Pearson Edexcel

Mark Scheme

Specimen Set 3

Pearson Edexcel GCSE In Computer Science (1CP2)
Paper 02: 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

Paper 2 Mark Scheme

Question number	Answer	Additional guidance	Mark
1	Award marks as shown. • Add a line to import the math library (1) Original: <Blank> Amended: `import math` • Create and set an integer variable (1) Original: <Blank> Amended: `radius = 0` • Create and set a real variable (1) Original: <Blank> Amended: `circumference = 0.0` • Complete the line to take input from the user Original: `radius =` Amended: `radius = int (input ("Enter the radius of a circle: "))` Call to input(), with a prompt (1) Call to int() to convert string to integer (1) • Complete the line to validate for negative radii (1) Original: `radius` Amended: `radius <= 0` • Add a line to print an invalid input message (1) Original: <Blank> Amended: `print ("Invalid radius")` • Complete the calculation of the circumference (1) Original: `circumference =` Amended: `circumference = 2 * math.pi * radius` • Complete the line to round circumference to three decimal places Original: `circumference =` Amended: `circumference = round (circumference, 3)` Call to round() (1) Correct parameters to round() (1)	• Bullet 4: Accept any appropriate string prompt. • Bullet 6: Accept any appropriate message. • Bullet 7: Accept any order for calculation. Accept approximate numeric value of Pi, if math library not used	**(10)**

```
# ------------------------------------------------------------------
# Import libraries
# ------------------------------------------------------------------
# =====> Import the math library
import math

# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
# =====> Create an integer variable named radius and set it to 0
radius = 0

# =====> Create a real variable named circumference and set it to 0.0
circumference = 0.0

```

```
# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------
# =====> Complete the line to assign an integer, input by
#        the user, to the variable radius
radius = int (input ("Enter the radius of a circle: "))

# =====> Complete the code in the brackets to check for
#        an invalid radius of zero or less input by the user
if (radius <= 0):
    # =====> Add a line to tell the user the entry is invalid
    print ("Invalid radius")
else:
    # =====> Complete the calculation of the circumference
    circumference = 2 * math.pi * radius

    # =====> Complete the line to round circumference to three
    #        decimal places using the round() function
    circumference = round (circumference, 3)

    print ("The circumference is", circumference)
```

Question number	Answer	Additional guidance	Mark
2	Award marks as shown. • (line 5) (1) From: `initials =` To: `initials = ""` (any string value) • (line 17) (1) From: `else if (len (initials) > 3):` To: `elif (len (initials) > 3):` • (line 19) (1) From: `else` To: `else:` • (line 11) (1) From: `while ((a == "Y") and (a == "y")):` To: `while ((a == "Y") or (a == "y")):` • (line 15) (1) From: `elif (len (initials) <= 2):` To: `elif (len (initials) < 2):` • (line 13) (1) From: `if (initials.isalpha ()):` To: `if (not initials.isalpha ()):` • (line 23) (1) From: `a == input ("Would you like to go again? ")` To: `a = input ("Would you like to go again? ")` • (line 20) (1) From: `initials = initials.lower ()` To: `initials = initials.upper ()` • across all lines, change 'a' to a suitable name such as 'answer' (1)	• Bullet 6: accept equivalent expressions • Bullet 7: the line may look different if the variable 'a' has already been changed • Bullet 10: any equivalent expression accepted	**(10)**

	• (line 21) a comment such as 'displays a string and a variable on the same line' / 'prints a string and a variable' / 'prints two strings' (1)		

```
# ---------------------------------------------------------------------
# Global variables
# ---------------------------------------------------------------------
answer = "Y"
initials = ""

# ---------------------------------------------------------------------
# Main program
# ---------------------------------------------------------------------

while ((answer == "Y") or (answer == "y")):
    initials = input ("Enter your initials, without spaces: ")
    if (not initials.isalpha ()):
        print ("Must be alphabetic characters")
    elif (len (initials) < 2):
        print ("Not long enough")
    elif (len (initials) > 3):
        print ("Too long")
    else:
        initials = initials.upper ()
        print ("Your initials are:", initials) # Displays two strings on the same line

    answer = input ("Would you like to go again? ")
```

Question number	Answer	Additional guidance	Mark

3	Award marks as shown.
	• Complete theVowels with a list of strings (1)
	`theVowels = ["A", "E", "I", "O", "U"]`
	• Complete theCounts with a list of integers (1)
	`theCounts = [0, 0, 0, 0, 0]`
	• Call to range() must use 'len (inSymbols)' (1)
	`for index in range (len (inSymbols)):`
	• First item in print statement should be the vowel/symbol using indexing (1)
	`inSymbols[index]`
	• Histogram line should be the vowel/symbol repeated using string multiplication (symbol*count) using indexing (1)
	`inSymbols[index] * inNumbers[index]`
	• Contents of data converted using <string>.upper() (1)
	`data = data.upper()`
	• Use of selection (if/elif) to count occurrences (1)
	• Use of relational operator (==) to check for vowels (1)
	• Call to displayHistogram has two arguments (1)
	`displayHistogram (theVowels, theCounts)`
	• Order of arguments to displayHistogram matches function definition (1)
	`displayHistogram (theVowels, theCounts)`
	Levels-based mark scheme to a maximum of 3, from:
	• Functionality (3)
		Considerations for levels-based mark scheme:
		• [6.1.2] Translates without error, even if reduced functionality
		• [6.1.6] Functions correctly for an empty string input (i.e. no values displayed, although axis is acceptable)
		• [6.4.1] Functions correctly for any string input, without carriage returns and line feeds
			(13)

Functionality (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	**Functionality (when the code is run)** • The component parts of the program are incorrect or incomplete, providing a program of limited functionality that meets some of the given requirements. • Program outputs are of limited accuracy and/or provide limited information. • Program responds predictably to some of the anticipated input. • Solution is not robust and may crash on anticipated or provided input.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that meets most of the stated requirements. • Program outputs are mostly accurate and informative. • Program responds predictably to most of the anticipated input. • Solution may not be robust within the constraints of the problem.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that fully meets the given requirements. • Program outputs are accurate, informative, and suitable for the user. • Program responds predictably to anticipated input. • Solution is robust within the constraints of the problem.	**3**

```
# ---------------------------------------------------------------------
# Global variables
# ---------------------------------------------------------------------
data = ""

# =====> Create a one-dimensional data structure holding the
#        five vowels in upper case
theVowels = ["A", "E", "I", "O", "U"]

# =====> Create a one-dimensional data structure to hold the count for each vowel
theCounts = [0, 0, 0, 0, 0]

# ---------------------------------------------------------------------
# Subprograms
# ---------------------------------------------------------------------
def displayHistogram (inSymbols, inNumbers):
    # =====> Complete the call to range (), using len ()
    for index in range (len (inSymbols)):
        # Repeat the symbol for the number of times required
        # =====> Complete the print statement to print the vowel
        #        for the number of times it was counted
        print (inSymbols[index] + "| " + inSymbols[index] * inNumbers[index])

```

```
# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------
# User types in a string
data = input ("Enter a string: ")

# =====> Complete the line to convert data to upper case
data = data.upper()

# Count each vowel in the input
for letter in data:
    # =====> Use selection to check for each vowel and increment the corresponding count
    if (letter == "A"):
        theCounts[0] = theCounts[0] + 1
    elif (letter == "E"):
        theCounts[1] = theCounts[1] + 1
    elif (letter == "I"):
        theCounts[2] = theCounts[2] + 1
    elif (letter == "O"):
        theCounts[3] = theCounts[3] + 1
    elif (letter == "U"):
        theCounts[4] = theCounts[4] + 1

# Print a horizontal histogram
# =====> Complete the call to the subprogram
displayHistogram (theVowels, theCounts)
```

Question number	Answer	Additional guidance	Mark

4	Award marks as shown.
	• White space used to aid readability (1)
	• User input accepted as first operation, after initialisations (1)
	• Modulus calculated before checkDigit calculation (1)
	• Repetition (while index < len (isbn)) around product calculation (1)
	• product calculated before incrementing index (1)
	• product calculated before decrementing multiplier (1)
	• total set after product calculated (1)
	• Initialisation of variables before calculations:
	  ○ multiplier (1)
	  ○ index (1)
	  ○ total (1)
	• Correct outputs for each set of test data:
	  ○ 071954400 – 9 (1)
	  ○ 047119047 - 0 (1)
	  ○ 061826941 – X (1)
	• One or more comments match blocks of code (1)
	• Printing of result is the last line in the program (1)
			(15)

```
# ----------------------------------------------------------------
# Global variables
# ----------------------------------------------------------------
isbn = ""
index = 0
product = 0
total = 0
multiplier = 10
modulus = 0
checkDigit = 0
strCheckDigit = ""

# ----------------------------------------------------------------
# Main program
# ----------------------------------------------------------------
# Get the user's input
isbn = input ("Enter a 9-digit ISBN number: ")

# Multiply each digit by its position in the number
while (index < len (isbn)):
    product = int (isbn[index]) * multiplier
    total = product + total
    index = index + 1
    multiplier = multiplier - 1

```

```
# Find the remainder from integer division by 11
modulus = total % 11

# Calculate the check digit
checkDigit = 11 - modulus

# Test for boundaries
if (checkDigit == 11):
    strCheckDigit = "0"
elif (checkDigit == 10):
    strCheckDigit = "X"
else:
    strCheckDigit = str (checkDigit)

print ("Check digit = " + strCheckDigit + " ISBN = " + isbn + strCheckDigit)
```

Question number	Answer	Additional guidance	Mark
5	Award marks as shown. • Calculate the mean of each row (1) • Strip carriage return (1) • Split line by comma (1) • File opened for reading only (1) • File closed before exiting program (1) • No global variables created (1) Levels-based mark scheme to a maximum of 6, from: • Solution design (3) • Functionality (3)	Considerations for levels-based mark scheme: • [6.1.2] Translates without error, even if reduced functionality • [6.4.1] Output is suitable for the audience and fit for purpose (aligned columns, labels) • [6.1.6] Means match numbers in file (33.90, 55.30, 41.50, 48.10, 54.50) • [6.2.2] Use of 'for' loop in preference to a 'while' loop for iteration through numbers after split • [6.3.2] Use of a variable to track the row being read from the file • [6.3.3] Use of <string>.format() to control decimals and column alignment	**(12)**

Solution design (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	• There has been little attempt to decompose the problem. • Some of the component parts of the problem can be seen in the solution, although this will not be complete. • Some parts of the logic are clear and appropriate to the problem. • The use of variables and data structures, appropriate to the problem, is limited. • The choice of programming constructs, appropriate to the problem, is limited.	• There has been some attempt to decompose the problem. • Most of the component parts of the problem can be seen in the solution. • Most parts of the logic are clear and appropriate to the problem. • The use of variables and data structures is mostly appropriate. • The choice of programming constructs is mostly appropriate to the problem.	• The problem has been decomposed clearly into component parts. • The component parts of the problem can be seen clearly in the solution. • The logic is clear and appropriate to the problem. • The choice of variables and data structures is appropriate to the problem. • The choice of programming constructs is accurate and appropriate to the problem.	**3**

Functionality (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	**Functionality (when the code is run)** • The component parts of the program are incorrect or incomplete, providing a program of limited functionality that meets some of the given requirements. • Program outputs are of limited accuracy and/or provide limited information. • Program responds predictably to some of the anticipated input. • Solution is not robust and may crash on anticipated or provided input.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that meets most of the stated requirements. • Program outputs are mostly accurate and informative. • Program responds predictably to most of the anticipated input. • Solution may not be robust within the constraints of the problem.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that fully meets the given requirements. • Program outputs are accurate, informative, and suitable for the user. • Program responds predictably to anticipated input. • Solution is robust within the constraints of the problem.	**3**

```
# ---------------------------------------------------------------------
# Constants
# ---------------------------------------------------------------------
FILE_NAME = "Q05_Data.txt"      # The input data file
NUMS_PER_LINE = 10              # Number of items per line in the file

# ---------------------------------------------------------------------
# Subprograms
# ---------------------------------------------------------------------
def processLines (inFile):
    # =====> Write your code here
    row = 0
    total = 0
    mean = 0.0
    items = []
    layout = "{:>4}    {:^5.2f}"

    theFile = open (inFile, "r")

    # Get the line of numbers from the file
    # =====> Write your code here
    for line in theFile:            # Process whole file
        line = line.strip ()        # Strip off the LF
        items = line.split (",")    # Split on the commas

```

```
        # Calculate the mean for the items and adjust the row counter
        # =====> Write your code here
        total = 0
        for number in items:
            total = total + int (number)
        mean = total / NUMS_PER_LINE
        row = row + 1

        # Display the information in columnar format
        # =====> Write your code here
        print (layout.format (row, mean))

    # =====> Write your code here
    # Close the file
    theFile.close ()

```

```
42  def displayTableHeaders ():
43      # =====> Write your code here
44      layout = "{:>4}    {:^7}"
45      print (layout.format ("Row", "Mean"))
46      print ("-" * 15)
47
48  # ---------------------------------------------------------------------
49  # Main program
50  # ---------------------------------------------------------------------
51  # Do the processing and the display
52  displayTableHeaders ()
53  processLines (FILE_NAME)
```

Question number	Answer	Additional guidance	Mark
6	Award marks as shown. Points-based mark scheme: • Use of two-dimensional indexing to get single letters for constructing code (1) • Conversion of date (integer) to string for constructing code (1) • Use of string concatenation to construct code (1) • Appending the new label to the data structure (1) • Initial value of date set to appropriate value (0 or negative) for use with relational operator (1) • A method for tracking the youngest artist (index, whole record) (1) Levels-based mark scheme to a maximum of 9, from: • Solution design (3) • Good programming practices (3) • Functionality (3)	Considerations for levels-based mark scheme: • [6.1.1] Use decomposition to solve problem and create solution • [6.2.2] Use of 'for' loop to iterate over a data structure, rather than a 'while' loop • [6.3.1] Conversion of data types to those required by program, e.g. strings • [6.1.2] Write in a high-level language • [6.1.4] Main program code is laid out in clear sections; white space is used to show different parts of the solution/functionality; variable names are meaningful; comments are provided and are helpful • [6.2.2] Use of iteration ('for'), to find youngest artist, as list is not sorted by date • [6.4.1] Printed outputs match requirements • [6.1.6] Functions correctly for any number of artists in the list	**(15)**

Solution design (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	• There has been little attempt to decompose the problem. • Some of the component parts of the problem can be seen in the solution, although this will not be complete. • Some parts of the logic are clear and appropriate to the problem. • The use of variables and data structures, appropriate to the problem, is limited. • The choice of programming constructs, appropriate to the problem, is limited.	• There has been some attempt to decompose the problem. • Most of the component parts of the problem can be seen in the solution. • Most parts of the logic are clear and appropriate to the problem. • The use of variables and data structures is mostly appropriate. • The choice of programming constructs is mostly appropriate to the problem.	• The problem has been decomposed clearly into component parts. • The component parts of the problem can be seen clearly in the solution. • The logic is clear and appropriate to the problem. • The choice of variables and data structures is appropriate to the problem. • The choice of programming constructs is accurate and appropriate to the problem.	**3**

Good programming practices (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	• There has been little attempt to lay out the code into identifiable sections to aid readability. • Some use of meaningful variable names. • Limited or excessive commenting. • Parts of the code are clear, with limited use of appropriate spacing and indentation.	• There has been some attempt to lay out the code to aid readability, although sections may still be mixed. • Uses mostly meaningful variable names. • Some use of appropriate commenting, although may be excessive. • Code is mostly clear, with some use of appropriate white space to aid readability.	• Layout of code is effective in separating sections, e.g. putting all variables together, putting all subprograms together as appropriate. • Meaningful variable names and subprogram interfaces are used where appropriate. • Effective commenting is used to explain logic of code blocks. • Code is clear, with good use of white space to aid readability.	**3**

Functionality (levels-based mark scheme)

0	1	2	3	Max.
No rewardable material	**Functionality (when the code is run)** • The component parts of the program are incorrect or incomplete, providing a program of limited functionality that meets some of the given requirements. • Program outputs are of limited accuracy and/or provide limited information. • Program responds predictably to some of the anticipated input. • Solution is not robust and may crash on anticipated or provided input.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that meets most of the stated requirements. • Program outputs are mostly accurate and informative. • Program responds predictably to most of the anticipated input. • Solution may not be robust within the constraints of the problem.	**Functionality (when the code is run)** • The component parts of the program are complete, providing a functional program that fully meets the given requirements. • Program outputs are accurate, informative, and suitable for the user. • Program responds predictably to anticipated input. • Solution is robust within the constraints of the problem.	**3**

```
# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
theArtists = [["Andy", "Warhol", 1928],
              ["Pablo", "Picasso", 1881],
              ["Salvador", "Dali", 1904],
              ["Lavinia", "Fontana", 1552],
              ["Jackson", "Pollock", 1912],
              ["Henri", "Matisse", 1869],
              ["Frida", "Kahlo", 1907],
              ["Georgia", "O'Keeffe", 1887],
              ["Kara", "Walker", 1969],
              ["Yayoi", "Kusama", 1929]]

theLabels = []   # Put the new user labels into this structure
# ====> Write your code here
maxDate = 0
newLabel = ""
maxPerson = []

```

```
# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------
# ====> Write your code here
# Make the artists' labels
for person in theArtists:
    newLabel = person[1][0] + person[0][0] + str (person[2])
    theLabels.append (newLabel)
    print (newLabel)

# Find and print the youngest person and their birthdate
for person in theArtists:
    if person[2] > maxDate:
        maxDate = person[2]
        maxPerson = person      # Save the whole record
print (maxPerson[0], maxPerson[1], "is youngest", str (maxPerson[2]))
```