

---

**COMPUTER SCIENCE**

**9608/41**

Paper 4 Written Paper

**October/November 2017**

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.

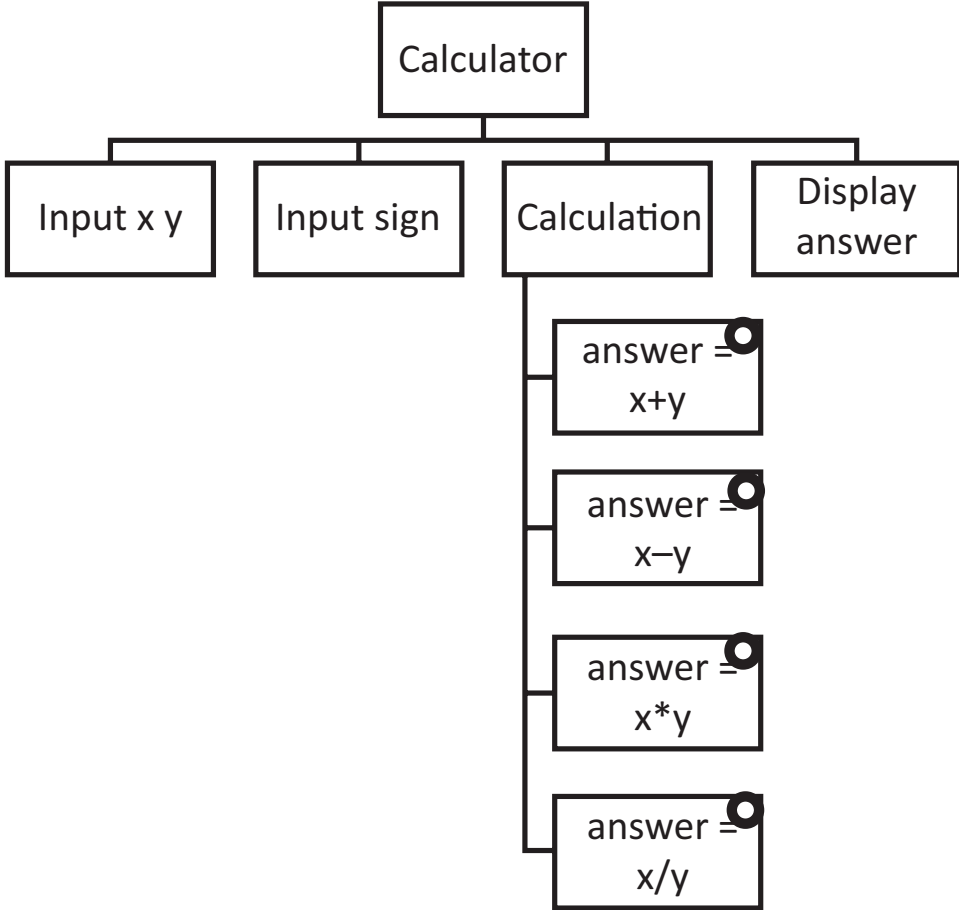
Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the October/November 2017 series for most Cambridge IGCSE<sup>®</sup>, Cambridge International A and AS Level components and some Cambridge O Level components.

Question	Answer	Marks
1	<p>1 mark for each completed statement</p>	7

Question	Answer	Marks
2(a)(i)	<ul style="list-style-type: none"> <li>• Asterisk (*) in the corner/top of the box(es)</li> </ul>	1
2(a)(ii)	<ul style="list-style-type: none"> <li>• Circle (o) in the corner/top of box(es)</li> </ul>	1

Question	Answer	Marks
2(b)	<p>1 mark per bullet</p> <ul style="list-style-type: none"> <li>• Inputting 2 numbers, stored in x and y</li> <li>• Inputting sign Selection used for all four calculations</li> <li>• .. underneath an appropriate box at level 1</li> <li>• Displaying the answer</li> </ul> <p>For example:</p>  <pre> graph TD     Calculator[Calculator] --&gt; InputXY[Input x y]     Calculator --&gt; InputSign[Input sign]     Calculator --&gt; Calculation[Calculation]     Calculator --&gt; DisplayAnswer[Display answer]     Calculation --&gt; AnswerPlus[answer = x+y]     Calculation --&gt; AnswerMinus[answer = x-y]     Calculation --&gt; AnswerMultiply[answer = x*y]     Calculation --&gt; AnswerDivide[answer = x/y]     </pre>	<b>5</b>

Question	Answer	Marks
3(a)	<p>1 mark per clause</p> <ul style="list-style-type: none"> <li>• person(mimi) .</li> <li>• food(lettuce) .</li> <li>• likes(mimi, chocolate) .</li> <li>• dislikes(mimi, sushi) .</li> <li>• dislikes(mimi, lettuce) .</li> </ul>	5
3(b)	<p>1 mark per answer chocolate, pizza</p>	2
3(c)	<p>1 mark per bullet</p> <ul style="list-style-type: none"> <li>• might_like(<b>B,A</b>)</li> <li>• Person(B)</li> <li>• Food(A)</li> <li>• AND</li> <li>• AND NOT</li> <li>• Dislikes predicate</li> </ul> <p>For example:</p> <pre> might_like(B, A) .            {            }  IF person(B) AND food(A)    {      {      }    {      {      }  AND NOT(dislikes(B, A)) .  {      {      } </pre>	6

Question	Answer				Marks		
4(a)	<b>Label</b>	<b>Op code</b>	<b>Operand</b>	<b>Comment</b>	<b>Marks</b>	<b>11</b>	
	START:	LDM	#63	// load ASCII value for '?'			
		OUT		// OUTPUT '?'	1		
		IN		// input GUESS	1		
		CMP	LETTERTOGUESS	// compare with stored letter	1		
		JPE	GUESSED	// if correct guess, go to GUESSED	1		
		LDD	ATTEMPTS	// increment ATTEMPTS	1		
		INC	ACC		1		
		STO	ATTEMPTS		1		
		CMP	#9	// is ATTEMPTS = 9 ?	1		
		JPE	ENDP	// if out of guesses, go to ENDP	1		
		JMP	START	// go back to beginning of loop	1		
	GUESSED:	LDM	#42	// load ASCII for '*'			
		OUT		// OUTPUT '*'	1		
	ENDP:	END		// end program			
	ATTEMPTS:		0				
LETTERTOGUESS:		'a'					

Question	Answer				Marks		
4(b)	<b>Label</b>	<b>Opcode</b>	<b>Operand</b>	<b>Comment</b>	<b>Mark</b>	<b>10</b>	
	START:	LDR	#0	// initialise the Index Register	1		
	LOOP:	LDX	NUMBERS	// load the value from NUMBERS	1 (LOOP) + 1 (LDX NUMBERS)		
		LSL	#2	// multiply by 4	1 (LSL) + 1 (#2)		
		STX	NUMBERS	// store the new value in NUMBERS	1		
		INC	IX	// increment the Index Register	1		
		LDD	COUNT	// increment COUNT	1		
		INC	ACC				
		STO	COUNT				
		CMP	#5	// is COUNT = 5 ?	1		
		JPN	LOOP	// repeat for next number	1		
	ENDP:	END					
	COUNT:		0				
	NUMBERS:		22				
			13				
			5				
			46				
			12				

Question	Answer	Marks
5(a)(i)	PERT / GANTT	1
5(a)(ii)	1 mark per bullet to max 3 For example: <ul style="list-style-type: none"> <li>• Calculate total minimum time required for project</li> <li>• Identify milestones</li> <li>• Task dependencies</li> <li>• Provides the critical path analysis</li> <li>• Identify which tasks need to be prioritised</li> <li>• Determine when to begin specific tasks/stages</li> <li>• Identify slack time</li> <li>• Identify when resources need allocating</li> <li>• Identify tasks that can be completed in parallel</li> </ul>	3
5(b)(i)	Integration	1
5(b)(ii)	Beta / acceptance	1

Question	Answer	Marks
6(a)	1 mark per bullet to max 6 <ul style="list-style-type: none"> <li>• Declaring a class with the name animal</li> <li>• Declaring variables for across, down and score (all Integers)</li> <li>• ...as private/protected</li> <li>• Correct constructor header and ending</li> <li>• Randomly generating an across between 0–39 inc. in constructor</li> <li>• Randomly generating a down between 0–39 inc. in constructor</li> <li>• Initialising Score to zero in constructor</li>   <li>• Correct get for <code>Across</code></li> <li>• Correct set for <code>Across</code></li> </ul>	6

Question	Answer	Marks
6(a)	<pre>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</pre>	



Question	Answer	Marks
6(a)	<p>or</p> <pre> 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 </pre>	

Question	Answer	Marks
6(a)	<pre> 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; </pre>	

Question	Answer	Marks
6(b)	<p>1 mark per bullet to max 5</p> <ul style="list-style-type: none"> <li>• constructor method heading and ending</li> <li>• Initialise all 40 by 40 elements of Grid as " or equivalent</li> <li>• Loop 5 times...</li> <li>• ...Creates a new instance of animal inside loop...</li> <li>• ...and adds it to array <code>AnimalList</code></li> </ul> <p>• Call <b>generate food and initialise</b> <code>StepCounter</code> to 0</p> <p><b>Example Python</b></p> <pre>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()</pre> <p><b>Example VB</b></p> <pre>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</pre>	<b>5</b>

Question	Answer	Marks
6(b)	<p>Example Pascal</p> <pre> 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; </pre>	
6(c)(i)	<p>1 mark per bullet:</p> <ul style="list-style-type: none"> <li>• Function header and ending taking one value as parameter</li> <li>• Check if coordinate = 0 (on lower bound)</li> <li>• ...generate random number (0 or 1)</li> <li>• Check if coordinate = 39 (on upper bound)</li> <li>• ...generate random number (-1 or 0)</li> <li>• Generate random number (e.g. -1, 0, 1)</li> <li>• Return the generated value</li> </ul>	<b>max 4</b>

Question	Answer	Marks
6(c)(i)	<p><b>Example VB</b></p> <pre>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</pre> <p><b>Example Python</b></p> <pre>def GenerateDirection(Coord) :     lowerBound = -1     upperBound = 1     if Coord == 0 :         lowerBound = 0     elif Coord == 39 :         upperBound = 0     return random.randint(lowerBound, upperBound)</pre>	

Question	Answer	Marks
6(c)(i)	<p><b>Example Pascal</b></p> <pre>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;</pre>	
6(c)(ii)	<p><b>1 mark per bullet to max 4</b></p> <ul style="list-style-type: none"> <li>• Procedure move header, no parameters</li> <li>• Calling GenerateDirection <b>twice</b> sending across and down as separate parameters</li> <li>• Add return value to Across</li> <li>• Add return value to Down</li> <li>• Check if the grid, at the (new) coordinates == "F"</li> <li>• ..if true, Call EatFood</li> </ul> <p><b>Example python</b></p> <pre>def Move(self) :   self.Across += GenerateChangeInCoordinate(self.Across)   self.Down += GenerateChangeInCoordinate(self.Down)   if grid[self.Across][self.Down] == 'F' :     self.EatFood()   return</pre>	<b>4</b>

Question	Answer	Marks
6(c)(ii)	<p><b>Example VB</b></p> <pre>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</pre> <p><b>Example Pascal</b></p> <pre>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;</pre>	
6(d)	<p>1 mark per bullet to max 3</p> <ul style="list-style-type: none"> <li>• Pre-compiled</li> <li>• Collection of Code/modules/routines</li> <li>• Each module performs a specific purpose/task</li> <li>• Each module can be linked/imported into the program</li> </ul>	<b>2</b>