Pearson Edexcel

Mark Scheme (Results)

Summer 2025

Pearson Edexcel GCSE

In Computer Science (1CP2/02)

Paper 02: Application of Computational Thinking

Summer 2025

Question Paper Log Number 78600A

Publications Code 1CP2_02_2506_MS

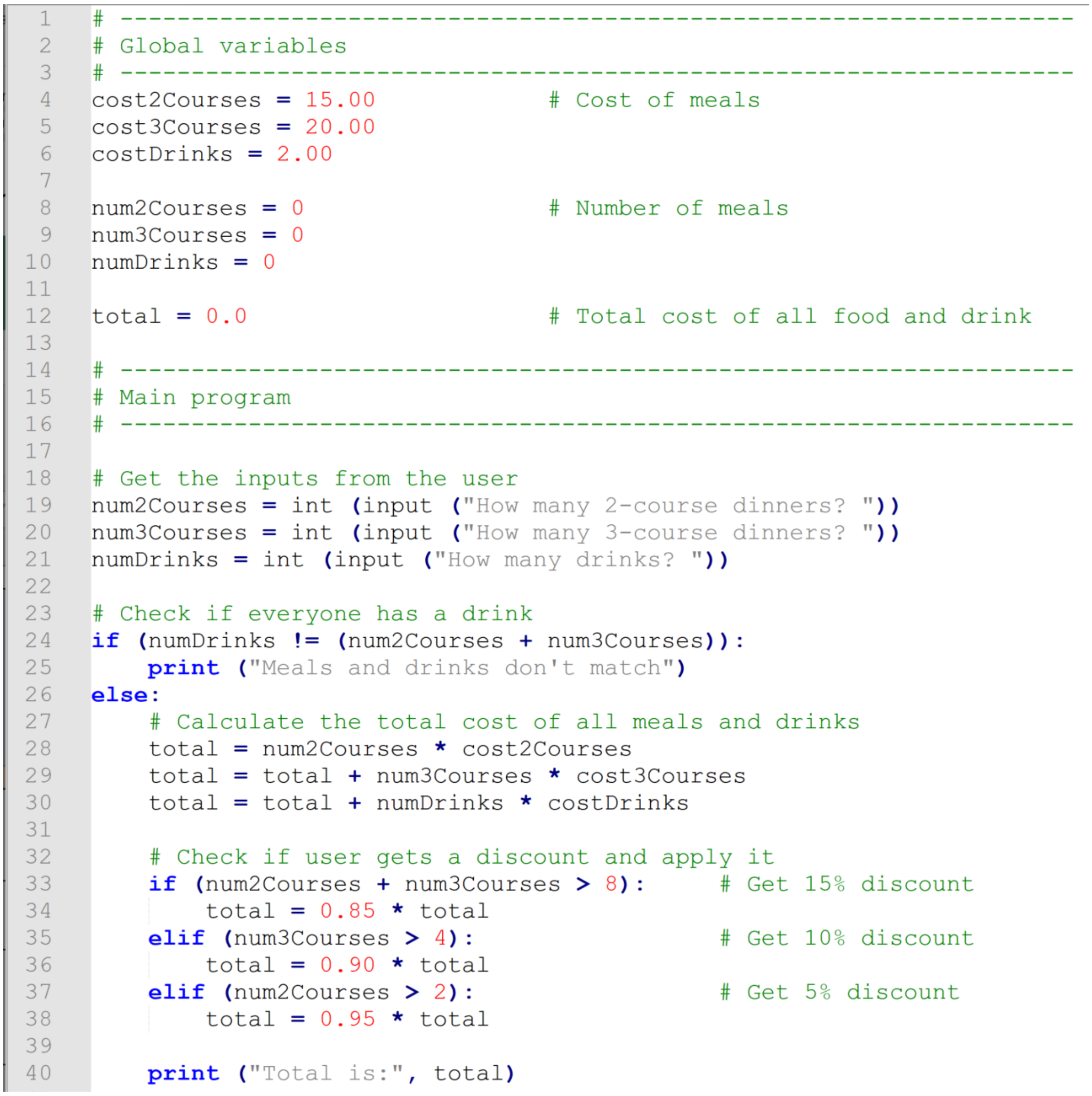
General marking guidance

- 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.

Question number	MP	Appx . Line	Answer	Additional guidance	Mark
1			Award marks as shown.	• Ignore precision on decimal places	
	1.1	4	Add comment symbol in front of 'Cost of meals'	• Do not award where the entire comment is deleted • `# Cost of meals`	
	1.2	6	Remove space between 'cost' and 'Drinks' to create a correct variable name	• `costDrinks = 2.00` • Do not award removing cost or Drinks • Allow any amendments that result in a syntactically correct identifier that is in context	
	1.3	10	Turn the assignment around the right way. Instruction cannot start with a number.	• `numDrinks = 0`	
	1.4	28	Remove double quotes from around "num2Courses"	• `total = num2Courses * cost2Courses`	
	1.5	29	Correct typo from 'cost33Courses' to 'cost3Courses'	• `total = total + num3Courses * cost3Courses`	
	1.6	33	Replace subtraction with addition	• `if (num2Courses + num3Courses > 8):`	
	1.7	34	Replace addition with multiplication	• `total = 0.85 * total`	
	1.8	38	Correct spelling from 'tatol' to 'total'	• `total = 0.95 * total`	
	1.9	40	Indent the print statement under the 'else'		**(9)**

Test data:

Two-course dinners	Three-course dinners	Drinks	Discount	Output	Additional guidance
2	2	5		Meals and drinks don't match	
5	6	11	15%	184.45	Ignore formatting of decimal values
1	5	6	10%	114.3	
5	0	5	5%	80.75	
1	2	3	No discount	61.0	



```python
# -------------------------------------------------------------------
# Global variables
# -------------------------------------------------------------------
cost2Courses = 15.00              # Cost of meals
cost3Courses = 20.00
costDrinks = 2.00

num2Courses = 0                   # Number of meals
num3Courses = 0
numDrinks = 0

total = 0.0                       # Total cost of all food and drink

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

# Get the inputs from the user
num2Courses = int (input ("How many 2-course dinners? "))
num3Courses = int (input ("How many 3-course dinners? "))
numDrinks = int (input ("How many drinks? "))

# Check if everyone has a drink
if (numDrinks != (num2Courses + num3Courses)):
    print ("Meals and drinks don't match")
else:
    # Calculate the total cost of all meals and drinks
    total = num2Courses * cost2Courses
    total = total + num3Courses * cost3Courses
    total = total + numDrinks * costDrinks

    # Check if user gets a discount and apply it
    if (num2Courses + num3Courses > 8):      # Get 15% discount
        total = 0.85 * total
    elif (num3Courses > 4):                  # Get 10% discount
        total = 0.90 * total
    elif (num2Courses > 2):                  # Get 5% discount
        total = 0.95 * total

    print ("Total is:", total)
```

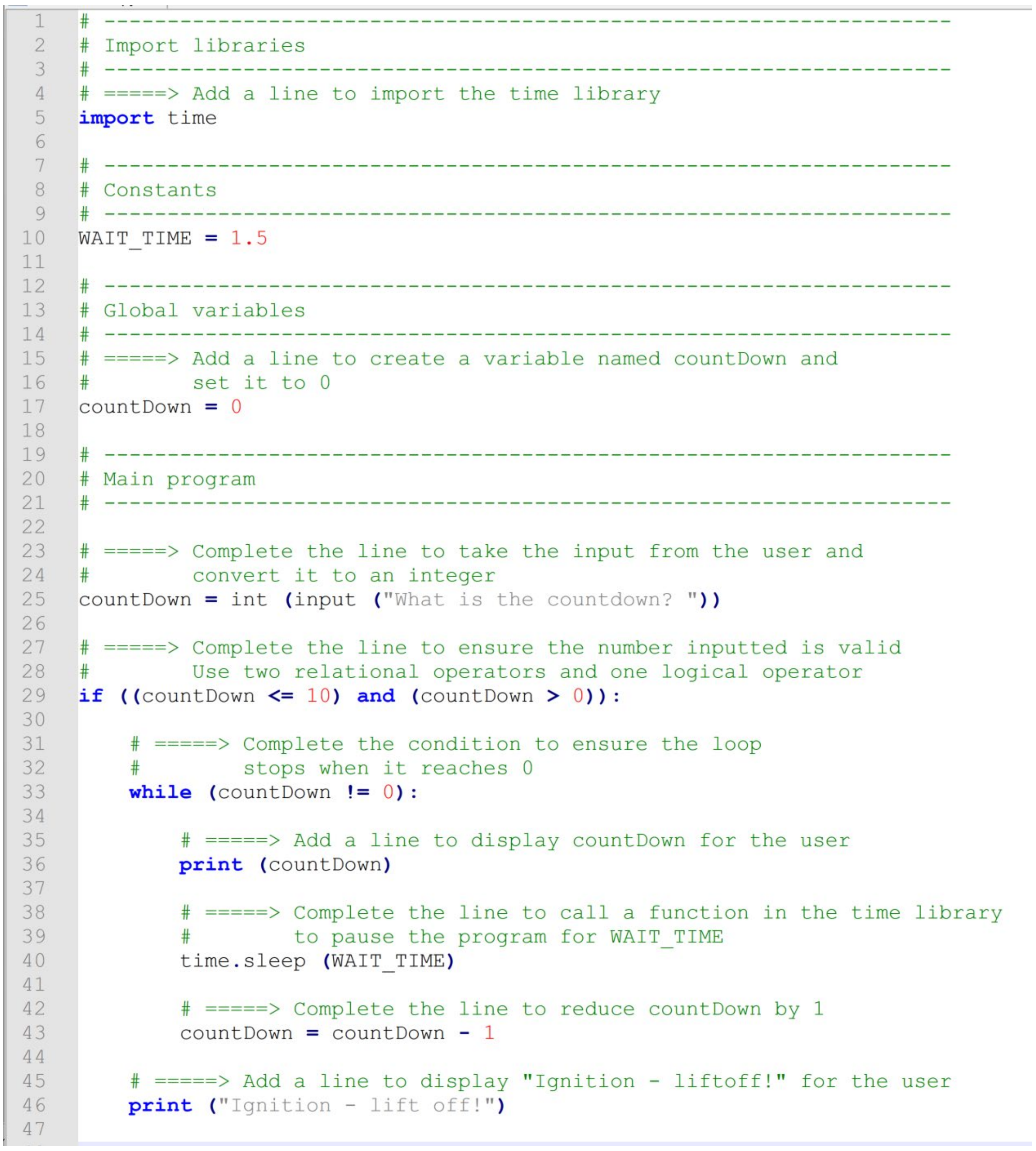
Question number	MP	Appx. Line	Answer	Additional guidance	Mark
2			Award marks as shown.	• Ignore indentation	
	2.1	5	Import time library	`import time`	
	2.2	17	Create countDown and set to 0	`countDown = 0`	
	2.3	25	Take input from user **AND** convert to integer	`countDown = int (input ("What is the countdown? "))`	

	2.4	29	Complete two relational tests to ensure number inputted is valid	`(countDown <= 10)`
				`(countDown > 0)`
				
				• Ignore logical operator
				• Allow equivalent expressions, where given lines are changed
				• Allow < 10	

	2.5	29	Complete line with the correct logical operator	`((<test>) and (<test>))` • Ignore tests • Allow equivalent expressions, where given lines are changed • Do not award OR	

	2.6	33	Complete while loop	`while (countDown != 0):`
				`while (countDown > 0):`
				Allow >= 0
					(10)

	2.7	36	Display countDown	`print (countDown)`	
	2.8	40	Call to sleep function with accurate time	`time.sleep (WAIT_TIME)` ● Allow `time.sleep (1.5)`	
	2.9	43	Reduce countDown - 1	● Allow equivalent expressions `countDown = countDown - 1`	
	2.10	46	Display 'lift off' inline with/outside while loop and inside if statement	`print ("Ignition – lift off!")` ● Allow any sensible message	



```python
# -------------------------------------------------------------------
# Import libraries
# -------------------------------------------------------------------
# =====> Add a line to import the time library
import time

# -------------------------------------------------------------------
# Constants
# -------------------------------------------------------------------
WAIT_TIME = 1.5

# -------------------------------------------------------------------
# Global variables
# -------------------------------------------------------------------
# =====> Add a line to create a variable named countDown and
#        set it to 0
countDown = 0

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

# =====> Complete the line to take the input from the user and
#        convert it to an integer
countDown = int (input ("What is the countdown? "))

# =====> Complete the line to ensure the number inputted is valid
#        Use two relational operators and one logical operator
if ((countDown <= 10) and (countDown > 0)):

    # =====> Complete the condition to ensure the loop
    #        stops when it reaches 0
    while (countDown != 0):

        # =====> Add a line to display countDown for the user
        print (countDown)

        # =====> Complete the line to call a function in the time library
        #        to pause the program for WAIT_TIME
        time.sleep (WAIT_TIME)

        # =====> Complete the line to reduce countDown by 1
        countDown = countDown - 1

    # =====> Add a line to display "Ignition - liftoff!" for the user
    print ("Ignition - lift off!")

```

Questio n number	MP	Appx . Line	Answer	Additional guidance	Mark
3			Award marks as shown.	• Award sequence only. • Ignore intervening lines. • Ignore changes made to provided lines. • Do not award same sequence found in supplied file	
			Constants and variables		
	3.1	1-15	Four constants and three variables moved before the first executable line of the main program (1)	• ZERO, THREE, FOUR, FIVE • inString, total, asciiValue • Placement does not have to be in the labelled	
			Input, outside loop, and length validation	• If no lines are moved, not changing the sequence, then award no marks in this subsection	
	3.2	21	Input taken as first line in the main program (1)		
	3.3	23	While loop comes after taking of an input call (1)	• Ignore intervening lines	
	3.4	24	If statement validating length of input string must come after taking of input (1)	• Ignore intervening lines	
			Processing and total		
	3.5	25	Resetting of total variable to 0 is done before 'for character in inString:' loop (1)	• Allow resetting outside for character in inString • Ignore intervening lines	**(15)**

				• Allow resetting after print (total, inString), as value will be 0 on first pass of while loop anyway	
	3.6	26	'For character in inString' loop must be inside while loop (1)	• Ignore intervening lines except a line that breaks scope	
	3.7	27	Conversion of character to ASCII must be inside for loop (1)	• Ignore intervening lines	
	3.8	28	Calculating new total must be done inside for loop **AND** after conversion to ASCII (1)	• Ignore intervening lines	
			Checking divisibility		
	3.9	31, 33, 35	If/elif/else key words for checking divisibility must appear in sequence (1)	• Ignore intervening lines • Check indentation matches to make sure the three keywords are in the correct sequence. Don't allow parts of the if/else that align with the length check.	
	3.10	32, 34, 36	Concatenation (all three) of digit at the end must be inside the test in the matching selection statement (1)	• Constants can be used to see this (FOUR, FIVE, ZERO) • Do not award where code has been changed to use hard-coded numeric values	
			Outputs and going around again		
	3.11	38	Printing output must follow all three concatenation instructions (1)	• Ignore intervening lines • Ignore accuracy of MP3.10	
	3.12	39, 40	The length check (if) followed by the (else, print) pattern (1)	• Ignore intervening lines `if (len(inString)>=THREE):`	

				.
				.
				.
				`else:`
				`   print ("String … long")`	

	3.13	42	User input accepted is last line in the code (1)		
			Additional		
	3.14		Indentation doesn’t generate a syntax error (1)	● Ignore accuracy of line placement and functionality	
	3.15		Functions for any sequence of characters (1)	● Execute with test data given in question paper	

Test data from question paper:

Input	Output	Notes
X	String must be three or more characters long	
XY	String must be three or more characters long	
cup	328 cup4	
snow	455 snow5	
Ian	280 Ian4	That's an uppercase 'I' as in the name 'Ian' not a 'local area network'.
HAT	221 HAT0	
<empty>	No output, but exits program	

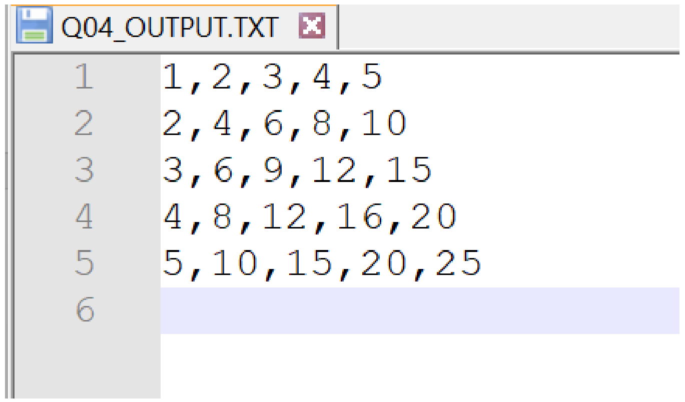
```python
# ------------------------------------------------------------------
# Constants
# ------------------------------------------------------------------
ZERO = 0
THREE = 3
FOUR = 4
FIVE = 5

# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
inString = ""
total = 0
asciiValue = 0

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

# Set up the input loop
inString = input ("Enter a string (blank to exit): ")

while (inString != ""):
    if (len (inString) >= THREE):
        total = 0                           # Reset each pass
        for character in inString:
            asciiValue = ord(character)
            total = total + asciiValue

        # Check if divisible by 4, 5, or neither
        if (total % FOUR == 0):
            inString = inString + str (FOUR)
        elif (total % FIVE == 0):
            inString = inString + str (FIVE)
        else:
            inString = inString + str (ZERO)

        print (total, inString)
    else:
        print ("String must be three or more characters long")

    inString = input ("Enter a string (blank to exit): ")
```

Question number	MP	Appx. Line	Answer	Additional guidance	Mark
4			Award marks as shown.		
			File and input		
	4.1	22	Output file opened for writing only (the 'w' must be present)	• Any method • Allow accurate hard-coded name matching the constant provided • Allow with open () • Where using open () - check for matching close ()	
			Building output line		
	4.2	30	Value is calculated as row * column	• Does not require assignment to a variable	
	4.3	31	New value is appended to outString	• Requires a type conversion • Ignore manipulation of comma for this mark • Allow any variable name for outString `outString = outString + str (num)`	
	4.4	35	Comma added as separator	• Ignore malformed output strings, but there must be the concept of a string being built • Allow hard-coded value instead of constant	
	4.5	38	Line feed added to last column in output line only	• Ignore comma, if present	
			Writing to file		
	4.6	41	Output line is added to file	• Do not award when the file.write() is over-indented to cause the writing of individual digits to the file. Full functionality marks are still available.	
			Closing file		**(11)**

	4.7	44	Output file is closed outside the ‘for row’ loop	• Allow 'with open' where processing is indented properly	
			Overall		
	4.8	Any	Accurate use of at least one of the constants provided	• OUT_FILE, COMMA, LF	
			Levels-based mark scheme to a maximum of 3, from:	Considerations for levels-based mark scheme:	
	4.9 4.10 4.11		Functionality (3)	• [6.1.5] No syntax or runtime errors in a reasonable attempt to solve the problem • [6.1.5] Some logic errors in a reasonable attempt to address all the requirements • [6.1.6] Fully meets requirements of a solution • [6.1.2] Write in a high-level language	

Output file

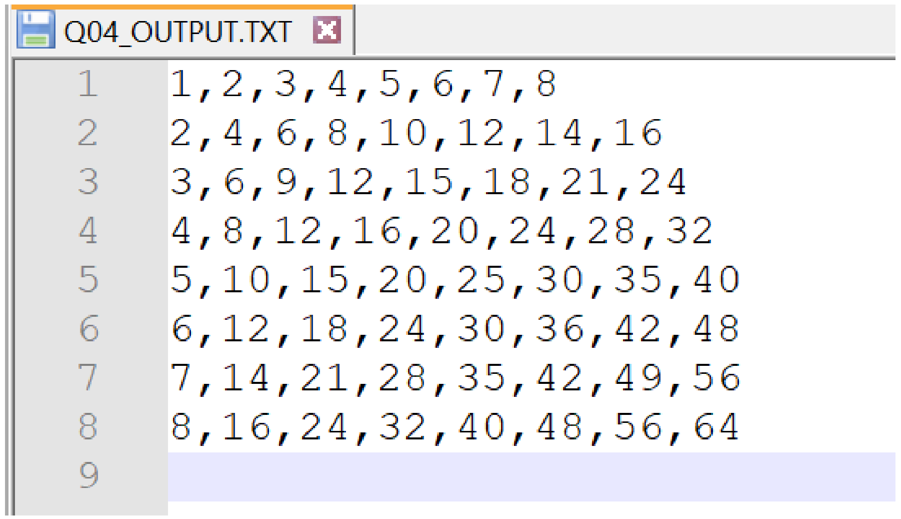
User enters the number 5:



Q04_OUTPUT.TXT

```
1,2,3,4,5
2,4,6,8,10
3,6,9,12,15
4,8,12,16,20
5,10,15,20,25

```

User enters the number 8:



Q04_OUTPUT.TXT

```
1,2,3,4,5,6,7,8
2,4,6,8,10,12,14,16
3,6,9,12,15,18,21,24
4,8,12,16,20,24,28,32
5,10,15,20,25,30,35,40
6,12,18,24,30,36,42,48
7,14,21,28,35,42,49,56
8,16,24,32,40,48,56,64

```

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**

```python
# -------------------------------------------------------------------
# Constants
# -------------------------------------------------------------------
OUT_FILE = "Q04_OUTPUT.TXT"
COMMA = ","
LF = "\n"

# -------------------------------------------------------------------
# Global variables
# -------------------------------------------------------------------
outString = ""
maxNum = 0
num = 0
row = 0
column = 0

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

# =====> Open the output file for writing
theFile = open (OUT_FILE, "w")

maxNum = int (input ("Enter a number: "))    # Get the number
for row in range (1, maxNum + 1):            # Going down the table
    outString = ""

    for column in range (1, maxNum + 1):     # Going across the table
        # =====> Calculate the new value
        num = row * column
        outString = outString + str (num)

        # =====> Add a comma to all except last column
        # =====> Add a line feed to the last column
        if (column < maxNum):
            outString = outString + COMMA
        else:
            outString = outString + LF

    # =====> Write the row to the file
    theFile.write (outString)

# =====> Close the file
theFile.close ()
```

Question number	MP	Appx. Line	Answer	Additional guidance	Mark
5			Award marks as shown.		
			Libraries		
	5.1		Import turtle library Import random library		
			Prepare the turtle		
	5.2		Set turtle speed to fastest	• Award default or new turtle `turtle.speed (0)` `turtle.speed ("fastest")`	
			Draw lines		
	5.3		A – Horizontal appx. (-200, 0) to (400, 0) B – Vertical appx. (0, 200) to (0, 400) Draw line A and line B (horizontal, vertical) in the right place	• Ignore length, colour, width, direction (left to right / right to left) of drawing and labelling of lines • Allow any method	
	5.4		C – Lower left to upper right appx. (-200, -200) for 45 degrees. D – Lower right to upper left appx. (200, -200) for 135 degrees (90+45). Draw Line C and line D (diagonals) in the right place	• Ignore length, colour, width, direction (left to right / right to left) of drawing and labelling of lines • Allow any method	
			Loop for nine		**(15)**

	5.5		Loop to draw nine circles	• Allow any method
				`for count in range (9):`
				`while index in range (0,9):`	

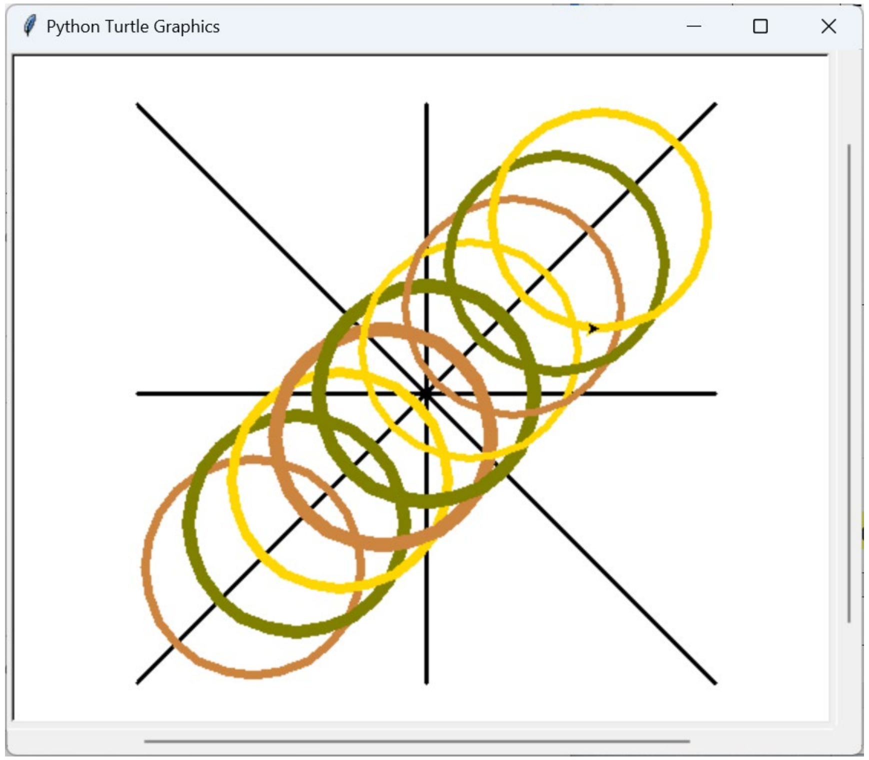
			Circle properties		

	5.6		Circle colours follow pattern of colours in the original myColours array	• Index for colours must be generated programmatically, not hard-coded
				• Allow any method
				`turtle.pencolor (myColours[count % 3])`
				
				Note, for this method, index must be initialised outside the loop and incremented inside the loop
				`index = 0`
				`turtle.pencolor (myColours[index])`
				`index += 1`	

	5.7		Call to generate random number (5 – 10)	• Ignore setting pen size `turtle.pensize (random.randint (5, 10))`	
	5.8		Circle radii are 75 units	`turtle.circle (75)`	
			Move to next position		
	5.9		Two position variables are incremented by 30 for each circle	• Arithmetic is given in the question paper. Award this mark for accurate translating of the formula and assignment to a position variable • Ignore functionality of loop `xPos = xPos + 30`	

				`yPos = yPos + 30`	
			Levels-based mark scheme to a maximum of 6, from:	Considerations for levels-based mark scheme:	
	5.10 5.11 5.12		Solution design (3)	• [6.5.1] Modulus used for finding colour • [6.3.2] Name and type of variables is appropriate to the problem • [6.1.4] Layout and comments aid understanding of logic	
	5.13 5.14 5.15		Functionality (3)	• [6.1.5] No syntax or runtime errors in a reasonable attempt to solve the problem • [6.1.5] Some logic errors in a reasonable attempt to address all the requirements • [6.1.6] Fully meets requirements of a solution • [6.1.2] Write in a high-level language	

Output



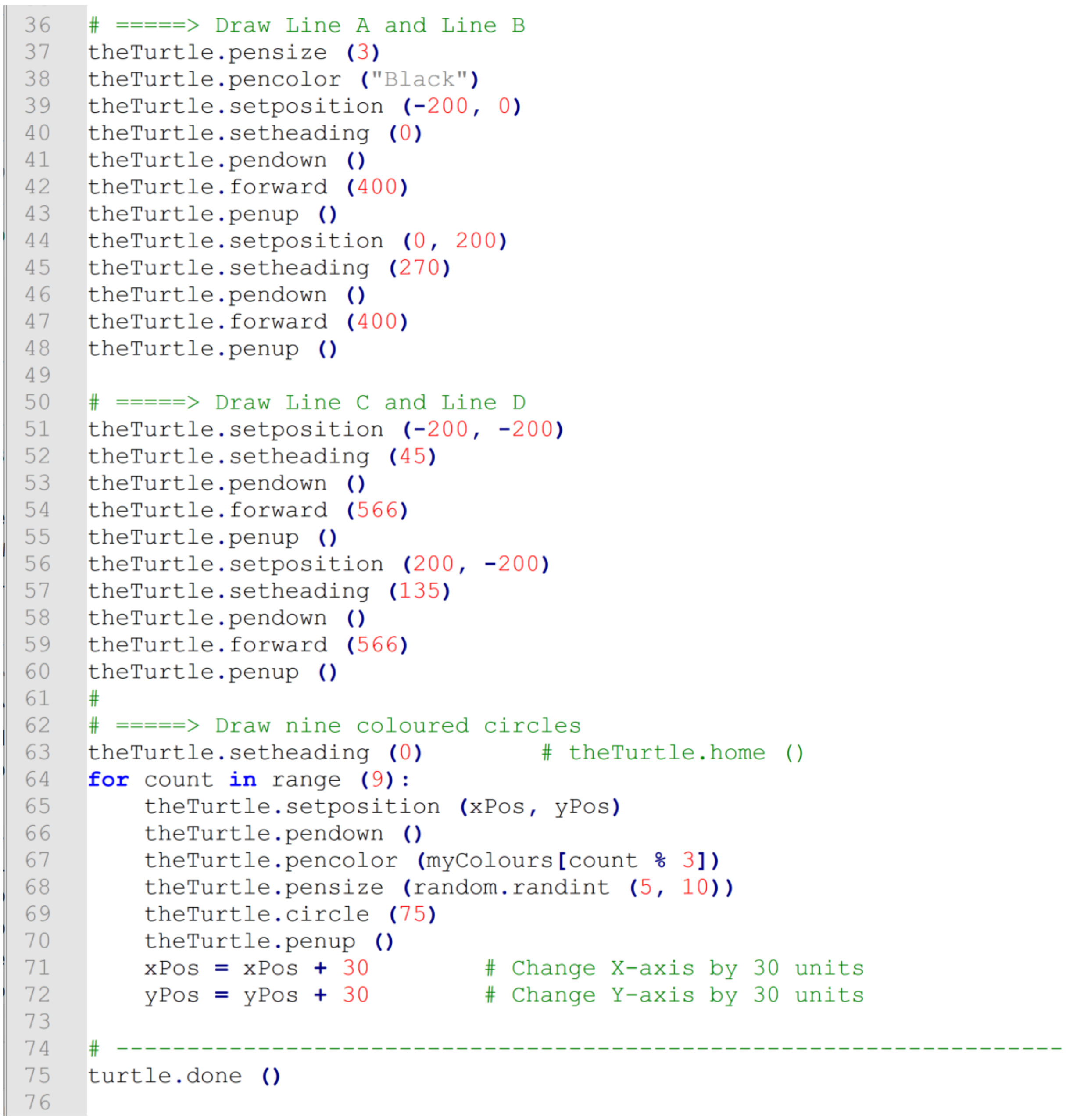
Solution design (levels-based mark scheme)

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**

```python
# -------------------------------------------------------------------
# Import libraries
# -------------------------------------------------------------------
# =====> Import the required libraries
import turtle
import random

# -------------------------------------------------------------------
# Constants
# -------------------------------------------------------------------
WIDTH = 800
HEIGHT = 600

# -------------------------------------------------------------------
# Global variables
# -------------------------------------------------------------------
myColours = ["peru", "olive", "gold"]
xPos = -120     # Location of circle 1, turtle pointing east
yPos = -195

# -------------------------------------------------------------------
# Main program
# -------------------------------------------------------------------
turtle.mode ("standard")    # Turtle points to the east
                            # Angles are counterclockwise
screen = turtle.Screen ()
screen.setup (WIDTH, HEIGHT)
turtle.screensize (WIDTH, HEIGHT)

# -------------------------------------------------------------------
# =====> Prepare the turtle, including speed
theTurtle = turtle.Turtle ()
theTurtle.speed (0)
theTurtle.penup ()

```



```python
# =====> Draw Line A and Line B
theTurtle.pensize (3)
theTurtle.pencolor ("Black")
theTurtle.setposition (-200, 0)
theTurtle.setheading (0)
theTurtle.pendown ()
theTurtle.forward (400)
theTurtle.penup ()
theTurtle.setposition (0, 200)
theTurtle.setheading (270)
theTurtle.pendown ()
theTurtle.forward (400)
theTurtle.penup ()

# =====> Draw Line C and Line D
theTurtle.setposition (-200, -200)
theTurtle.setheading (45)
theTurtle.pendown ()
theTurtle.forward (566)
theTurtle.penup ()
theTurtle.setposition (200, -200)
theTurtle.setheading (135)
theTurtle.pendown ()
theTurtle.forward (566)
theTurtle.penup ()
#
# =====> Draw nine coloured circles
theTurtle.setheading (0)        # theTurtle.home ()
for count in range (9):
    theTurtle.setposition (xPos, yPos)
    theTurtle.pendown ()
    theTurtle.pencolor (myColours[count % 3])
    theTurtle.pensize (random.randint (5, 10))
    theTurtle.circle (75)
    theTurtle.penup ()
    xPos = xPos + 30        # Change X-axis by 30 units
    yPos = yPos + 30        # Change Y-axis by 30 units

# -------------------------------------------------------------------
turtle.done ()
```

Question number	MP	Appx. Line	Answer	Additional guidance	Mark
6			Award marks as shown.		
	6.1		Loop for processing every record in the array	• Any method • Use of for/for each expected • `for record in theRecords:` • `for index in range (len (theRecords)):`	
	6.2		Range check on signal >= MIN_SIGNAL AND Range check on signal <= MAX_SIGNAL	• Lower bound • Allow 1 or 1.0 for lower bound • Allow 5 or 5.0 for upper bound • Allow equivalent statement • Allow reversed logic for invalid check	
	6.3		Mechanism for excluding non-alphanumeric characters, e.g. &	• Any method • Must exclude other characters, e.g. & • Do not award malformed function calls	

	6.4		Length check used to determine if key is valid or invalid.	• Any method
				• Testing for valid key:
				`len (inKey) >= MIN_KEY_LENGTH)`
				• Testing for invalid key:
				`len (inKey) < MIN_KEY_LENGTH)`
				• Code following test must match intent of the test
				• Allow > for >=
					(15)

	6.5		Mechanism for summing digits in the key	• Any method • Use of for() expected • Use of isdigit() expected • Use of int() expected • Allow not <string>.isalpha(), to skip over characters, rather than use of <string>.isdigit() to choose digits	
	6.6		Indication of two-dimensional indexing (key, reading)	• Any method	
			Levels-based mark scheme to a maximum of 9, from:	Considerations for levels-based mark scheme:	
	6.7 6.8 6.9		Solution design (3)	• [6.3.2] Use of constants throughout • [6.4.3] Length check of strings (key) should be done to prevent indexing errors when key is too short • [6.6.2] Use of subprograms (ideal problem for using isValid…)	
	6.10 6.11 6.12		Good programming practices (3)	• [6.1.4] Layout in visible sections with good, but not excessive, use of white space • [6.1.4] Sufficient, but not excessive, comments provided to explain blocks of logic • [6.1.4] Variable names are meaningful and data types appropriate to the problem and solution	
	6.13 6.14		Functionality (3)	• [6.1.5] No syntax or runtime errors in a reasonable attempt to solve the problem	

	6.15			• [6.1.5] Some logic errors in a reasonable attempt to address all the requirements • [6.1.5] Fully meets requirements of a solution • [6.4.1] User is informed which field is invalid, not just which record is invalid • [6.1.1] Use decomposition to solve problem and create solution • [6.1.2] Write in a high-level language	

Required Output (ignore format):

```
Invalid key ['7JA3B1', 3.41]

Invalid key ['A7B', 2.33]

Invalid key ['Y8R4K', 2.78]

Invalid key ['A1&2X3', 1.25]


Invalid signal ['B1P6Y7', -1.23]

Invalid signal ['F8D7L5', 5.17]

Invalid signal ['X0B9A9', 0]

Invalid signal ['Q6B7T3', 0.5]


Invalid key ['P3Y1M4V7', 4.35]

Invalid key ['X', 2.3]

Invalid key ['W64T18B1', 1.51]

Invalid key ['A3B1C14', 4.59]
```

Test data:

Record	Where	Reason
['7JA3B1', 3.41]	Key	Incorrect checksum
['A7B', 2.33]	Key	Missing checksum digit / checksum not equal to last character
['Y8R4K', 2.78]	Key	Does not end with a digit / checksum not equal to last character
['A1&2X3', 1.25]	Key	Contains a symbol, not alphabetic and digits only
['B1P6Y7', -1.23]	Signal	Below minimum
['F8D7L5', 5.17]	Signal	Above maximum
['X0B9A9', 0]	Signal	Below minimum
['Q6B7T3', 0.5]	Signal	Below minimum
['P3Y1M4V7', 4.35]	Key	Incorrect checksum
['X', 2.3]	Key	Key to short
['W64T18B1', 1.51]	Key	Incorrect checksum
['A3B1C14', 4.59]	Key	Incorrect checksum

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**

```python
# -------------------------------------------------------------------
# Constants
# -------------------------------------------------------------------
MIN_KEY_LENGTH = 3      # A00
MAX_SIGNAL = 5.0        # Max five
MIN_SIGNAL = 1.0        # Min 1

# -------------------------------------------------------------------
# Global variables
# -------------------------------------------------------------------
theRecords = [["7JA3B1", 3.41], ["A7B", 2.33], ["Y8R4K", 2.78],
              ["O9N1K0", 5.0], ["A1&2X3", 1.25],
              ["A12B3", 2.47], ["B1P6Y7", -1.23], ["F8D7L5", 5.17],
              ["AB23A5", 2.47], ["X0B9A9", 0], ["Q6B7T3", 0.5],
              ["A15B6C2", 2.56], ["A12340", 2.5], ["P3Y1M4V7", 4.35],
              ["J1H0Q1", 1.0], ["X", 2.3], ["W64T18B1", 1.51],
              ["A00", 1.99], ["A3B1C14", 4.59]
            ]

# -------------------------------------------------------------------
# Subprograms
# -------------------------------------------------------------------
def isKeyValid (pKey):
    valid = False

    # Length, mixture of letters and digits, ends with a digit
    # Not only letters, not only digits
    # No requirement to start with an uppercase letter
    if ((len (pKey) >= MIN_KEY_LENGTH) and (pKey.isalnum()) and
            (pKey[len (pKey) - 1].isdigit()) and
            (not pKey[0:len (pKey) - 1].isalpha ()) and
            (not pKey[0:len (pKey) - 1].isdigit ())):
        valid = True

    return (valid)

def calcCheckDigit (pKey):
    index = 0
    total = 0

    while (index < (len (pKey) - 2)):       # One short of check digit
        if (pKey[index].isdigit ()):
            total = total + int (pKey[index])
        index = index + 1

    # Only want the last digit of the calculated check digit
    strTotal = str (total)
    chkDigit = strTotal[len (strTotal) - 1]

    return (chkDigit)

```

```python
def isSignalValid (pSignal):
    valid = False

    if ((pSignal >= MIN_SIGNAL) and (pSignal <= MAX_SIGNAL)):
        valid = True

    return (valid)

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

for record in theRecords:
    keyPart = record[0]               # Ease of reading
    signalPart = record[1]
    checkDigitLocation = len (keyPart) - 1
    checkDigit = ""                   # A string for comparison

    if (isKeyValid (keyPart)):
        checkDigit = calcCheckDigit(keyPart)
        if (checkDigit == keyPart[checkDigitLocation]):
            if (not isSignalValid (signalPart)):
                print ("Invalid signal", record)
        else:
            print ("Invalid key", record)
    else:
        print ("Invalid key", record)
```