

2(b)(iii)	**One** mark for screenshot showing **93**	**1**

Question	Answer	Marks

3(a)(i)	**One** mark each:
	• Class declaration
	• Four attributes with correct data types
	• Constructor header
	• … taking 4 parameters
	• Setting attributes to parameter values
	
	Example program code:
	
	Java
	
	`class Character{`
	`  private String CharacterName;`
	`  private Date DateOfBirth;`
	`  private Double Intelligence;`
	`  private Integer Speed;`
	
	`  public Character(String CName, Date DBirth, Double Intell, Integer SpeedP){`
	`  CharacterName = CName;`
	`  DateOfBirth = DBirth;`
	`  Intelligence = Intell;`
	`  Speed = SpeedP;`
	`  }`
	`}`
	
	VB.NET
	
	`Class Character`
	`  Private CharacterName As String`
	`  Private DateOfBirth As Date`
	`  Private Intelligence As Single`
	`  Private Speed As Integer`
	
	`  Sub New(CName, DBirth, Intell, SpeedP)`
	`  CharacterName = CName`
	`  DateOfBirth = DBirth`
		5

Question	Answer	Marks

3(a)(i)	Intelligence = Intell
	Speed = SpeedP
	End Sub
	End Class
	
	Python
	
	class Character:
	#self.__CharacterName string
	#self.__DateOfBirth date
	#self.__Intelligence real
	#self.__Speed integer
	
	def __init__(self, CName, DBirth, Intell, SpeedP):
	self.__CharacterName = CName
	self.__DateOfBirth = DBirth
	self.__Intelligence = Intell
	self.__Speed = SpeedP	

Question	Answer	Marks

3(a)(ii)	**One** mark each:
	• 1 get header with no parameter …
	• … returning attribute
	• Second correct get method
	
	Example program code:
	
	Java
	
	`public Double GetIntelligence(){`
	`  return Intelligence;`
	`}`
	
	`public String GetName(){`
	`  return CharacterName;`
	`}`
	
	VB.NET
	
	`Function GetIntelligence()`
	`  Return Intelligence`
	`End Function`
	
	`Function GetName()`
	`  Return CharacterName`
	`End Function`
	
	Python
	
	`def GetIntelligence(self):`
	`  return self.__Intelligence`
	
	`def GetName(self):`
	`  return self.__CharacterName`
		3

Question	Answer	Marks

3(a)(iii)	**One** mark each
	• Set header with 1 parameter …
	• … assigns parameter to attribute
	
	Example program code:
	
	Java
	
	`public void SetIntelligence(Double NewValue){`
	`  Intelligence = NewValue;`
	`}`
	
	VB.NET
	
	`Sub SetIntelligence(NewValue)`
	`  Intelligence = NewValue`
	`End Sub+`
	
	Python
	
	`def SetIntelligence(self, NewValue):`
	`  self.__Intelligence = NewValue`
		2

Question	Answer	Marks

3(a)(iv)	**One** mark for method multiplying **attribute** intelligence by 1.1 (or equivalent) and storing in **attribute**.
	
	Example program code:
	
	Java
	
	`public void Learn(){`
	`  Intelligence = Intelligence * 1.1;`
	`}`
	
	VB.NET
	
	`Overridable Sub Learn()`
	`  Intelligence = Intelligence * 1.1`
	`End Sub`
	
	Python
	
	`def Learn(self):`
	`  self.__Intelligence = self.__Intelligence * 1.1`
		1

Question	Answer	Marks

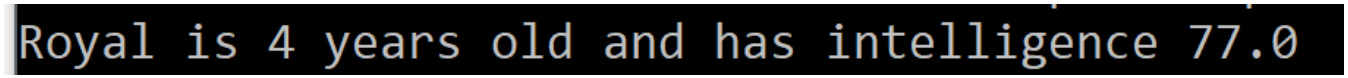
3(a)(v)	**One** mark each
	• Method (function) header (and end where appropriate) no parameter, returning a calculated age
	• Extracting attribute year of birth from date and subtracting from 2023
	
	Example program code:
	
	Java
	
	`public Integer ReturnAge(){`
	`  return 2023 - DateOfBirth.getYear();`
	`}`
	
	VB.NET
	
	`Function ReturnAge()`
	`  Return DateDiff(DateInterval.Year, DateOfBirth, #01/01/2023#)`
	`End Function`
	
	Python
	
	`  def ReturnAge(self):`
	`  return 2023 - self.__DateOfBirth.year`
		2

Question	Answer	Marks

3(b)(i)	**One** mark each:
	• Creating new instance of `Character` with identifier `FirstCharacter` …
	• … sending correct values as parameters
	
	Example program code:
	
	Java
	
	`Character FirstCharacter = new Character("Royal", new Date(2019,01,01), 70.0, 30);`
	
	VB.NET
	
	`Sub Main(args As String())`
	`  Dim FirstCharacter As Character`
	`  FirstCharacter = New Character("Royal", #1/1/2019#, 70, 30)`
	`End Sub`
	
	Python
	
	`FirstCharacter = Character("Royal", datetime.datetime(2019, 1, 1), 70, 30)`
		2

Question	Answer	Marks

3(b)(ii)	**One** mark each
	• Calling `Learn()` for `FirstCharacter`
	• Calling `ReturnAge()` and outputting return value
	• Outputting name and intelligence using gets with suitable message
	
	Example program code:
	
	Java
	
	`FirstCharacter.Learn();`
	`System.out.println(FirstCharacter.GetName() + " is " + FirstCharacter.ReturnAge() + " years old and has intelligence " + FirstCharacter.GetIntelligence());`
	
	VB.NET
	
	`FirstCharacter.Learn()`
	`Console.WriteLine(FirstCharacter.GetName() & " is " & FirstCharacter.ReturnAge() & " years old and has intelligence " & FirstCharacter.GetIntelligence())`
	
	Python
	
	`FirstCharacter.Learn()`
	`print(FirstCharacter.GetName(), "is", FirstCharacter.ReturnAge(), "years old and has intelligence" , FirstCharacter.GetIntelligence())`
		3

3(b(iii)	**One** mark for screenshot with Royal, 4 years, 77 intelligence e.g. 	**1**

Question	Answer	Marks

3(c)(i)	**One** mark each:
	• Class header inheriting from `Character`
	• Declaring `Element` as string
	• Constructor header taking 5 parameters …
	• … calling parent constructor with the 4 parameters
	• … assigning parameter to `Element`
	
	Example program code:
	
	Java
	
	`class MagicCharacter extends Character{`
	`  private String Element;`
	
	`  public MagicCharacter(String ElementP, String CName, Date DBirth, Double Intell, Integer SpeedP){`
	`  super(CName, DBirth, Intell, SpeedP);`
	`  Element = ElementP;`
	`  }`
	`}`
	
	VB.NET
	
	`  Class MagicCharacter`
	`  Inherits Character`
	`  Private Element As String`
	`  Sub New(ElementP, CName, DBirth, Intell, SpeedP)`
	`  MyBase.New(CName, DBirth, Intell, SpeedP)`
	`  Element = ElementP`
	`  End Sub`
	
	`  End Class`
		5

Question	Answer	Marks

3(c)(i)	Python
	
	`class MagicCharacter(Character):`
	`  #self.__Element String`
	
	`  def __init__(self, ElementP, CName, DBirth, Intell, SpeedP):`
	`  super().__init__(CName, DBirth, Intell, SpeedP)`
	`  self.__Element = ElementP`	

Question	Answer	Marks

3(c)(ii)	**One** mark each:
	• Method header overriding parent method but no parameters
	• Checking element value …
	• … correct calculations with attribute intelligence and storing
	
	Example program code:
	
	Java
	
	`public void Learn(){`
	`  if(Element.equals("fire") || Element.equals("water")){`
	`  super.SetIntelligence(super.GetIntelligence() * 1.2);`
	`  }else if(Element.equals("earth")){`
	`  super.SetIntelligence(super.GetIntelligence() * 1.3);`
	`  }else{`
	`  super.SetIntelligence(super.GetIntelligence() * 1.1);`
	`  }`
	`}`
	
	VB.NET
	
	`Overrides Sub Learn()`
	`  If Element = "fire" Or Element = "water" Then`
	`  SetIntelligence(GetIntelligence() * 1.2)`
	`  ElseIf Element = "earth" Then`
	`  SetIntelligence(GetIntelligence() * 1.3)`
	`  Else`
	`  SetIntelligence(GetIntelligence() * 1.1)`
	`  End If`
	
	`End Sub`
		3

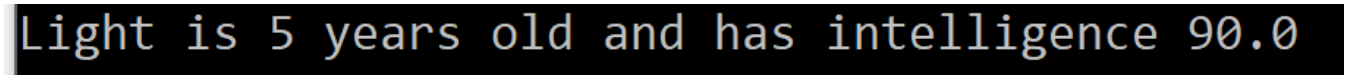
Question	Answer	Marks

3(c)(ii)	Python
	
	`def Learn(self):`
	`  if(self.__Element == "fire" or self.__Element == "water"):`
	`  super().SetIntelligence(super().GetIntelligence() * 1.2)`
	`  elif self.__Element == "earth":`
	`  super().SetIntelligence(super().GetIntelligence() * 1.3)`
	`  else:`
	`  super().SetIntelligence(super().GetIntelligence() * 1.1)`	

3(d)(i)	**One** mark each:
	• Declaring `MagicCharacter` with identifier `FirstMagic` …
	• … with correct parameters
	
	Example program code:
	
	Java
	`MagicCharacter FirstMagic = new MagicCharacter("fire", "Light", new Date(2018,03,03), 75.0, 22);`
	
	VB.NET
	`Dim FirstMagic As MagicCharacter`
	`FirstMagic = New MagicCharacter("fire", "Light", #3/3/2018#, 75, 22)`
	
	Python
	`FirstMagic = MagicCharacter("fire", "Light", datetime.datetime(2018, 3, 3), 75, 22)`
		2

Question	Answer	Marks

3(d)(ii)	**One** mark for calling `Learn()` for `FirstMagic` and outputting all required data in appropriate message using gets.
	
	Example program code:
	
	Java
	
	`FirstMagic.Learn();`
	`System.out.println(FirstMagic.GetName() + " is " + FirstMagic.ReturnAge() + " years old and has intelligence " + FirstMagic.GetIntelligence());`
	
	VB.NET
	
	`FirstMagic.Learn()`
	`Console.WriteLine(FirstMagic.GetName() & " is " & FirstMagic.ReturnAge() & " years old and has intelligence " & FirstMagic.GetIntelligence())`
	
	Python
	
	`FirstMagic.Learn()`
	`print(FirstMagic.GetName(), "is", FirstMagic.ReturnAge(), "years old and has intelligence", FirstMagic.GetIntelligence())`
		1

3(d)(iii)	**One** mark for screenshot e.g. 	**1**