Mark Scheme (Results)

November 2020

Pearson Edexcel International GCSE

In Computer Science (4CP0/2A)

Paper 2: Application of Computational Thinking

Edexcel and BTEC Qualifications

Edexcel and BTEC qualifications are awarded by Pearson, the UK's largest awarding body. We provide a wide range of qualifications including academic, vocational, occupational and specific programmes for employers. For further information visit our qualifications websites at www.edexcel.com or www.btec.co.uk. Alternatively, you can get in touch with us using the details on our contact us page at www.edexcel.com/contactus.

Pearson: helping people progress, everywhere

Pearson aspires to be the world's leading learning company. Our aim is to help everyone progress in their lives through education. We believe in every kind of learning, for all kinds of people, wherever they are in the world. We've been involved in education for over 150 years, and by working across 70 countries, in 100 languages, we have built an international reputation for our commitment to high standards and raising achievement through innovation in education. Find out more about how we can help you and your students at: www.pearson.com/uk

Autumn 2020
Publications Code 4CP0_2A_2011_MS

General Marking Guidance

- All candidates must receive the same treatment. Examiners must mark the first candidate in exactly the same way as they mark the last.
- Mark schemes should be applied positively. Candidates must be rewarded for what they have shown they can do rather than penalised for omissions.
- Examiners should mark according to the mark scheme not according to their perception of where the grade boundaries may lie.
- There is no ceiling on achievement. All marks on the mark scheme should be used appropriately.
- All the marks on the mark scheme are designed to be awarded. Examiners should always award full marks if deserved, i.e. if the answer matches the mark scheme. Examiners should also be prepared to award zero marks if the candidate's response is not worthy of credit according to the mark scheme.
- Where some judgement is required, mark schemes will provide the principles by which marks will be awarded and exemplification may be limited.
- When examiners are in doubt regarding the application of the mark scheme to a candidate's response, the team leader must be consulted.
- Crossed out work should be marked UNLESS the candidate has replaced it with an alternative response.

Mark Scheme - Theory

Question	mp	Answer	Additional Guidance	Mark
1 (a)	A1	**C (**equals) (1)		(1)

Question	mp	Answer	Additional Guidance	Mark
1 (b)	B1	Award one mark for any of: • TRUE **and** FALSE (1) • Yes **and** No (1) • T **and** F (1) • 0 **and** 1 (1)	Ignore capitalisation	(1)

Question	mp	Answer	Additional Guidance	Mark
1 (d)	D1	Award one mark for any of: • (Symbolic name associated with) a value that may be changed (1) • A container used to store data (1) • A data store whose contents can change whilst a program is executing (1)		(1)

Question	mp	Answer	Additional Guidance	Mark
1 (e) (i)	E1	Python 6 C# 12 Java 8		(1)
1 (e) (ii)	E2	Python 3 C# 7 Java 3		(1)
1 (e) (iii)	E3	Award one mark for any of: • amount (1) • x (1) • total (1) • count (1)	Accept listNumbers or numArray	(1)
1 (e) (iv)	E4	message		(1)

Question	mp	Answer	Additional Guidance	Mark
3 (a)	A1	Award one mark for any of: • Conversion of plain text into cipher (1) • Converting information/data into a code (1) • Converting information/data into an unreadable format (1)	Accept alternative wording	(1)

Question	mp	Answer	Additional Guidance	Mark
3 (b)	B1	Award one mark for any of: • To ensure that the data can only be read by an authorised person / can't be read by an unauthorised person. (1)	Accept alternative similar wording. Do not accept hacking.	(1)

Question	mp	Answer	Additional Guidance	Mark
3 (c) (i)	C1 C2 C3 C4	Award 4 marks for a correct response: • CAL|OTTA|MUIN|PO • PO|MUIN|OTTA|CAL Award 1 mark each up to a maximum of 3 for: • 4 lines of text (1) • Zigzag arrangement of letters (1) • Letters from each line reproduced / consistent use of the key 4: GSQTYXEXMSREP/ YKILQPWPEKJWH (1)		(4)

Question	mp	Answer	Additional Guidance	Mark
3 (c) (ii)	C5 C6	Award 2 marks for a linked explanation such as: • Limited number of usable keys (1) to allow for sufficient movement of characters / so can easily be decoded by trial		(2)

		and error / easy to use brute force to crack (1)		

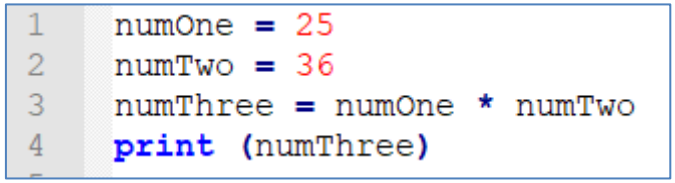
<table>
<tr><th>Question</th><th>mp</th><th>ref</th><th>Answer</th><th>Additional Guidance</th><th>Mark</th></tr>
<tr><td>5 (a)</td><td>A1
A2
A3
A4</td><td>2.1.4</td><td>Award 1 mark for each appropriate validation test (up to 2) and 1 mark for suitable example of erroneous data that matches the test e.g.

<table><tr><td>First 3 characters are upper case letters</td><td>Fff123456o</td></tr><tr><td>Characters 4 to 9 are non-zero numbers</td><td>FFF101111o</td></tr><tr><td>Consists of only 10 characters</td><td>FFF111111ee</td></tr><tr><td>Final character correct for sum of numbers</td><td>FFF112233o</td></tr></table></td><td></td><td>(4)</td></tr>
</table>

Question	mp	ref	Answer	Additional Guidance	Mark
5 (b) (i)	B1 B2	1.1.9	Award 1 mark each for any of: • Simple implementation (1) • Can be used for sorted or unsorted lists (1) • If the target is at the beginning of the data structure the search will be faster than a binary search	Accept disadvantages of a binary compared with a linear search.	(2)
5 (b) (ii)	B3 B4	1.1.9	Award 1 mark each for any of: • May need to compare with all items in list before search complete (1) • May need longer time for searching a large list (1)	Accept advantages of a binary compared with a linear search.	(2)

Question	mp	ref	Answer	Additional Guidance	Mark
5 (c) (i)	C1 C2	2.6.2	Award 1 mark each up to a maximum of 2 for: • Both can use arguments / parameter passing (1) • Both can use local variables (1) • Both can be called from anywhere within the program (1) • Code can be reused without being rewritten (1) • Both can be library files (1) • The code can be independently tested for both (1)		(2)
5 (c) (ii)	C3 C4	2.6.2	Award two marks for a linked explanation such as: • A function must always return a result (1) whereas a procedure does not (1) • A function interface must have a data type (1) to signify the type of data that will be returned / a procedure does not need this (1) • The result of a function must always be used (either assigned to a variable or as part of a condition) (1) whereas a procedure does not explicitly return a result to be used (1) • A function produces information (1) whereas a procedure performs a task (1)		(2)

Mark Scheme – Python Coding

Question	mp	ref	Answer	Additional Guidance	Mark
1 (c)	C1	2.1.5	Delete space in variable name (1)		
	C2	2.1.5	Capitalise O in variable name (1)		
	C3	2.1.5	Correction to print statement (1)		(3)
Code example					
Python					

```
1  numOne = 25
2  numTwo = 36
3  numThree = numOne * numTwo
4  print (numThree)
```

Question	mp	ref	Answer	Additional Guidance	Mark
2 (a)	A1	2.3.4	Set variable base to 50 or heightChk to true	Logic of algorithm must be followed as set out. Alternatives must address each point. Do not penalise candidates who attempt more than the stated requirements. Don't penalise spelling mistakes and alternative wording of the output.	
	A2	2.2.2	Create WHILE DO loop		
	A3	2.4.1	Request input of height		
	A4	2.2.2	if statement checks if height >= 1 or <= 100		
	A5	2.2.2	if statement checks if height is >= 1 and <=100		
	A6	2.2.2	heightCheck set to false if condition is met		
	A7	2.5.1	Set area using given formula (0.5 x base x height)		
	A8	2.4.1	Display one of base, height and area with appropriate label or displays two of base, height and area without labels		
	A9	2.4.1	Display all of base, height and area with appropriate labels		
	A10	1.1.6	Executing and producing correct output		(10)
Code example					
Python					

```python
#   Q02aFINISHED

#   Initialise variables

base = 50

#   Print prompt and take input from user

    heightChk = TRUE
    while heightChk:
        height = int(input("Enter the height (between 1 and 100): "))
        if (height >= 1 and height <= 100):
            heightChk = FALSE

#   Calculate area and print out values

    area = 0.5 * base * height

    print ("Base length of triangle is : ", base)
    print ("Height of triangle is : ", height)
    print ("Area of triangle is : ", area)

```

Question	mp	ref	Answer	Additional Guidance	Mark
2 (b)	B1	2.2.1	Meaningful variable names used (1)	The code example shown is one way of responding to the task. Other methods should be credited accordingly.	(7)
	B2	2.4.1	Meaningful prompts for input (1)		
	B3	2.4.1	Input of length and width as whole numbers (1)		
	B4	2.5.1	Perimeter calculated (= 2 x length + 2 x width) (1)		
	B5	2.5.1	Gap deducted from perimeter (1)		
	B6	2.4.1	Meaningful output (1)		
	B7	2.1.1	Code is fit for purpose. Must have mark points 1-6 and be easy to read (1)		
Code example					
Python					

```
#   Q02bFINISHED

#   Request input

length = int(input("Enter the length: "))
width = int(input("Enter the width: "))

#   Calculate number of panels needed

panels = 2 * length + 2 * width - 4

#   Print out number of panels needed

print ("Number of panels needed: " , panels)
```

Question	mp	ref	Answer	Additional Guidance	Mark
4 (a)	A1	2.5.2	english < 40, maths < 50 (1)		
	A2	2.5.3	Use of **and** operator in IF (1)		
	A3	2.5.2	english < 40, maths < 50 (1)		
	A4	2.5.3	Use of **or** operator in 'else if 1' (1)		
	A5	2.5.2	english >= 80 AND maths >=85 (1)		
	A6	2.5.3	Correct completion of final else with messageIndex = 3 (1)		(6)
Code example					

Python			`if (english < 40 and maths < 50):`
			`    messageIndex = 0`
			`elif (english < 40 or maths < 50):`
			`    messageIndex = 1`
			`elif (english >= 80 and maths >= 85):`
			`    messageIndex = 2`
			`else:`
			`    messageIndex = 3`		

Question	mp	ref	Answer	Additional Guidance	Mark
4 (b)	B1	2.2.2	Loop checks each pupil for low attendance		
	B2	2.1.6	Display of name of low attendances		
	B3	2.3.1	Counter for high attenders initialised		
	B4	2.2.2	Loop uses correct comparison (>=)		
	B5	2.6.2	Number of high attenders displayed correctly		
	B6	2.6.3	Subprogram for option 1 called correctly		
	B7	2.6.3	Subprogram for option 2 called correctly		(7)

Code example

Python

```
3   #   write subprogram for menu option 1 here
4   def lowAttendance():
5       for x in range(len(pupilAttendance)):
6           if pupilAttendance[x-1][2] < 75 :
7               print (pupilAttendance[x-1][0], pupilAttendance[x-1][1])
8       return
9
10  #   write subprogram for menu option 2 here
11
12  def highAttendance():
13      count = 0
14      for x in range(len(pupilAttendance)):
15          if pupilAttendance[x-1][2] >= 90 :
16              count += 1
17      return count
18
```

```
Faroukh Salah
Amara Grzinski
Taz Grimstow
Sadia Bhatti
Fernado Askabat
Siyao Wang
```

```
45  if option == 1:
46      # complete if statement
47      lowAttendance()
48
49  elif option == 2:
50      # complete else if statement
51      highAtt = highAttendance()
52      print ("There were ", highAtt," high attenders")
53
```

```
There were  7  high attenders
```

For Q6, the first 11 marks are for coding that matches requirements of task. The remaining 9 marks should be allocated on a best fit.

Question	mp	ref	Answer	Additional Guidance	Mark
6	A1	2.2.1	Initialise variable for total sales by member of staff (1)	Sales for each member of staff George Taylor 28466 Fehlix Chayne 35250 Sumren Bergen 30186 Samira Beckle 37430 Nellie Bogart 20448 Cheryl Grouth 34383 Danuta Graunt 26421 Jaiden Deckle 21507 Jimran Caliks 32415 Deynar Derran 28590 ========== Total staff sales: 295096 Highest sales by: Samira Beckle with 37430 Second highest sales by: Fehlix Chayne with 35250 >>>	(11)
	A2	2.5.1	Add succeeding months' sales to total sales (1)		
	A3	2.1.6	Display total sales by member of staff (1)		
	A4	2.2.2	Loop repeats for each member of staff (1)		
	A5	2.2.1	Running team total sales initialised (1)		
	A6	2.2.2	Add succeeding individual staff sales to team total (1)		
	A7	2.1.6	Display final running total (1)		
	A8	2.1.6	Staff member with highest individual sales identified (1)		
	A9	2.1.6	Staff member with second highest individual sales identified (1)		
	A10	2.1.6	Information about staff with highest sales displayed (1)		
	A11	2.1.6	Information about staff with second highest sales displayed (1)		

Band 1 (1-3 marks)	Band 2 (4-6 marks)	Band 3 (7-9 marks)	Mark
Little attempt to decompose the problem into component parts	Some attempt to decompose the problem into component parts	The problem has been decomposed into component parts	
Some parts of the logic are clear and appropriate to the problem	Most parts of the logic are clear and mostly appropriate to the problem	The logic is clear and appropriate to the problem	
Some appropriate use and manipulation of data types, variables, data structures and program constructs	The use and manipulation of data types, variables and data structures and program constructs is mostly appropriate	The use and manipulation of data types, variables and data structures and program constructs is appropriate	
Parts of the code are clear and readable	Code is mostly clear and readable	Code is clear and readable	
Finished program will not be flexible enough with other data sets or input	Finished program will function with some but not all other data sets or input	Finished program could be used with other data sets or input	
The program meets some of the given requirements	The program meets most of the given requirements	The program fully meets the given requirements	(9)

Code example

Python

```
allStaffTotal = 0
highestSales = 0
secondSales = 0
high = 0
second = 0

print ("Sales for each member of staff","\n\n")
for staff in staffSales:
    staffTotal = 0
    sales = 3
    while (sales < 9):
        staffTotal = staffTotal + staff[sales]
        sales = sales + 1
    print (staff[1],staff[2],staffTotal)
    if (staffTotal > secondSales):
        if (staffTotal > highestSales):
            highestSales = staffTotal
            high = staff
        else:
            secondSales = staffTotal
            second = staff
    allStaffTotal = allStaffTotal + staffTotal
print ("==========================\n\nTotal sales for all staff: ", allStaffTotal)

print ("\n\nHighest sales by: ",high[1],high[2], " with ", highestSales)
print ("\n\nSecond highest sales by: ",second[1],second[2], " with ", secondSales)
```