Examiners' Report
Principal Examiner Feedback

June 2022

Pearson Edexcel GCSE In
Computer Science (1CP2/02)
Paper 2: Application of Computational Thinking

June 2022
Publications Code 1CP2_02_ER_20220825

Introduction

This is the first examination of the Edexcel GCSE Computer Science (9-1), with the paper two onscreen exam. The programming language required is Python 3.

Students are supplied with a question paper, a programming language subset document, and a code file for each question. Students are required to amend the code files and save their work, using a different file name.

Centres compress the code file responses for each student. The compressed files are uploaded to Edexcel for external assessment, via the Learner Work Transfer program.

Centre submissions

The ICE document for this series set out the format in which students' completed code files were to be submitted. The majority of centres were able to follow the instructions accurately, ensuring that a single zipped file of the COMPLETED_CODE folder was provided for each student. The submissions were correctly identified with the centre and student number.

General

Attempting all questions

There were a number of scripts where students did not attempt Q05 and Q06, thereby missing an opportunity to access some marks. There are partial marks that could be awarded in each question. Students are reminded to attempt all the questions on the paper.

Readability

It is not necessary to comment every line of code in a solution. Some examples were seen where the number of comments exceeded the number of code lines. Comments are to help understand the logic, so should be placed, more helpfully, at the start of blocks of code. Excessive commenting makes the response difficult to read.

White space also can help with readability, but there is no requirement to double space code. Use white space between blocks of logic. Single spacing is appropriate for code.

Execute and test the code

In early questions, where precise instructions are given, students should attempt to run their code. Students were seen to have created the correct code line, but not used the correct indentation level. This made the code crash.

Spelling errors were also seen, which means that the code will not give the correct output. Students could find and correct these simple errors by running the code.

In the Parson's problem, the code should be executed with the test data given in the question paper. Execution would quickly identify that some lines were still out of order.

Q1 – Complete the code

Solutions required completion of the provided lines or adding new lines of Python code to the given file.

The majority of students submitted good responses.

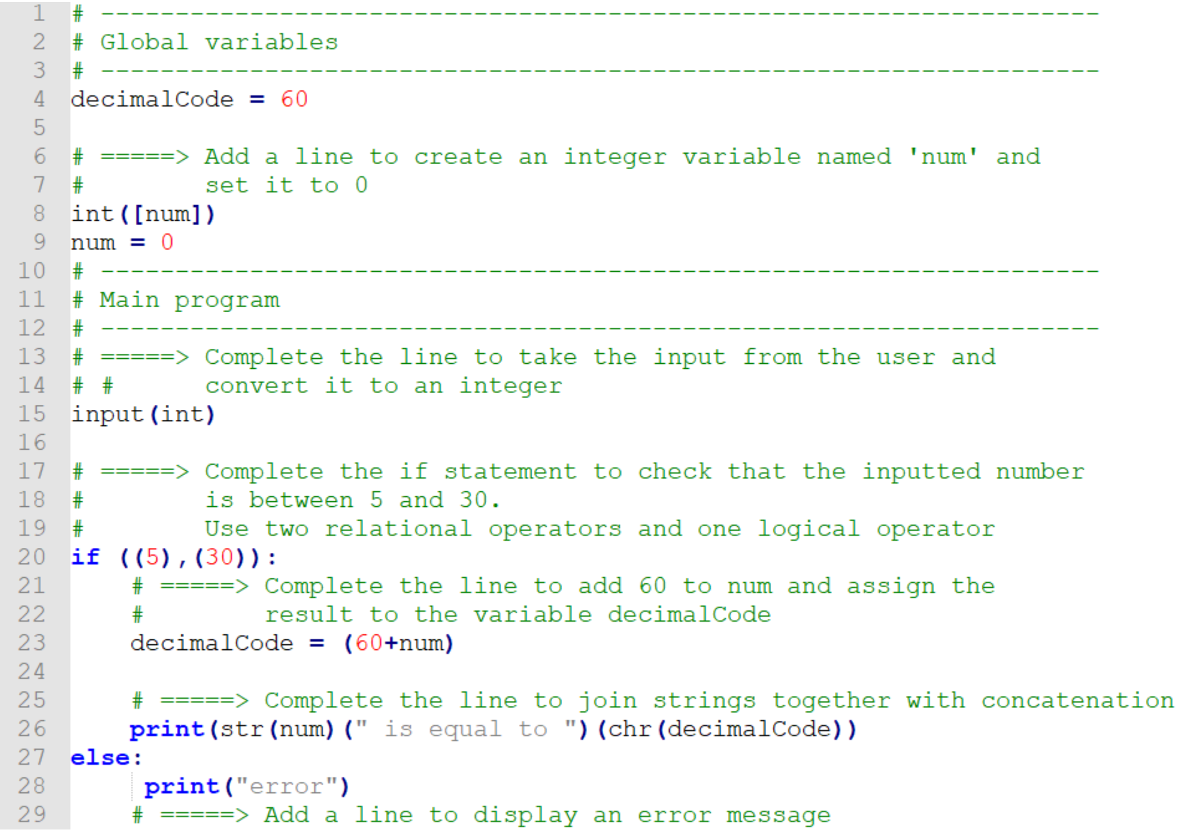
Fewer marks were awarded to responses that demonstrated deleting, rewriting, or rearranging sections of the supplied code.

Some responses demonstrated an inaccurate use of relational and logical operators.

Some responses used commas to join the strings on line 25. The use of commas to join strings produces a tuple of strings, not a single string. The `print()` function will accept a tuple of strings and display them one after another.

Valid range checks for the input numbers were awarded for those that excluded the values of 5 and 30 and those that included the values of 5 and 30.

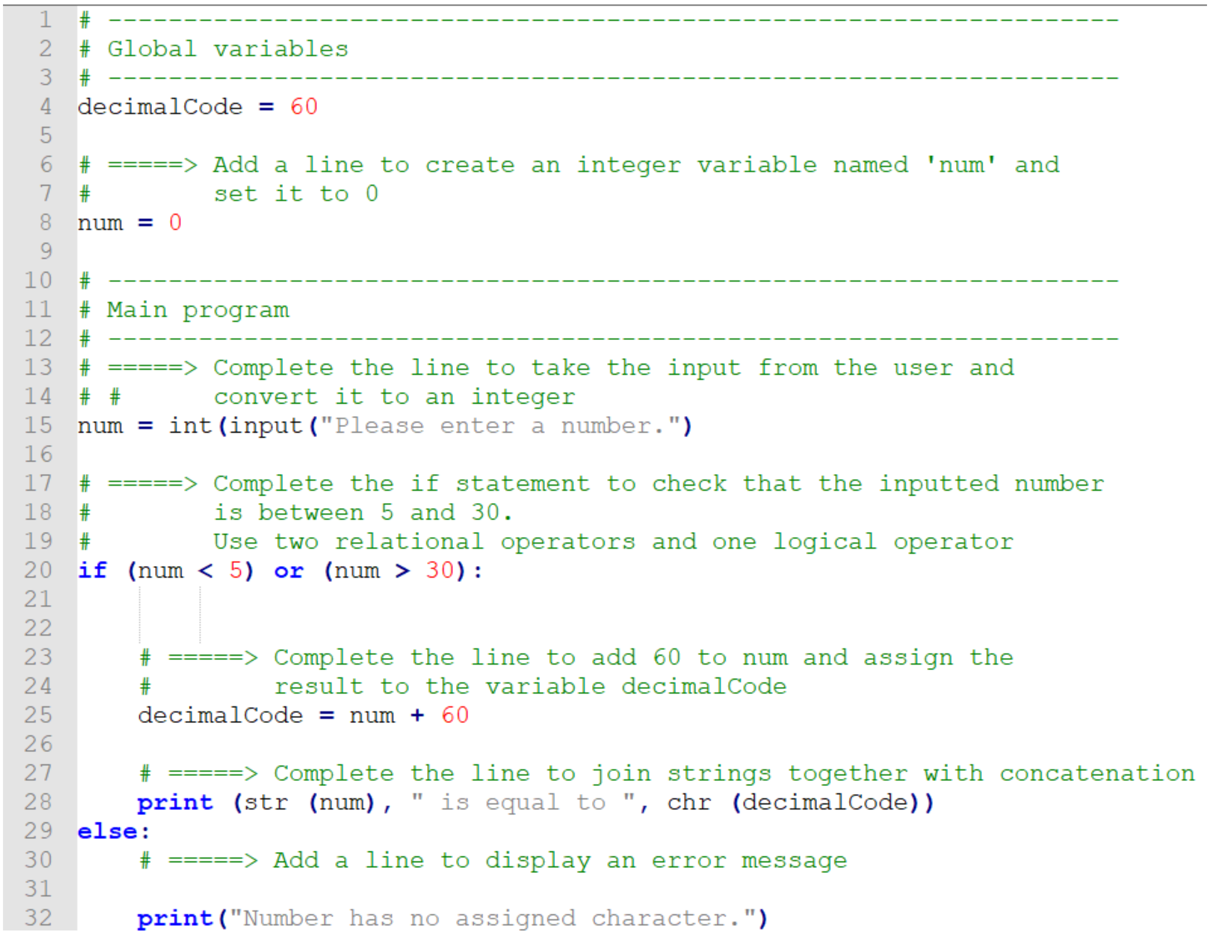
Q1 Example 1



```
 1  # -------------------------------------------------------------------
 2  # Global variables
 3  # -------------------------------------------------------------------
 4  decimalCode = 60
 5
 6  # =====> Add a line to create an integer variable named 'num' and
 7  #        set it to 0
 8  int([num])
 9  num = 0
10  # -------------------------------------------------------------------
11  # Main program
12  # -------------------------------------------------------------------
13  # =====> Complete the line to take the input from the user and
14  # #      convert it to an integer
15  input(int)
16
17  # =====> Complete the if statement to check that the inputted number
18  #        is between 5 and 30.
19  #        Use two relational operators and one logical operator
20  if ((5),(30)):
21      # =====> Complete the line to add 60 to num and assign the
22      #        result to the variable decimalCode
23      decimalCode = (60+num)
24
25      # =====> Complete the line to join strings together with concatenation
26      print(str(num)(" is equal to ")(chr(decimalCode))
27  else:
28       print("error")
29      # =====> Add a line to display an error message
```

This question was awarded five marks. It is a good example demonstrating understanding of the fundamental concepts of data types, arithmetic, assignment, and input and output.

Q1 Example 2



```python
# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
decimalCode = 60

# =====> Add a line to create an integer variable named 'num' and
#        set it to 0
num = 0

# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------
# =====> Complete the line to take the input from the user and
# #      convert it to an integer
num = int(input("Please enter a number.")

# =====> Complete the if statement to check that the inputted number
#        is between 5 and 30.
#        Use two relational operators and one logical operator
if (num < 5) or (num > 30):


    # =====> Complete the line to add 60 to num and assign the
    #        result to the variable decimalCode
    decimalCode = num + 60

    # =====> Complete the line to join strings together with concatenation
    print (str (num), " is equal to ", chr (decimalCode))
else:
    # =====> Add a line to display an error message

    print("Number has no assigned character.")
```

This response was awarded six marks. It includes a correct conversion between data types, in addition to the fundamental understanding of input and output.

Q02 – Fix the errors

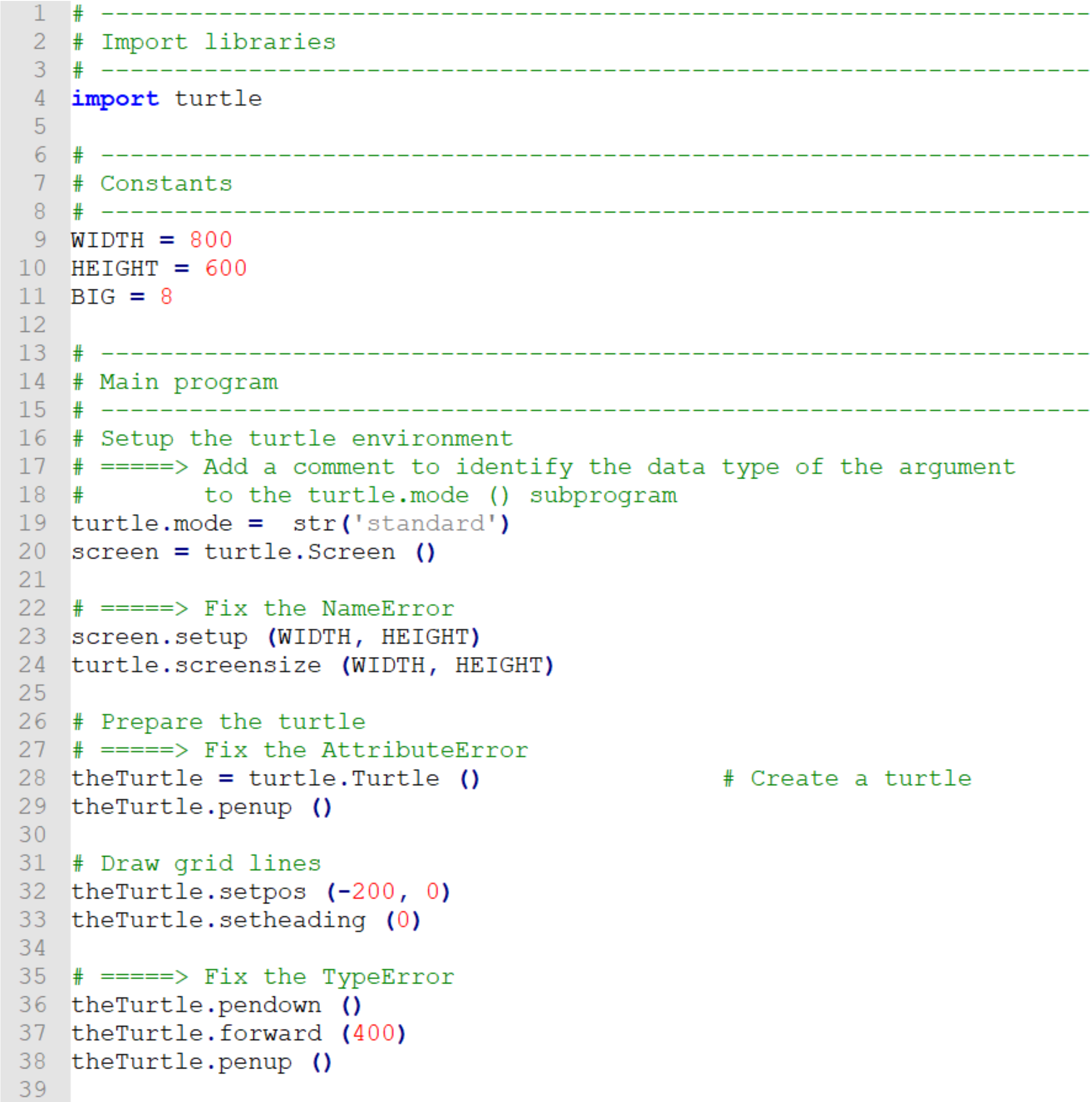
Solutions required students to identify a data type, fix syntax errors, fix logic errors, and add code to produce the required functionality. This question is set in the context of turtle graphics.

The majority of students submitted good responses.

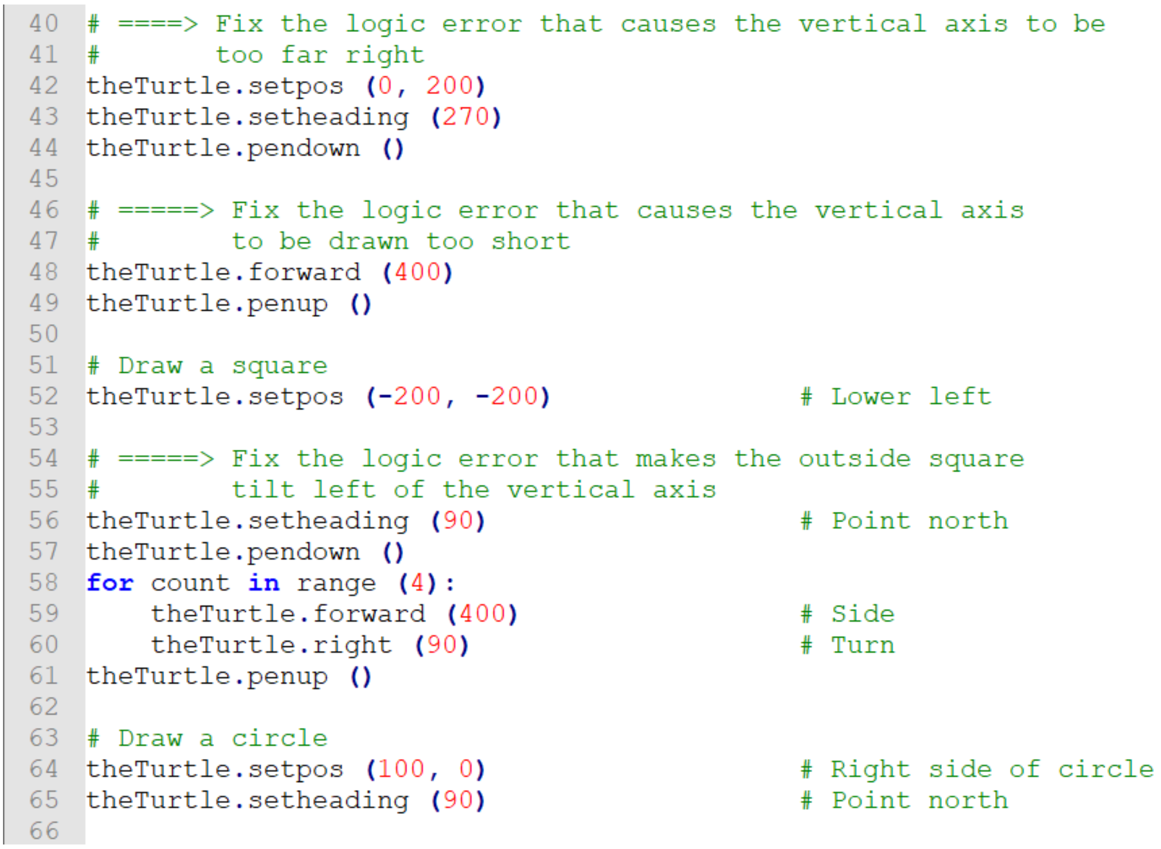
The most frequently lost mark was the comment required to identify the understanding of a string data type. Fixing the syntax errors were the most commonly awarded marks. Fixing the logic errors were the least commonly awarded marks.

Responses that used the variable `theTurtle` when adding the required new lines of code, received full marks. Those that used the default turtle, `turtle`, could not access all the available marks.

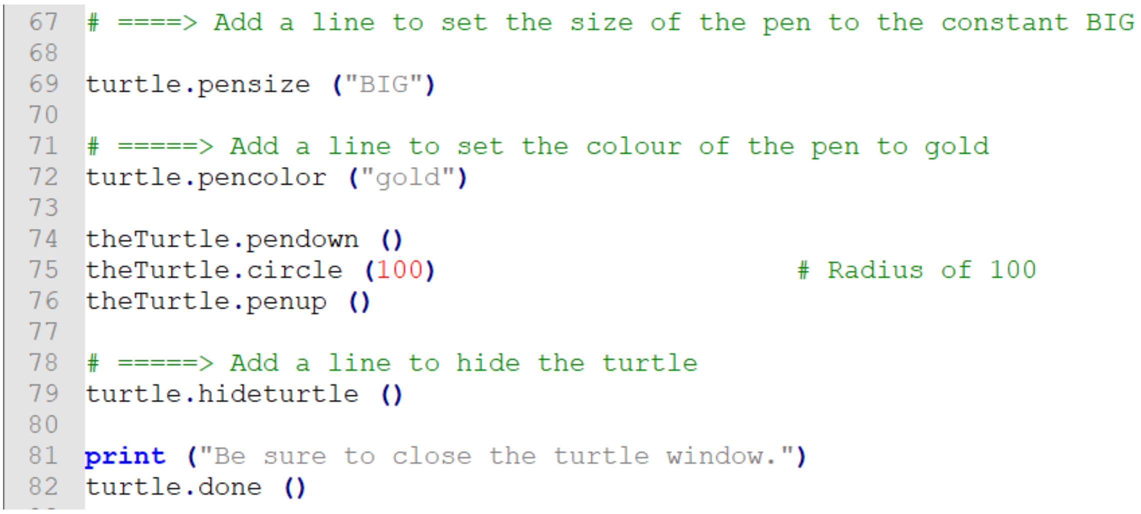
Q2 Example 1



```
# -------------------------------------------------------------------
# Import libraries
# -------------------------------------------------------------------
import turtle

# -------------------------------------------------------------------
# Constants
# -------------------------------------------------------------------
WIDTH = 800
HEIGHT = 600
BIG = 8

# -------------------------------------------------------------------
# Main program
# -------------------------------------------------------------------
# Setup the turtle environment
# =====> Add a comment to identify the data type of the argument
#        to the turtle.mode () subprogram
turtle.mode =  str('standard')
screen = turtle.Screen ()

# =====> Fix the NameError
screen.setup (WIDTH, HEIGHT)
turtle.screensize (WIDTH, HEIGHT)

# Prepare the turtle
# =====> Fix the AttributeError
theTurtle = turtle.Turtle ()                # Create a turtle
theTurtle.penup ()

# Draw grid lines
theTurtle.setpos (-200, 0)
theTurtle.setheading (0)

# =====> Fix the TypeError
theTurtle.pendown ()
theTurtle.forward (400)
theTurtle.penup ()

```



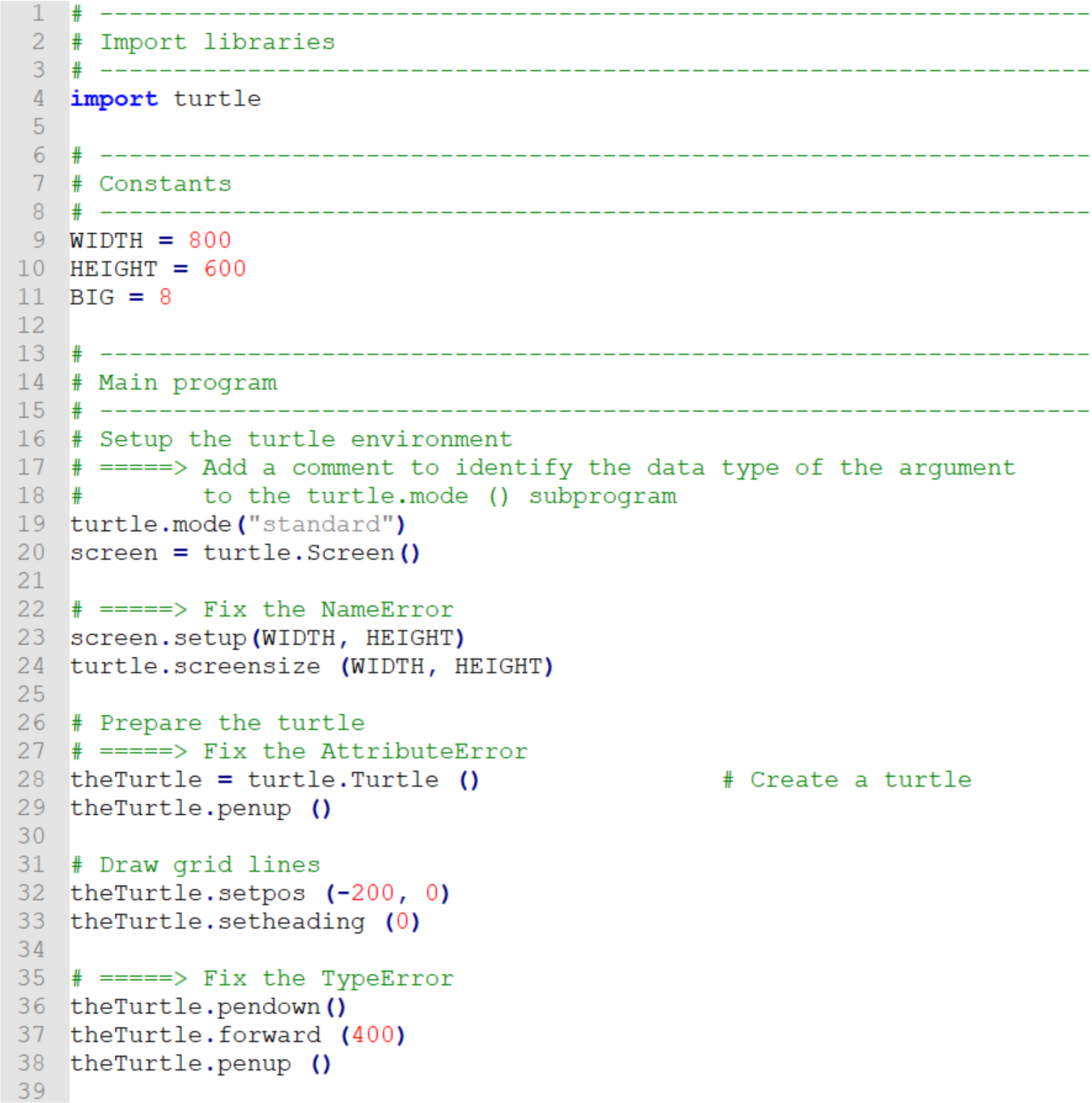
```
# ====> Fix the logic error that causes the vertical axis to be
#       too far right
theTurtle.setpos (0, 200)
theTurtle.setheading (270)
theTurtle.pendown ()

# =====> Fix the logic error that causes the vertical axis
#        to be drawn too short
theTurtle.forward (400)
theTurtle.penup ()

# Draw a square
theTurtle.setpos (-200, -200)               # Lower left

# =====> Fix the logic error that makes the outside square
#        tilt left of the vertical axis
theTurtle.setheading (90)                   # Point north
theTurtle.pendown ()
for count in range (4):
    theTurtle.forward (400)                 # Side
    theTurtle.right (90)                    # Turn
theTurtle.penup ()

# Draw a circle
theTurtle.setpos (100, 0)                   # Right side of circle
theTurtle.setheading (90)                   # Point north

```



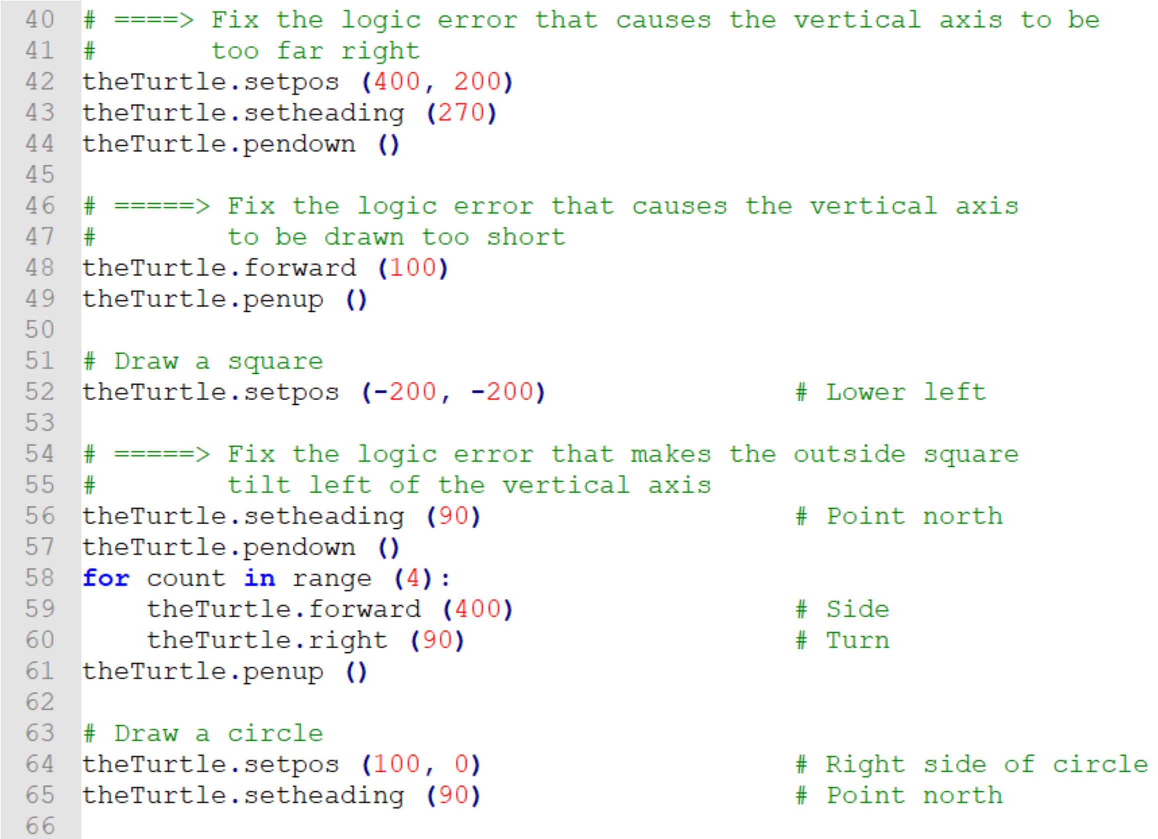
```
# ====> Add a line to set the size of the pen to the constant BIG

turtle.pensize ("BIG")

# =====> Add a line to set the colour of the pen to gold
turtle.pencolor ("gold")

theTurtle.pendown ()
theTurtle.circle (100)                      # Radius of 100
theTurtle.penup ()

# =====> Add a line to hide the turtle
turtle.hideturtle ()

print ("Be sure to close the turtle window.")
turtle.done ()
```

This response was awarded seven marks. This is a good example showing correction of the three runtime errors and the three logic errors.

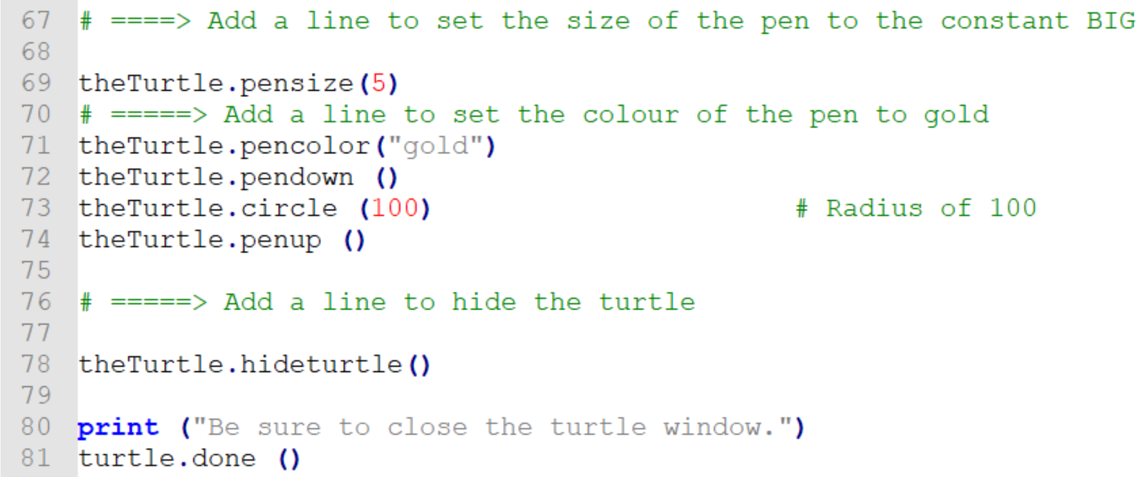
Q2 Example 2



```
# -------------------------------------------------------------------
# Import libraries
# -------------------------------------------------------------------
import turtle

# -------------------------------------------------------------------
# Constants
# -------------------------------------------------------------------
WIDTH = 800
HEIGHT = 600
BIG = 8

# -------------------------------------------------------------------
# Main program
# -------------------------------------------------------------------
# Setup the turtle environment
# =====> Add a comment to identify the data type of the argument
#        to the turtle.mode () subprogram
turtle.mode("standard")
screen = turtle.Screen()

# =====> Fix the NameError
screen.setup(WIDTH, HEIGHT)
turtle.screensize (WIDTH, HEIGHT)

# Prepare the turtle
# =====> Fix the AttributeError
theTurtle = turtle.Turtle ()                # Create a turtle
theTurtle.penup ()

# Draw grid lines
theTurtle.setpos (-200, 0)
theTurtle.setheading (0)

# =====> Fix the TypeError
theTurtle.pendown()
theTurtle.forward (400)
theTurtle.penup ()

```



```
# ====> Fix the logic error that causes the vertical axis to be
#        too far right
theTurtle.setpos (400, 200)
theTurtle.setheading (270)
theTurtle.pendown ()

# =====> Fix the logic error that causes the vertical axis
#         to be drawn too short
theTurtle.forward (100)
theTurtle.penup ()

# Draw a square
theTurtle.setpos (-200, -200)               # Lower left

# =====> Fix the logic error that makes the outside square
#         tilt left of the vertical axis
theTurtle.setheading (90)                   # Point north
theTurtle.pendown ()
for count in range (4):
    theTurtle.forward (400)                 # Side
    theTurtle.right (90)                    # Turn
theTurtle.penup ()

# Draw a circle
theTurtle.setpos (100, 0)                   # Right side of circle
theTurtle.setheading (90)                   # Point north

```



```
# ====> Add a line to set the size of the pen to the constant BIG

theTurtle.pensize(5)
# =====> Add a line to set the colour of the pen to gold
theTurtle.pencolor("gold")
theTurtle.pendown ()
theTurtle.circle (100)                      # Radius of 100
theTurtle.penup ()

# =====> Add a line to hide the turtle

theTurtle.hideturtle()

print ("Be sure to close the turtle window.")
turtle.done ()
```

This example was awarded six marks. Again, the runtime errors have been corrected. In addition, the colour of the pen is changed and the correct turtle is hidden.

Q03

Solutions required completion of the given code or adding new lines of Python code to the given file.

The majority of students submitted good responses.

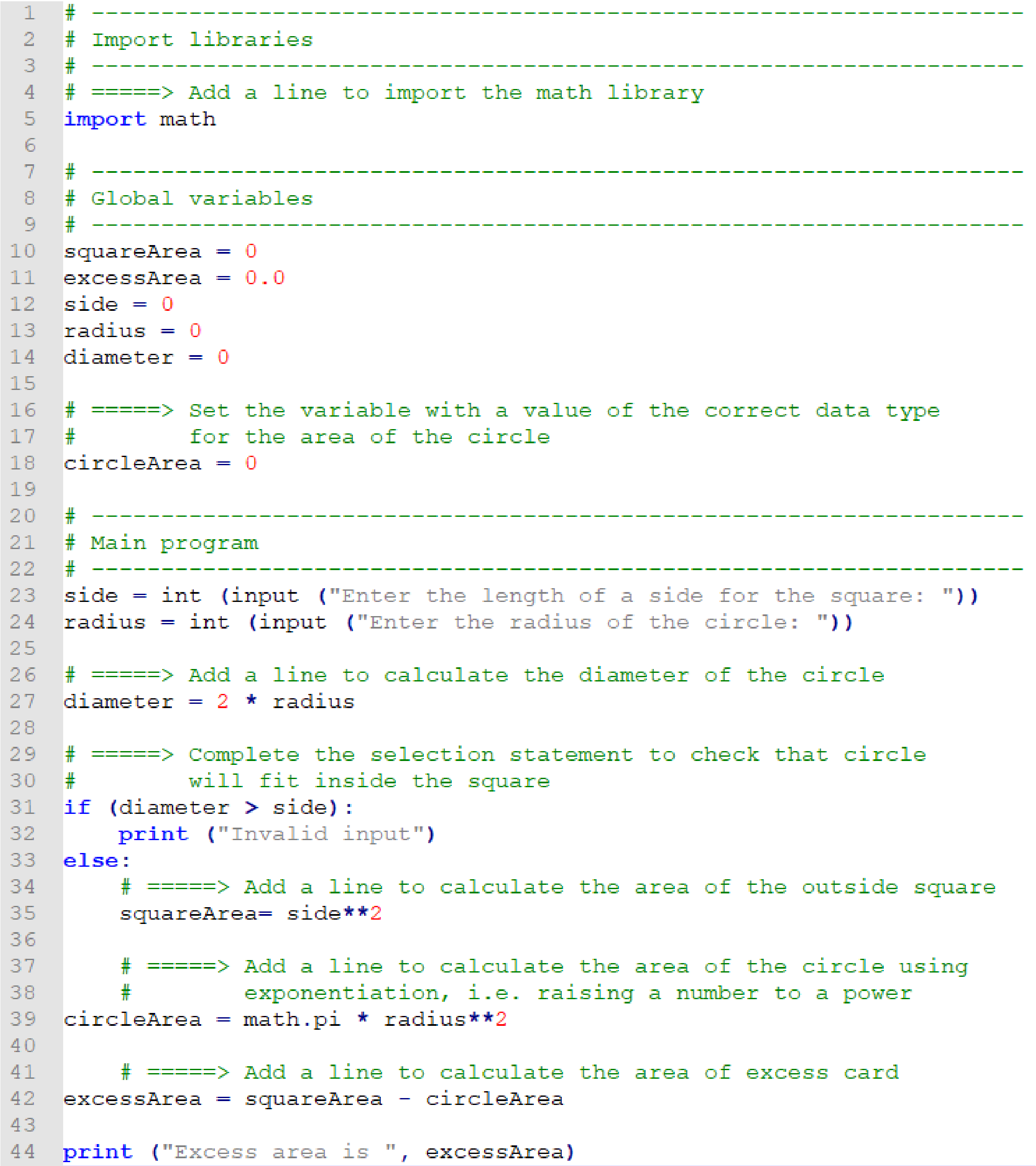
Fewer marks were awarded to responses that demonstrated deleting, rewriting, or rearranging sections of the supplied code.

The most frequent error was initialisation of the variable for the area of a circle to an integer value, rather than a real value, with a decimal.

Again, errors were seen in the use of relational operators.

The marks in the levels-based mark scheme for functionality were awarded based on the translation, execution, and accurate outputs produced by the response.

Q3 Example 1



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

# --------------------------------------------------------------------
# Global variables
# --------------------------------------------------------------------
squareArea = 0
excessArea = 0.0
side = 0
radius = 0
diameter = 0

# =====> Set the variable with a value of the correct data type
#        for the area of the circle
circleArea = 0

# --------------------------------------------------------------------
# Main program
# --------------------------------------------------------------------
side = int (input ("Enter the length of a side for the square: "))
radius = int (input ("Enter the radius of the circle: "))

# =====> Add a line to calculate the diameter of the circle
diameter = 2 * radius

# =====> Complete the selection statement to check that circle
#        will fit inside the square
if (diameter > side):
    print ("Invalid input")
else:
    # =====> Add a line to calculate the area of the outside square
    squareArea= side**2

    # =====> Add a line to calculate the area of the circle using
    #        exponentiation, i.e. raising a number to a power
circleArea = math.pi * radius**2

    # =====> Add a line to calculate the area of excess card
excessArea = squareArea - circleArea

print ("Excess area is ", excessArea)
```

This response was awarded eight marks. It is a very good example that demonstrates the skills of code construction. However, as indicated above, the indentation has introduced inaccurate behaviours.

Q3 Example 2

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

# -------------------------------------------------------------------
# Global variables
# -------------------------------------------------------------------
squareArea = 0
excessArea = 0.0
side = 0
radius = 0
diameter = 0

# =====> Set the variable with a value of the correct data type
#        for the area of the circle
circleArea = 0.0

# -------------------------------------------------------------------
# Main program
# -------------------------------------------------------------------
side = int (input ("Enter the length of a side for the square: "))
radius = int (input ("Enter the radius of the circle: "))

# =====> Add a line to calculate the diameter of the circle
diameter = radius * 2

# =====> Complete the selection statement to check that circle
#        will fit inside the square
if (diameter >= side):
    print ("Invalid input")
else:
    # =====> Add a line to calculate the area of the outside square
    sqaureArea = side**2

    # =====> Add a line to calculate the area of the circle using
    #        exponentiation, i.e. raising a number to a power
    circleArea = radius**2 * math.pi ()

    # =====> Add a line to calculate the area ofexcess card
    excessArea = squareArea - circleArea

    print ("Excess area is ", excessArea)
```

This response was awarded seven marks. It is a good example demonstrating skills in constructing individual lines of code. However, the solution either crashes or the outputs do not meet requirements.

Q04 – Parsons problem

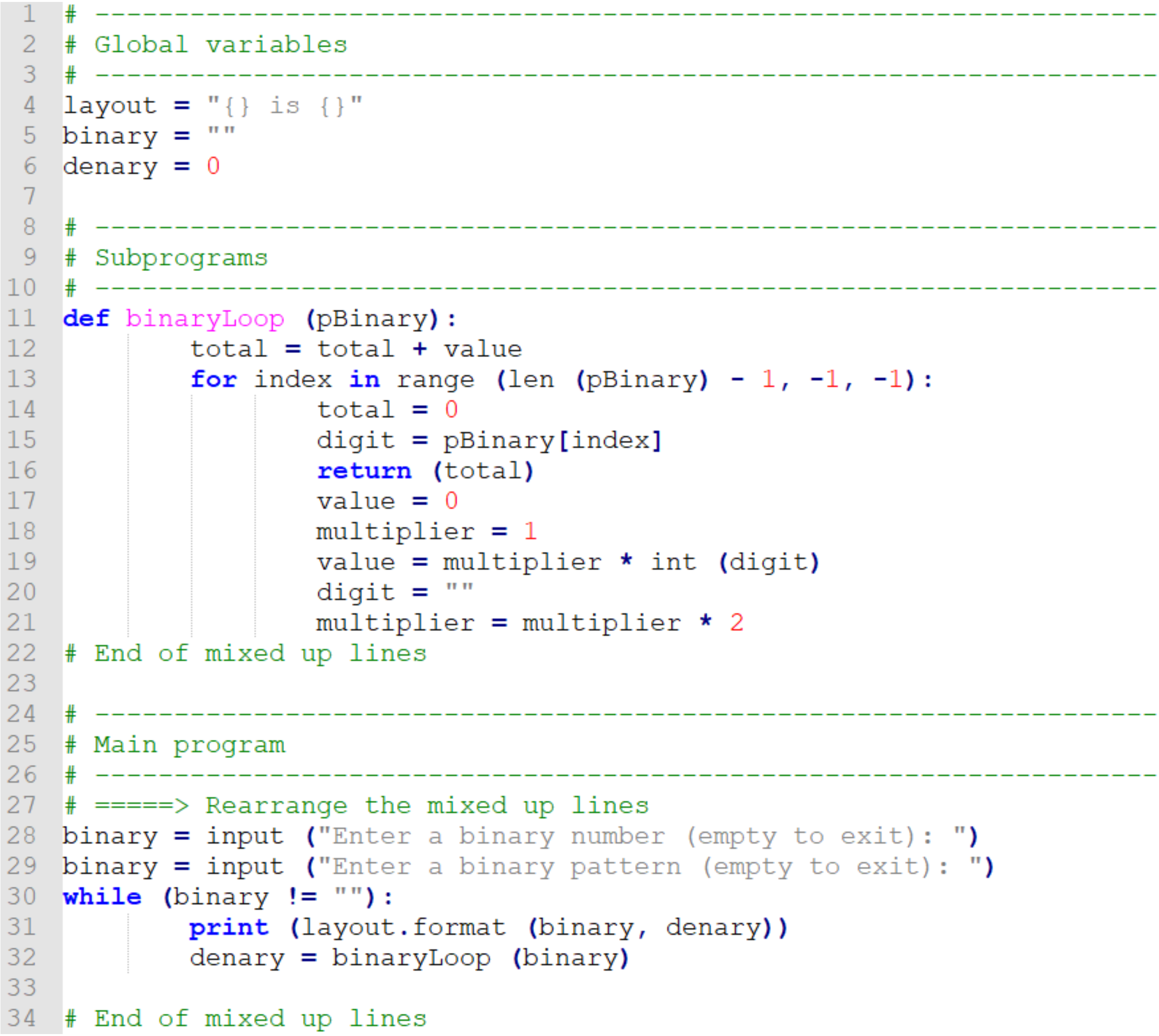
This question required reordering of the provided code lines to create a functional program that converted binary patterns to unsigned integer values. The lines of code were presented as a subprogram and a main program. The levels of indentation for each line were correct in the given code file.

Many responses were seen that achieved full marks on this question.

Some responses changed the levels of indentation for the supplied lines. Doing this means that the response will not execute as designed. As a result, those responses could not access the last two mark points, which were based on the functionality of the final result.

Where there was no vertical movement in a section, subprogram or main program, no marks could be awarded.

Q04 Example 1



```
# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
layout = "{} is {}"
binary = ""
denary = 0

# ------------------------------------------------------------------
# Subprograms
# ------------------------------------------------------------------
def binaryLoop (pBinary):
    total = total + value
    for index in range (len (pBinary) - 1, -1, -1):
        total = 0
        digit = pBinary[index]
        return (total)
        value = 0
        multiplier = 1
        value = multiplier * int (digit)
        digit = ""
        multiplier = multiplier * 2
# End of mixed up lines

# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------
# =====> Rearrange the mixed up lines
binary = input ("Enter a binary number (empty to exit): ")
binary = input ("Enter a binary pattern (empty to exit): ")
while (binary != ""):
    print (layout.format (binary, denary))
    denary = binaryLoop (binary)

# End of mixed up lines
```

This response was awarded four marks. Reordering of the lines has been attempted and there are no indentation errors introduced. This response demonstrates, even in high-tariff questions, there are opportunities to earn marks.

Q04 Example 2

```python
# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
layout = "{} is {}"
binary = ""
denary = 0

# ------------------------------------------------------------------
# Subprograms
# ------------------------------------------------------------------
def binaryLoop (pBinary):
# =====> Rearrange the mixed up lines
    while (binary != ""):
        total = 0
        value = 0
        multiplier = 1
        digit = ""
        for index in range (len (pBinary) - 1, -1, -1):
            digit = pBinary[index]
            value = multiplier * int (digit)
            multiplier = multiplier * 2
            total = total + value
        return (total)
# End of mixed up lines

# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------
# =====> Rearrange the mixed up lines
binary = input ("Enter a binary number (empty to exit): ")
binary = input ("Enter a binary pattern (empty to exit): ")
denary = binaryLoop (binary)
print (layout.format (binary, denary))

# End of mixed up lines
```

This response was awarded 11 marks. The majority of the given lines have been ordered correctly and there are no indentation errors. Although the response does execute, the outputs do not meet requirements.

Q05 – Writing a file

This question required responses that opened a file, wrote records from an internal data structure to that file, and then closed the file.

There were many creative responses to this problem, some of which achieved full marks.

The levels-based mark schemes for design and functionality provided students with opportunities to be rewarded for their approaches to the solution, independently of its functionality.

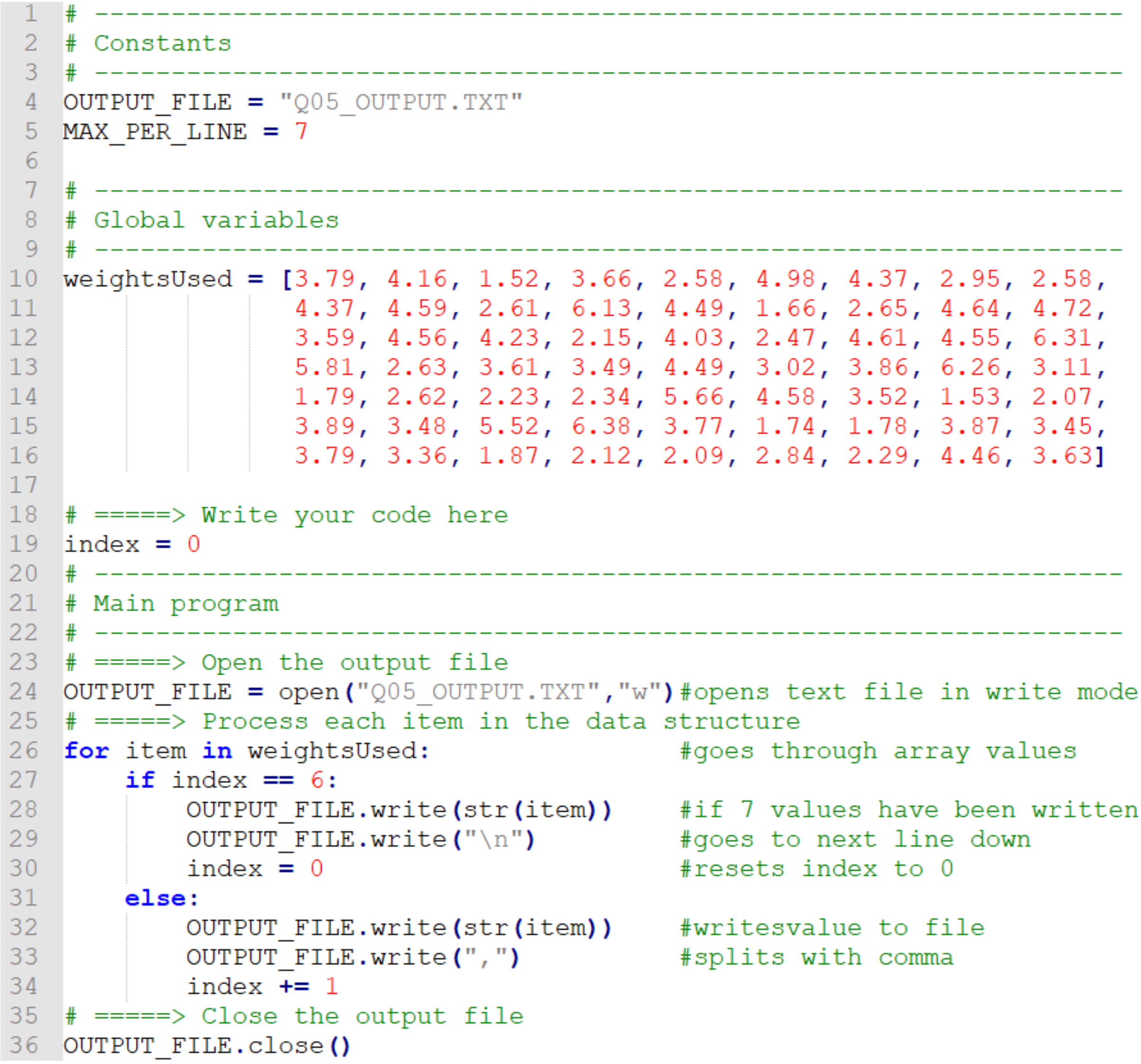
Simple design approaches, such as iterating across the array, building an output string of seven columns, and then writing that string to the file were seen. They often achieved full marks.

Other solutions included slicing the data structure into a sequence of lists, converting the lists to strings, and writing them to the file. These solutions often did not handle the commas correctly, as the conversion from a list to a string introduced spaces as well as commas and square brackets.

Two common errors included the use of the constants provided in the file. The file name constant was used inaccurately for the file open instruction. The constant controlling for the seven columns was either not used or used in combinations with a hard-coded seven.

Although the modulus function was not required in the solution, many responses included it in the calculation for determining when a line feed was needed in the output.

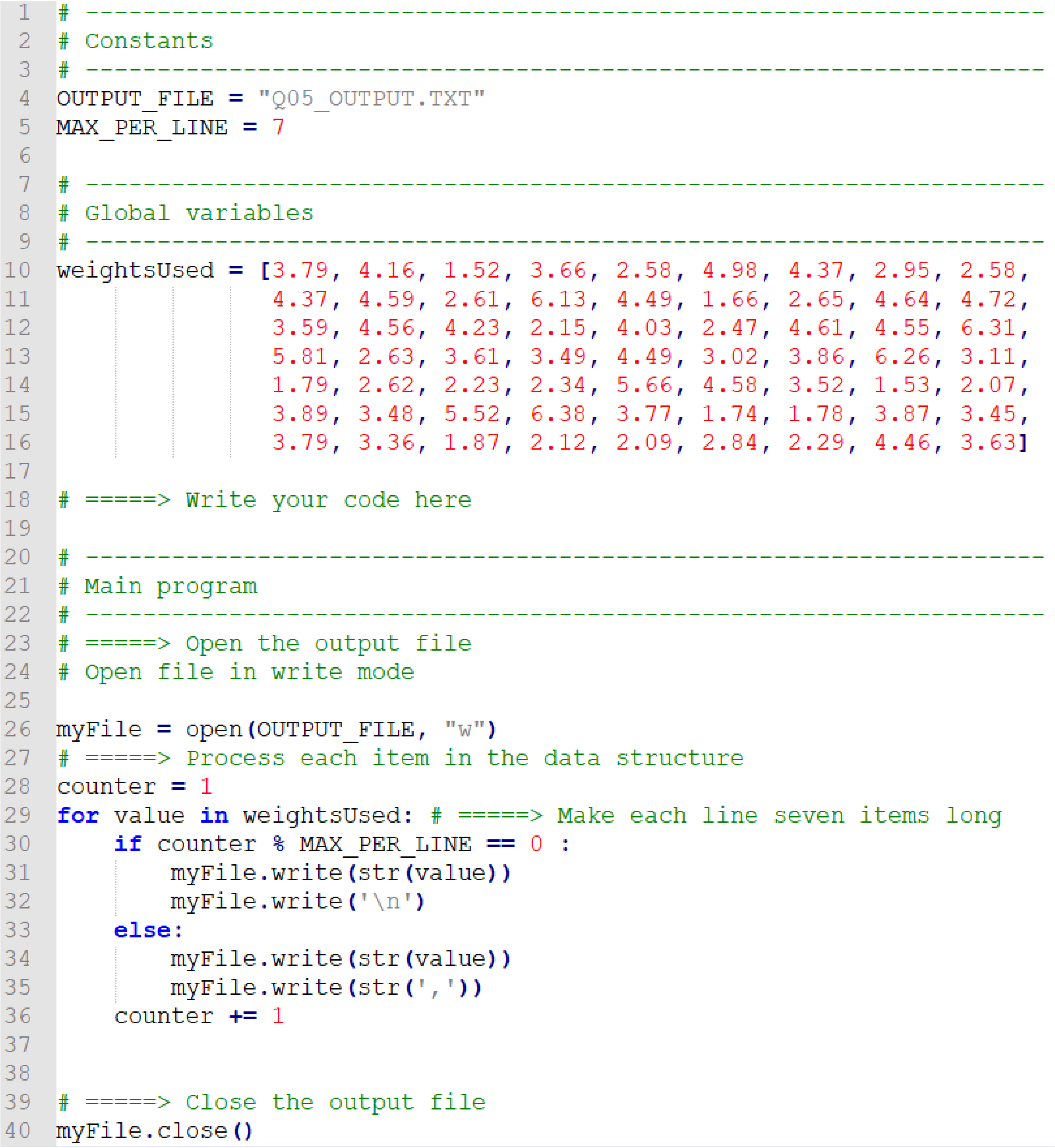
Q5 Example 1



```
# -------------------------------------------------------------------
# Constants
# -------------------------------------------------------------------
OUTPUT_FILE = "Q05_OUTPUT.TXT"
MAX_PER_LINE = 7

# -------------------------------------------------------------------
# Global variables
# -------------------------------------------------------------------
weightsUsed = [3.79, 4.16, 1.52, 3.66, 2.58, 4.98, 4.37, 2.95, 2.58,
               4.37, 4.59, 2.61, 6.13, 4.49, 1.66, 2.65, 4.64, 4.72,
               3.59, 4.56, 4.23, 2.15, 4.03, 2.47, 4.61, 4.55, 6.31,
               5.81, 2.63, 3.61, 3.49, 4.49, 3.02, 3.86, 6.26, 3.11,
               1.79, 2.62, 2.23, 2.34, 5.66, 4.58, 3.52, 1.53, 2.07,
               3.89, 3.48, 5.52, 6.38, 3.77, 1.74, 1.78, 3.87, 3.45,
               3.79, 3.36, 1.87, 2.12, 2.09, 2.84, 2.29, 4.46, 3.63]

# =====> Write your code here
index = 0
# -------------------------------------------------------------------
# Main program
# -------------------------------------------------------------------
# =====> Open the output file
OUTPUT_FILE = open("Q05_OUTPUT.TXT","w")#opens text file in write mode
# =====> Process each item in the data structure
for item in weightsUsed:                #goes through array values
    if index == 6:
        OUTPUT_FILE.write(str(item))    #if 7 values have been written
        OUTPUT_FILE.write("\n")         #goes to next line down
        index = 0                       #resets index to 0
    else:
        OUTPUT_FILE.write(str(item))    #writesvalue to file
        OUTPUT_FILE.write(",")          #splits with comma
        index += 1
# =====> Close the output file
OUTPUT_FILE.close()
```

This response was awarded 13 marks. This is a good response that demonstrates logical and coherent solution design, dealing with the requirement for seven columns per line. The solution is functional and the outputs meet the requirements set out in the question paper.

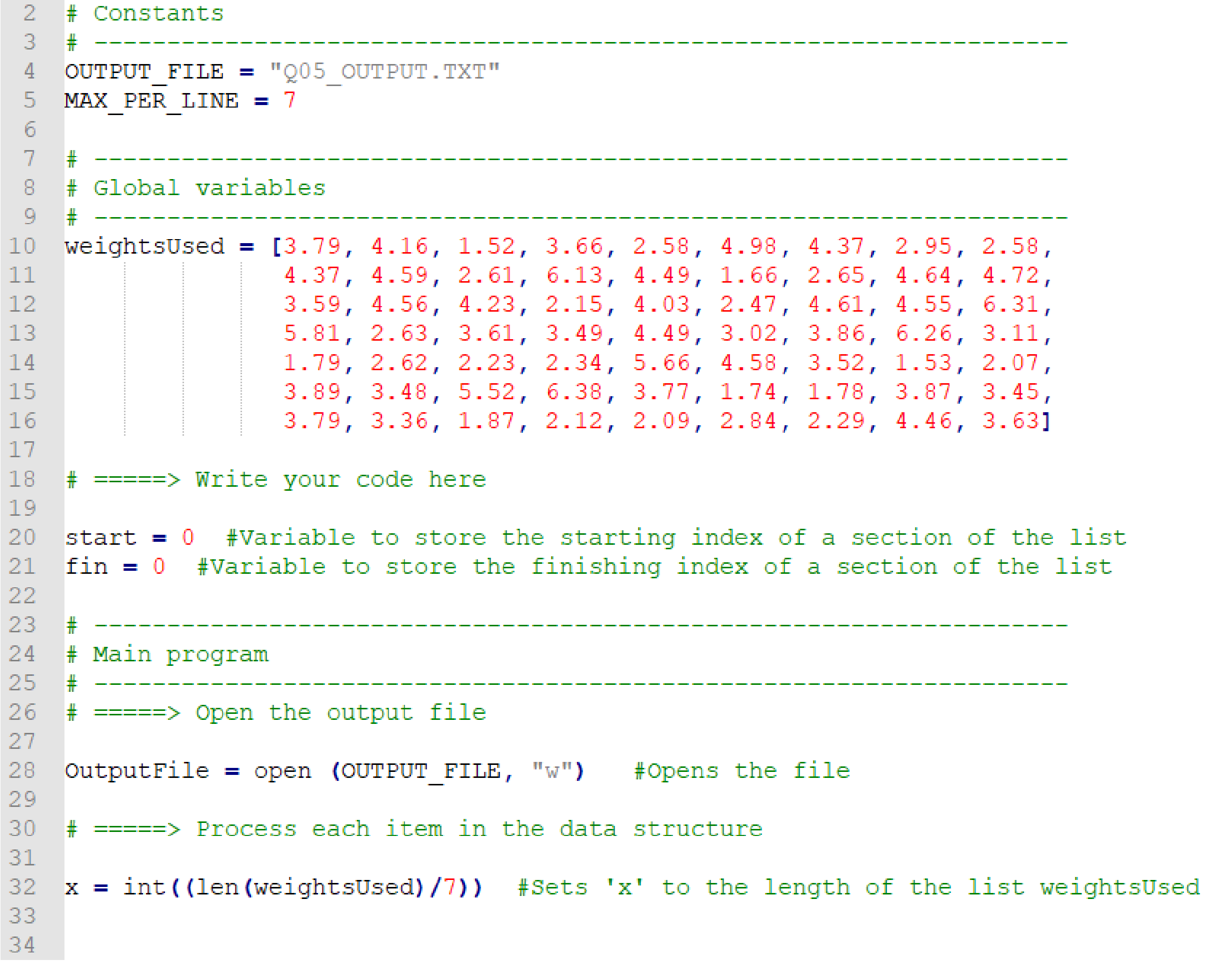
Q5 Example 2



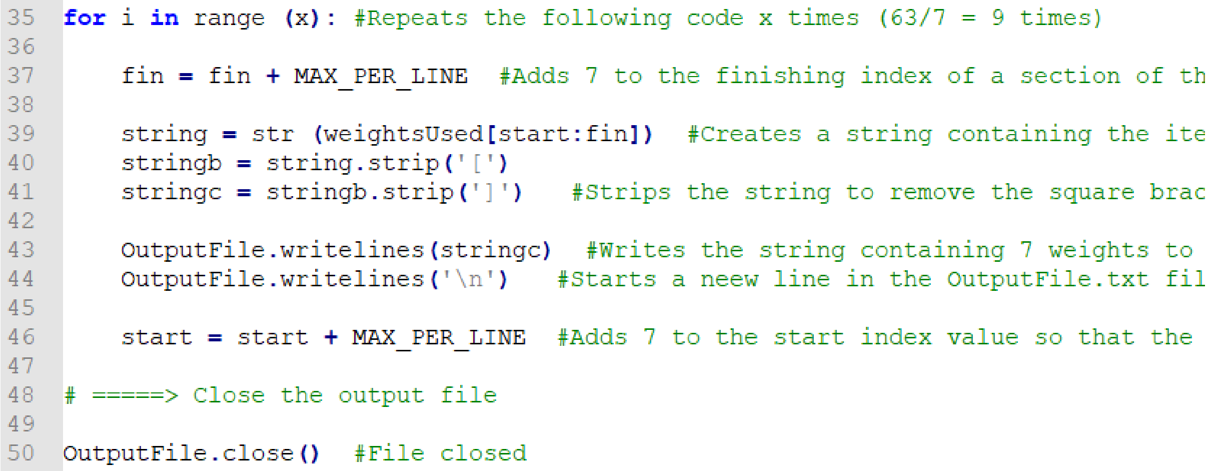
```
# ---------------------------------------------------------------------
# Constants
# ---------------------------------------------------------------------
OUTPUT_FILE = "Q05_OUTPUT.TXT"
MAX_PER_LINE = 7

# ---------------------------------------------------------------------
# Global variables
# ---------------------------------------------------------------------
weightsUsed = [3.79, 4.16, 1.52, 3.66, 2.58, 4.98, 4.37, 2.95, 2.58,
               4.37, 4.59, 2.61, 6.13, 4.49, 1.66, 2.65, 4.64, 4.72,
               3.59, 4.56, 4.23, 2.15, 4.03, 2.47, 4.61, 4.55, 6.31,
               5.81, 2.63, 3.61, 3.49, 4.49, 3.02, 3.86, 6.26, 3.11,
               1.79, 2.62, 2.23, 2.34, 5.66, 4.58, 3.52, 1.53, 2.07,
               3.89, 3.48, 5.52, 6.38, 3.77, 1.74, 1.78, 3.87, 3.45,
               3.79, 3.36, 1.87, 2.12, 2.09, 2.84, 2.29, 4.46, 3.63]

# =====> Write your code here

# ---------------------------------------------------------------------
# Main program
# ---------------------------------------------------------------------
# =====> Open the output file
# Open file in write mode

myFile = open(OUTPUT_FILE, "w")
# =====> Process each item in the data structure
counter = 1
for value in weightsUsed: # =====> Make each line seven items long
    if counter % MAX_PER_LINE == 0 :
        myFile.write(str(value))
        myFile.write('\n')
    else:
        myFile.write(str(value))
        myFile.write(str(','))
    counter += 1


# =====> Close the output file
myFile.close()
```

This response was awarded 15 marks. In this response, the constants have been used correctly. It is another good example of solution decomposition and functionality.

Q5 Example 3



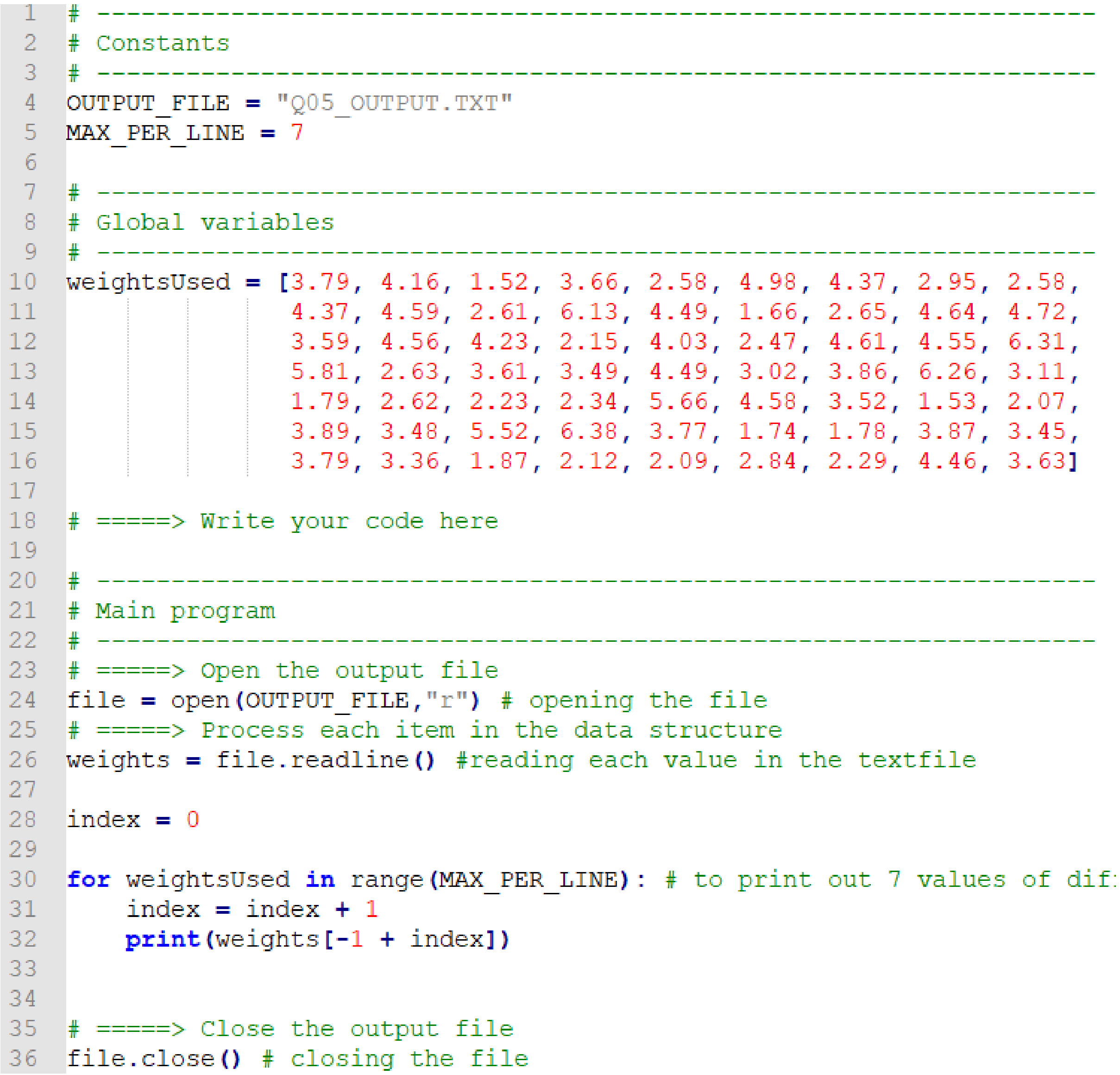
```
# Constants
# ---------------------------------------------------------------------
OUTPUT_FILE = "Q05_OUTPUT.TXT"
MAX_PER_LINE = 7

# ---------------------------------------------------------------------
# Global variables
# ---------------------------------------------------------------------
weightsUsed = [3.79, 4.16, 1.52, 3.66, 2.58, 4.98, 4.37, 2.95, 2.58,
               4.37, 4.59, 2.61, 6.13, 4.49, 1.66, 2.65, 4.64, 4.72,
               3.59, 4.56, 4.23, 2.15, 4.03, 2.47, 4.61, 4.55, 6.31,
               5.81, 2.63, 3.61, 3.49, 4.49, 3.02, 3.86, 6.26, 3.11,
               1.79, 2.62, 2.23, 2.34, 5.66, 4.58, 3.52, 1.53, 2.07,
               3.89, 3.48, 5.52, 6.38, 3.77, 1.74, 1.78, 3.87, 3.45,
               3.79, 3.36, 1.87, 2.12, 2.09, 2.84, 2.29, 4.46, 3.63]

# =====> Write your code here

start = 0  #Variable to store the starting index of a section of the list
fin = 0  #Variable to store the finishing index of a section of the list

# ---------------------------------------------------------------------
# Main program
# ---------------------------------------------------------------------
# =====> Open the output file

OutputFile = open (OUTPUT_FILE, "w")   #Opens the file

# =====> Process each item in the data structure

x = int((len(weightsUsed)/7))  #Sets 'x' to the length of the list weightsUsed


```



```
for i in range (x): #Repeats the following code x times (63/7 = 9 times)

    fin = fin + MAX_PER_LINE  #Adds 7 to the finishing index of a section of th

    string = str (weightsUsed[start:fin])  #Creates a string containing the ite
    stringb = string.strip('[')
    stringc = stringb.strip(']')   #Strips the string to remove the square brac

    OutputFile.writelines(stringc)  #Writes the string containing 7 weights to
    OutputFile.writelines('\n')   #Starts a neew line in the OutputFile.txt fil

    start = start + MAX_PER_LINE  #Adds 7 to the start index value so that the

# =====> Close the output file

OutputFile.close()  #File closed
```

This response was awarded 13 marks. This is a good example demonstrating the use of slicing and conversion of a list to a string for outputting. However, the conversion has introduced a space after the commas, so it is not fully functional.

Q5 Example 4



```
# ------------------------------------------------------------------
# Constants
# ------------------------------------------------------------------
OUTPUT_FILE = "Q05_OUTPUT.TXT"
MAX_PER_LINE = 7

# ------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------
weightsUsed = [3.79, 4.16, 1.52, 3.66, 2.58, 4.98, 4.37, 2.95, 2.58,
               4.37, 4.59, 2.61, 6.13, 4.49, 1.66, 2.65, 4.64, 4.72,
               3.59, 4.56, 4.23, 2.15, 4.03, 2.47, 4.61, 4.55, 6.31,
               5.81, 2.63, 3.61, 3.49, 4.49, 3.02, 3.86, 6.26, 3.11,
               1.79, 2.62, 2.23, 2.34, 5.66, 4.58, 3.52, 1.53, 2.07,
               3.89, 3.48, 5.52, 6.38, 3.77, 1.74, 1.78, 3.87, 3.45,
               3.79, 3.36, 1.87, 2.12, 2.09, 2.84, 2.29, 4.46, 3.63]

# =====> Write your code here

# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------
# =====> Open the output file
file = open(OUTPUT_FILE,"r") # opening the file
# =====> Process each item in the data structure
weights = file.readline() #reading each value in the textfile

index = 0

for weightsUsed in range(MAX_PER_LINE): # to print out 7 values of dif
    index = index + 1
    print(weights[-1 + index])


# =====> Close the output file
file.close() # closing the file
```

This response earned five marks. This response opens and closes a file, so does have some functionality.

Q06 – 2D linear search

Many responses were able to search the data structure for an identified word, but were not able to identify when a location was passed over or if the item was not in the data structure.

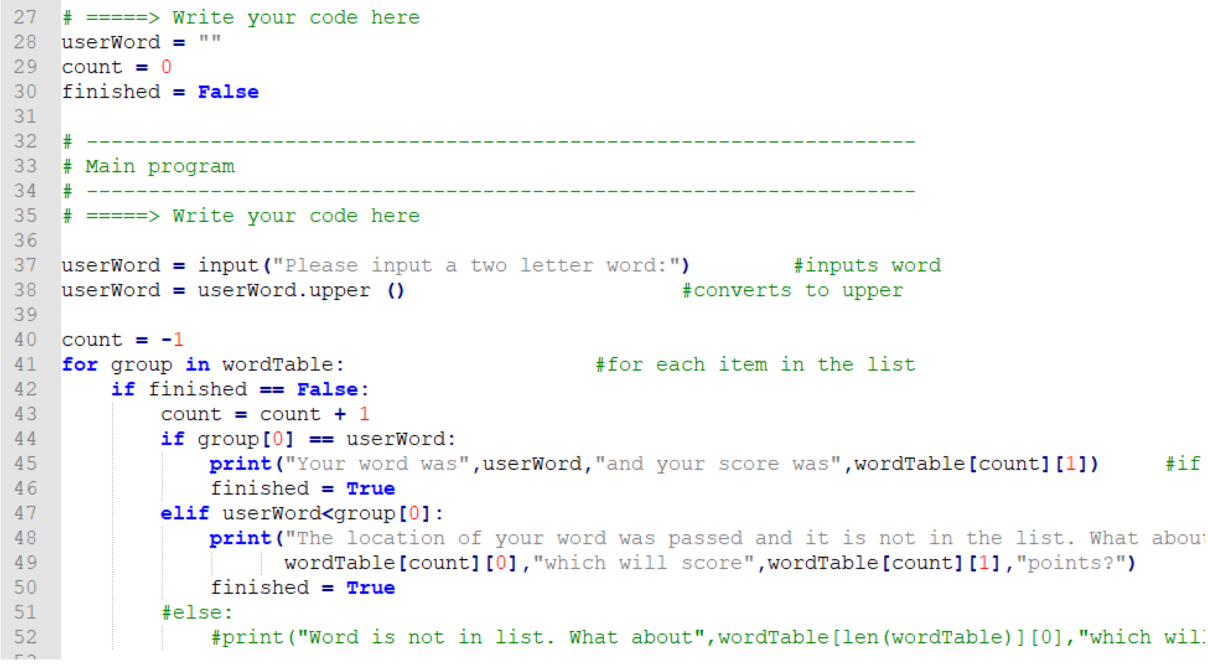
Some responses included mechanisms for correctly tracking the suggested word, although many tried to just suggest the next word along, with no consideration if it was outside the end of the array. This led to runtime errors. Conversely, other responses always suggested the last word in the array, regardless if it were the correct suggestion.

There were some examples of confusion with string comparisons. Some responses attempted to compare the target word with the tile word, letter by letter, rather than with a simple relational operator over the entire two-letter string. Most of these attempts did not work successfully.

The levels-based mark scheme for design is included in this question. This mark scheme takes into consideration the number of compares in loop passes. Some responses that correctly used a loop to process the data, used a selection to check if the item was the last in the list, as well. Other responses used an iterative loop and processed every item in the data, ignoring the requirements for an early exit.

The levels-based mark scheme for programming practice is included in this question. The majority of responses received two of these marks. A few responses missed out on the third mark because of the excessive amounts of comments. For a band three, the response should include effective commenting used to explain logic of code blocks and code that is clear, with good use of white space to aid readability. Excessive commenting makes the code less readable, rather than more readable.

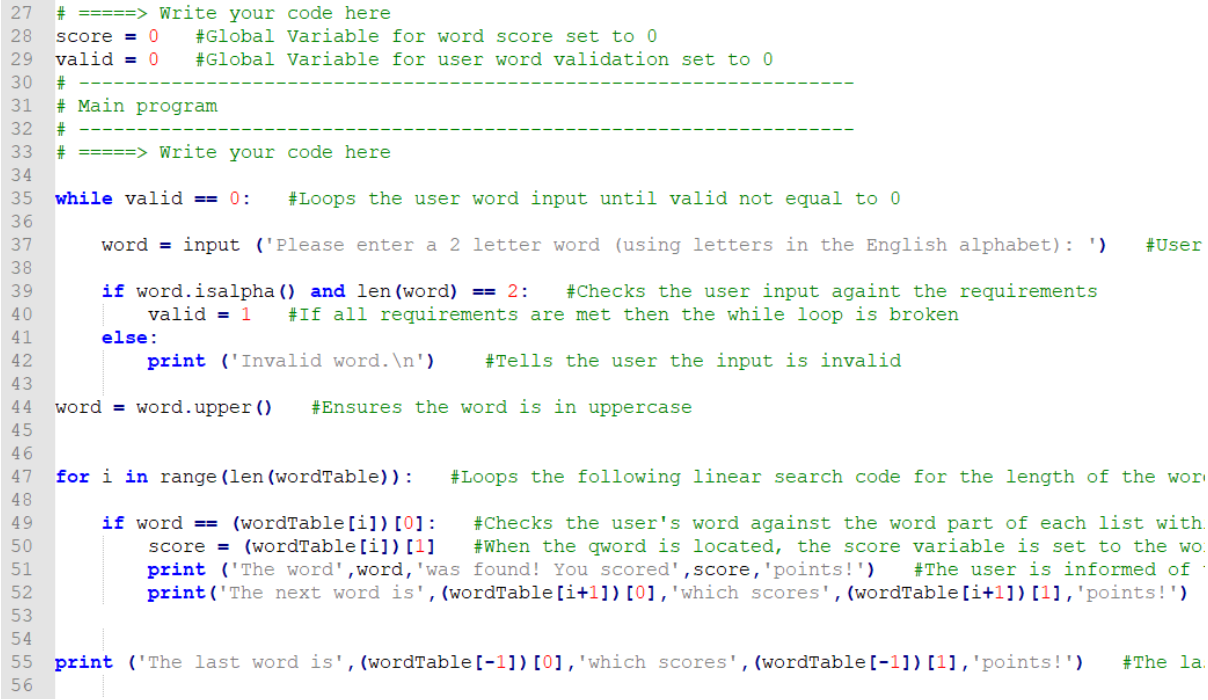
Q6 Example 1



```
# =====> Write your code here
userWord = ""
count = 0
finished = False

# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------
# =====> Write your code here

userWord = input("Please input a two letter word:")        #inputs word
userWord = userWord.upper ()                      #converts to upper

count = -1
for group in wordTable:                    #for each item in the list
    if finished == False:
        count = count + 1
        if group[0] == userWord:
            print("Your word was",userWord,"and your score was",wordTable[count][1])     #if
            finished = True
        elif userWord<group[0]:
            print("The location of your word was passed and it is not in the list. What abou
                  wordTable[count][0],"which will score",wordTable[count][1],"points?")
            finished = True
        #else:
            #print("Word is not in list. What about",wordTable[len(wordTable)][0],"which wil
```

This response was awarded 11 marks. This is a good example demonstrating locating the target in the list and identifying if the target location was passed over. The requirement for an early exit has not been met.

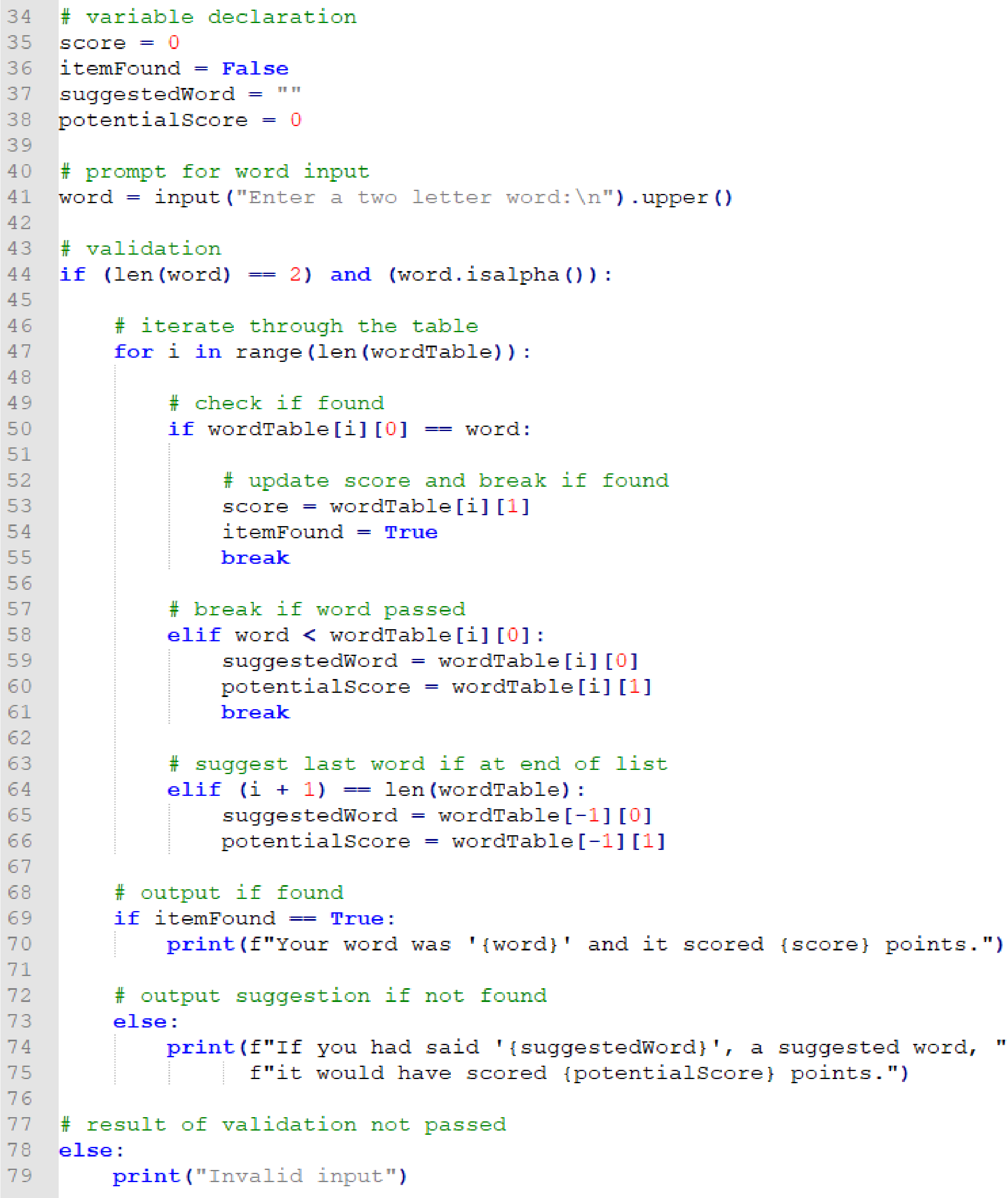
Q6 Example 2



```
27  # =====> Write your code here
28  score = 0    #Global Variable for word score set to 0
29  valid = 0    #Global Variable for user word validation set to 0
30  # ------------------------------------------------------------------
31  # Main program
32  # ------------------------------------------------------------------
33  # =====> Write your code here
34
35  while valid == 0:   #Loops the user word input until valid not equal to 0
36
37      word = input ('Please enter a 2 letter word (using letters in the English alphabet): ')   #User
38
39      if word.isalpha() and len(word) == 2:   #Checks the user input againt the requirements
40          valid = 1   #If all requirements are met then the while loop is broken
41      else:
42          print ('Invalid word.\n')    #Tells the user the input is invalid
43
44  word = word.upper()   #Ensures the word is in uppercase
45
46
47  for i in range(len(wordTable)):   #Loops the following linear search code for the length of the wor
48
49      if word == (wordTable[i])[0]:   #Checks the user's word against the word part of each list with
50          score = (wordTable[i])[1]   #When the qword is located, the score variable is set to the wo
51          print ('The word',word,'was found! You scored',score,'points!')   #The user is informed of
52          print('The next word is',(wordTable[i+1])[0],'which scores',(wordTable[i+1])[1],'points!')
53
54
55  print ('The last word is',(wordTable[-1])[0],'which scores',(wordTable[-1])[1],'points!')   #The la
56
```

This response was awarded eight marks. This is a good attempt at solving an unseen problem. The response demonstrates a loop and comparison for matching items, as required in a solution. As validation of input was not a requirement of the solution, no marks were awarded for lines 35 to 42, whether or not they function correctly.

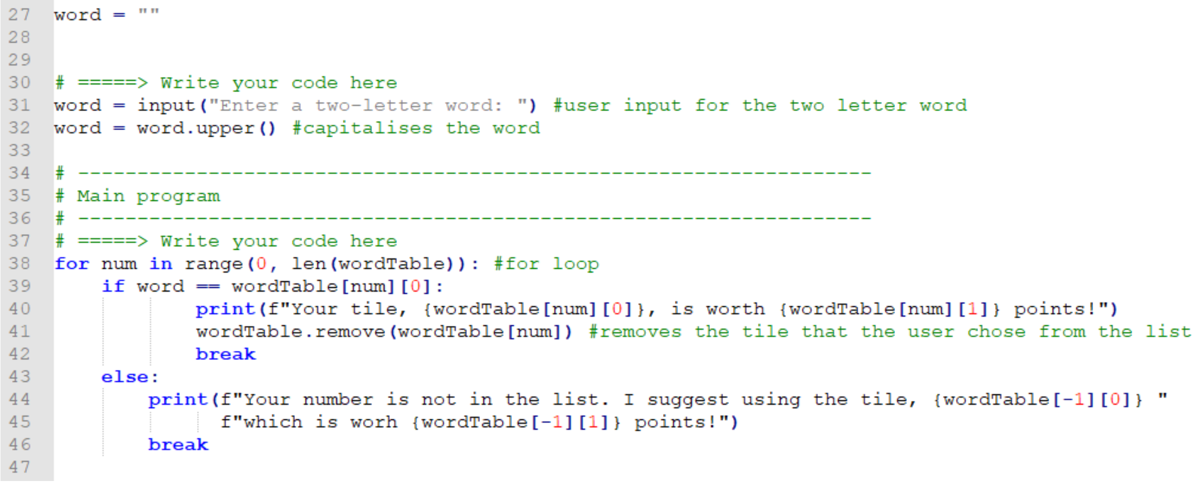
Q6 Example 3



```
34  # variable declaration
35  score = 0
36  itemFound = False
37  suggestedWord = ""
38  potentialScore = 0
39
40  # prompt for word input
41  word = input("Enter a two letter word:\n").upper()
42
43  # validation
44  if (len(word) == 2) and (word.isalpha()):
45
46      # iterate through the table
47      for i in range(len(wordTable)):
48
49          # check if found
50          if wordTable[i][0] == word:
51
52              # update score and break if found
53              score = wordTable[i][1]
54              itemFound = True
55              break
56
57          # break if word passed
58          elif word < wordTable[i][0]:
59              suggestedWord = wordTable[i][0]
60              potentialScore = wordTable[i][1]
61              break
62
63          # suggest last word if at end of list
64          elif (i + 1) == len(wordTable):
65              suggestedWord = wordTable[-1][0]
66              potentialScore = wordTable[-1][1]
67
68      # output if found
69      if itemFound == True:
70          print(f"Your word was '{word}' and it scored {score} points.")
71
72      # output suggestion if not found
73      else:
74          print(f"If you had said '{suggestedWord}', a suggested word, "
75                f"it would have scored {potentialScore} points.")
76
77  # result of validation not passed
78  else:
79      print("Invalid input")
```

This response was awarded 15 marks. This is a good example of a well-designed, coded, and functional solution. The validation on line 44 is not required. Comments have been used to explain the logic, but are not excessive.

Q6 Example 4



```
word = ""


# =====> Write your code here
word = input("Enter a two-letter word: ") #user input for the two letter word
word = word.upper() #capitalises the word

# ------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------
# =====> Write your code here
for num in range(0, len(wordTable)): #for loop
    if word == wordTable[num][0]:
            print(f"Your tile, {wordTable[num][0]}, is worth {wordTable[num][1]} points!")
            wordTable.remove(wordTable[num]) #removes the tile that the user chose from the list
            break
    else:
        print(f"Your number is not in the list. I suggest using the tile, {wordTable[-1][0]} "
              f"which is worh {wordTable[-1][1]} points!")
        break

```

This response was awarded five marks. It demonstrates the use of a loop and selection, as required in a solution. Although the solution is not functional, it demonstrates that marks can be earned by attempting a solution.

Summary

Students should:

- Attempt every question in the paper.
- Follow the instructions in the paper and do not rewrite the supplied code.
- Remove all the syntax errors from ode so that it will translate.
- Execute and test code with the data supplied in the question.
- Use effective, but not excessive, commenting and white space to make the program logic clear.