

# Pearson Edexcel Level 1/Level 2 GCSE (9–1)

Sample assessment material for first teaching  
September 2020

Time: 2 hours

Paper Reference **1CP2/02**

## Computer Science

### Paper 2: Application of Computational Thinking

#### You must have:

- a computer workstation with appropriate programming language code editing software and tools, including an IDE that you are familiar with which shows line numbers
- a 'STUDENT CODING' folder containing code and data files
- printed and electronic copies of the Programming Language Subset (PLS) document.

#### Instructions

- Answer all the questions on your computer.
- Save new or amended code using the file name provided and place it in the 'COMPLETED CODING' folder.
- You must **not** use the internet at any time during the examination.

#### Information

- The 'STUDENT CODING' folder in your user area includes all the code and data files you need.
- The total mark for this paper is 75.
- The marks for each question are shown in brackets.

#### Advice

- Read each question carefully before you start to answer it.
- Save your work regularly.
- Check your answers and work if you have time at the end.

S67286A

©2020 Pearson Education Ltd.

1/1/1/1



Turn over ►

**Answer ALL questions.**

**Suggested time: 10 minutes**

**1** A program simulates the roll of a dice. The program uses a random number generator to create a random integer, between 1 and 6, to represent the roll.

Open file **Q01**.

Amend the code to add or complete lines to:

- import the random library
- create one variable
- create one constant
- assign the result of a library call to a variable
- display a message and the contents of a variable on the screen.

Do **not** add any additional functionality.

Save your amended code file as **Q01FINISHED.py**

---

**(Total for Question 1 = 7 marks)**

**Suggested time: 20 minutes**

2 A programmer has started to write a program, but it does not work correctly. The program should ask the user, "Would you like me to sing?". The user then responds 'y' or 'n'. If the user types 'y', then the computer displays the lines of a song.

Open file **Q02**.

Amend the code to:

- fix the syntax error on line 13
- fix the syntax error on line 15
- fix the syntax error on line 17
- change the identifier x to a more meaningful name
- display a suitable question for the user
- accept the user's input of 'y' or 'n' (no validation is required)
- add a comment to explain the effect of the range function's last parameter, set to -1
- add two uses of white space to aid readability.

Do **not** add any additional functionality.

Save your amended code file as **Q02FINISHED.py**

---

**(Total for Question 2 = 10 marks)**

**Suggested time: 20 minutes**

**3** A program is needed that must meet the following requirements:

- accept only numbers from 1 to 20, inclusive
- print 'invalid input' if the input is outside the permitted range. You can assume only numbers will be entered
- print the input number followed by 'is even', if the number is even
- print the input number followed by 'is odd', if the number is odd.

Open file **Q03**.

Amend the code to:

- fix the runtime error caused by inputting a valid number such as 4 (no other validation is required)
- fix the logic errors that cause incorrect or no outputs for acceptable numbers from 1 to 20, inclusive
- fix the logic error that causes the program to execute, even if a number outside the permitted range is entered
- make all output messages match the requirements listed above.

Do **not** add any additional functionality.

Save your amended code file as **Q03FINISHED.py**

---

**(Total for Question 3 = 13 marks)**

**Suggested time: 20 minutes**

**4** A program takes a year group as input and outputs the stage of education the year group belongs to.

The program loops continually until the user inputs 0 to stop the program.

Input	Output
0	Exits program
1, 2, 3, 4, 5, 6	Primary
7, 8, 9, 10, 11	Secondary
12, 13	College

The lines of code in the program are mixed up.

Open file **Q04**.

Amend the code to make the program work and produce the correct output.  
You will need to rearrange the lines.

Use comments, white space, indentation and layout to make the program easier to read and understand.

Do **not** change the functionality of the given lines of code.

Do **not** add any additional functionality.

Save your amended code file as **Q04FINISHED.py**

**(Total for Question 4 = 15 marks)**

**Suggested time: 25 minutes**

5 Students are collecting data about the amount of water needed to fill different sized paper cones. Their measurements are compared to a calculated volume.

The formula to calculate the volume of a cone is:

$$V = \frac{1}{3} \pi r^2 h$$

- $V$  is volume
- $\pi$  is the constant Pi
- $r$  is the radius of the base of the cone
- $h$  is the height of the cone.

A program and subprogram have been started to carry out the calculation.

Open file **Q05**.

Amend the program and subprogram to meet the following requirements:

- the subprogram must work for any values of radius and height passed as parameters. You can assume values passed to the subprogram will always be numbers. No validation is required
- the subprogram must calculate the volume based on the input parameters
- the main program must print the volume, formatted to show three decimal places (e.g. 16.135).

Do **not** add any additional functionality.

Save your amended code as **Q05FINISHED.py**

**(Total for Question 5 = 15 marks)**

**Suggested time: 25 minutes**

**6** A program is needed to authenticate system logins consisting of a login name and a four-digit passcode.

These items for current users are stored in a two-dimensional list with 19 records. The list contains user number, last name, first name, login name and passcode.

The list is sorted by login name. All users have passcodes that are integers between 1000 and 9999.

Open file **Q06**.

Write a program to meet the following requirements:

### Inputs

- Prompt for and accept a login name, no validation required
- Prompt for and accept a four-digit passcode. You can assume only numbers will be entered
- Ensure the passcode is between 1000 and 9999, inclusive

### Process

- Work with any number of users in the list
- Use a linear search to find the record with the correct combination of login name and passcode
- Stop searching when the location of where the record should have been found is passed. For example:
  - if looking for 'Jam118' and find 'Joy116', then the record is not in the list, so the search can stop
- Does not need to loop continuously

### Outputs

- Display a welcome message, including the user's first and last names, if the login name and passcode are found
- Display an invalid input message, if the login name and passcode are not found.

Do **not** add any additional functionality.

Use comments, white space and layout to make the program easier to read and understand.

Save your amended code as **Q06FINISHED.py**

---

**(Total for Question 6 = 15 marks)**

---

**TOTAL FOR PAPER = 75 MARKS**

## Paper 2 mark scheme

Please note the following regarding the application of this mark scheme:

- Pearson have developed the Programming Language Subset (PLS) document to specify which parts of Python 3 students are required to learn in order to access the assessments. Pearson will not go beyond the scope of the PLS in its assessments. However, if students use alternative valid techniques to answer questions in the assessments, they will also be credited.
- At the time of publication, this qualification uses Python 3 as the underlying programming paradigm for the assessment and teaching and learning of the qualification. Python 2 will not be supported and should not be used in the delivery of this qualification. If at some time in the future a Python 4 is released, we will work closely with our centres before deciding whether to remain with Python 3 or to move to the newer version of the language.
- Google are developing some Google Python library equivalents (gPython) that might be required for Chrome-based schools. Centres that avail of these Google library functions will also be credited.
- The AO breakdown is shown for paper 2 to give greater clarity to the skills demonstrated in each marking point. This is necessary for this practical paper as the majority of the marking points are interlinked in a way that is more complex than in a traditional written paper.

Question number	Answer	Additional guidance	Mark
<b>1</b>	<p>The following assessment objectives are assessed:</p> <ul style="list-style-type: none"><li>• AO2.1b</li><li>• AO3.2b</li></ul> <p>Award marks as shown.</p> <ul style="list-style-type: none"><li>• import random (1)</li><li>• roll = (1) &lt;integer&gt; 0 (1)</li><li>• SIDES = (1) &lt;integer&gt; 6 (1)</li><li>• roll = random.randint(1, SIDES) (1)</li><li>• print string (1) and roll (1)</li></ul>	<ul style="list-style-type: none"><li>• Output of print statements may appear on separate lines</li><li>• Items in '&lt; &gt;' do not form part of the response, but provide clarification</li></ul>	<b>(7)</b>

```
1 # -----
2 # Import libraries
3 # -----
4
5 # ===> Complete this line to import the random library
6 import random
7
8 # -----
9 # Global variables
10 # -----
11
12 # ===> Create an integer variable named roll and set it to 0
13 roll = 0
14
15 # ===> Create a constant named SIDES and set it to 6
16 SIDES = 6
17
18 # -----
19 # Main program
20 # -----
21
22 # ===> Assign the result of this library call to the variable roll
23 roll = random.randint(1, SIDES)
24
25 # ===> Display the message "Your roll is" and the variable roll
26 print ("Your roll is", roll)
27
```

Question number	Answer	Additional guidance	Mark
<b>2</b>	<p>The following assessment objectives are assessed:</p> <ul style="list-style-type: none"> <li>• AO2.1a</li> <li>• AO2.1b</li> <li>• AO3.1</li> <li>• AO3.2a</li> <li>• AO3.2b</li> <li>• AO3.2c</li> </ul> <p>Award marks as shown.</p> <ul style="list-style-type: none"> <li>• Add ':' at end of the line: <code>if (choice == 'y'): (1)</code></li> <li>• Add missing ')' before ':' in the line: <code>for num in range(5, -1, -1): (1)</code></li> <li>• Add missing " before end bracket in the line: <code>print("Goodbye") (1)</code></li> <li>• Printing a suitable question for the user based on context, i.e. "Do you want me to sing?" (1)</li> <li>• Accept user input of 'y' and 'n' (1)</li> <li>• Changing the variable name 'x' to a more meaningful name (1) such as 'choice' throughout the code</li> <li>• Addition of comment indicating reverse stepping (1)</li> <li>• One mark each for insertion of white space to aid readability, up to a maximum of two marks (2)</li> <li>• Correct output for 'y' (count down 5 to 0 and then Goodbye) and correct output for 'n' (Goodbye) (1)</li> </ul>		<b>(10)</b>

```
1 # -----
2 # Global variables
3 # -----
4
5 # ===> Change the identifier x to a more meaningful name
6 choice = ""
7
8 # -----
9 # Main program
10 # -----
11 # ===> Display a suitable question to the user
12 print ("Would you like me to sing?")
13
14 # ===> Accept the user's input (no validation required)
15 choice = input("Choose 'y' for yes and 'n' for no")
16
17 if (choice == 'y'):
18     # ===> Add a comment to explain the effect of the last -1 in this call
19     # Counting backwards by using step as -1
20     for num in range(5, -1, -1):
21         print(num, "green bottles sitting on the wall")
22
23 print("Goodbye")
24
```

Question number	Answer	Additional guidance	Mark
<b>3</b>	<p>The following assessment objectives are assessed:</p> <ul style="list-style-type: none"> <li>• AO2.1b</li> <li>• AO3.1</li> <li>• AO3.2a</li> <li>• AO3.2b</li> <li>• AO3.2c</li> </ul> <p>Award marks as shown.</p> <ul style="list-style-type: none"> <li>• Fixing runtime error by coercion of input to 'int' (1)</li> <li>• Fixing errors by using modulus (1)</li> <li>• Use of at least one appropriate 'if' statement in the solution (1)</li> <li>• Adding validation for input numbers using: <ul style="list-style-type: none"> <li>◦ relational operator (<math>\leq 20</math>) (1)</li> <li>◦ relational operator (<math>\geq 1</math>) (1)</li> <li>◦ correct Boolean operator (and/or) (1)</li> </ul> </li> <li>• Corrects output message for even numbers and odd numbers (1)</li> </ul> <p>Levels-based mark scheme to a maximum of 6, from:</p> <ul style="list-style-type: none"> <li>• Solution design (3)</li> <li>• Functionality (3)</li> </ul>	<ul style="list-style-type: none"> <li>• Fixing error with odd numbers can be done in several different ways (see examples)</li> <li>• Award any accurate tests for validation range</li> </ul> <p>Considerations:</p> <ul style="list-style-type: none"> <li>• 6.1.6 Using test data to evaluate a program, such as extreme data [a character], normal data [1...20] and boundary data [0, 21]</li> <li>• 6.2.2 Appropriate use of sequencing, selection and repetition</li> <li>• 6.1.1 Use analysis to solve problems</li> <li>• 6.1.6 Use logical reasoning to evaluate efficiency (i.e. reduce tests)</li> </ul>	<b>(13)</b>

## Solution design (levels-based mark scheme)

0	1	2	3	Max.
<i>No rewardable material</i>	<ul style="list-style-type: none"> <li>• There has been little attempt to decompose the problem.</li> <li>• Some of the component parts of the problem can be seen in the solution, although this will not be complete.</li> <li>• Some parts of the logic are clear and appropriate to the problem.</li> <li>• The use of variables and data structures, appropriate to the problem, is limited.</li> <li>• The choice of programming constructs, appropriate to the problem, is limited.</li> </ul>	<ul style="list-style-type: none"> <li>• There has been some attempt to decompose the problem.</li> <li>• Most of the component parts of the problem can be seen in the solution.</li> <li>• Most parts of the logic are clear and appropriate to the problem.</li> <li>• The use of variables and data structures is mostly appropriate.</li> <li>• The choice of programming constructs is mostly appropriate to the problem.</li> </ul>	<ul style="list-style-type: none"> <li>• The problem has been decomposed clearly into component parts.</li> <li>• The component parts of the problem can be seen clearly in the solution.</li> <li>• The logic is clear and appropriate to the problem.</li> <li>• The choice of variables and data structures is appropriate to the problem.</li> <li>• The choice of programming constructs is accurate and appropriate to the problem.</li> </ul>	<b>3</b>

## Functionality (levels-based mark scheme)

0	1	2	3	Max.
<i>No rewardable material</i>	<p><b>Functionality (when the code is run)</b></p> <ul style="list-style-type: none"> <li>• The component parts of the program are incorrect or incomplete, providing a program of limited functionality that meets some of the given requirements.</li> <li>• Program outputs are of limited accuracy and/or provide limited information.</li> <li>• Program responds predictably to some of the anticipated input.</li> <li>• Solution is not robust and may crash on anticipated or provided input.</li> </ul>	<p><b>Functionality (when the code is run)</b></p> <ul style="list-style-type: none"> <li>• The component parts of the program are complete, providing a functional program that meets most of the stated requirements.</li> <li>• Program outputs are mostly accurate and informative.</li> <li>• Program responds predictably to most of the anticipated input.</li> <li>• Solution may not be robust within the constraints of the problem.</li> </ul>	<p><b>Functionality (when the code is run)</b></p> <ul style="list-style-type: none"> <li>• The component parts of the program are complete, providing a functional program that fully meets the given requirements.</li> <li>• Program outputs are accurate, informative, and suitable for the user.</li> <li>• Program responds predictably to anticipated input.</li> <li>• Solution is robust within the constraints of the problem.</li> </ul>	<b>3</b>

```
1 # -----
2 # Global variables
3 # -----
4 num = 0
5
6 # -----
7 # Main program
8 # -----
9
10 # ----- Solution 1 -----
11 # Coerce the input from string to integer
12 num = int (input ("Please enter a number (1 ... 20)"))
13
14 # Check for valid input numbers
15 if (num >=1 and num <= 20):
16     if (num % 2 == 0):
17         print (num, "is even")
18     else:
19         # If num is not even, then it must be odd
20         print (num, "is odd")
21 else:
22     print ("Invalid input")      # Error message for bad input
23
```

```
24 # ----- Solution 2 -----
25 num = int (input ("Please enter a number (1 ... 20)"))
26
27 if (num < 1):
28     print ("Bad input")
29 elif (num > 20):
30     print ("Bad input")
31 elif (num % 2 == 0):
32     print (num, "is even")
33 else:
34     print (num, "is odd")
35
36 # ----- Solution 3 -----
37 num = int (input ("Please enter a number (1 ... 20)"))
38
39 if (num >= 1) and (num <= 20):
40     if (num % 2 == 0):
41         print (num, " is even")
42     elif (num %2 != 0):
43         # Extra check for odd
44         print (num, " is odd")
45 else:
46     print ("Bad input")      # Error message for the user
47
48 # ----- Solution 4 -----
49 num = int (input ("Please enter a number (1 ... 20)"))
50
51 # Keeps looping until a good input is identified
52 while (num < 1) or (num > 20):
53     print ("Invalid input")
54     num = int (input ("Please enter a number (1 ...20)"))
55
56 if (num % 2 == 0):
57     print (num, "is even")
58 else:
59     print (num, "is odd")
60
```

Question number	Answer	Additional guidance	Mark
<b>4</b>	<p>The following assessment objectives are assessed:</p> <ul style="list-style-type: none"> <li>• AO2.1b</li> <li>• AO3.1</li> <li>• AO3.2a</li> <li>• AO3.2b</li> <li>• AO3.2c</li> </ul> <p>Award marks as shown.</p> <ul style="list-style-type: none"> <li>• Use of comments, white space and layout to aid readability (1)</li> <li>• Initial input done outside loop, to handle first entry is '0' (1)</li> <li>• Repetition (while) used as outermost loop (1)</li> <li>• 'elif (year &gt; 13)' is placed later in the logic than 'if (year &lt; 1)' (1)</li> <li>• 'elif (year &lt; 12)' is placed later in the logic than 'elif (year &lt; 7)' (1)</li> <li>• Accepting next round of input done inside loop (1)</li> <li>• Validation messages match validation tests: <ul style="list-style-type: none"> <li>◦ Year too small (1)</li> <li>◦ Year too big (1)</li> </ul> </li> <li>• Institution messages match tests: <ul style="list-style-type: none"> <li>◦ Primary (1)</li> <li>◦ Secondary (1)</li> <li>◦ College (1)</li> </ul> </li> <li>• Correct outputs for each set of test data: <ul style="list-style-type: none"> <li>◦ 0 = exiting (1)</li> <li>◦ 1 and 6 = Primary (1)</li> <li>◦ 7 and 11 = Secondary (1)</li> <li>◦ 12 = College (1)</li> </ul> </li> </ul>		<b>(15)</b>

```
1 # -----
2 # Global variables
3 #
4 year = 0           # Do not move this line
5 strYear = ""        # Do not move this line
6
7 #
8 # Main program
9 #
10
11 # Put the lines into the correct order to solve the problem.
12 # A user types in a year group. The program indicates which stage
13 # of education the year group belongs to. The program loops until
14 # the user enters 0.
15 # Example:
16 # Input           Output
17 #
18 # 0           Exits program
19 # 1, 2, 3, 4, 5, 6 Primary
20 # 7, 8, 9, 10, 11 Secondary
21 # 12, 13      College
22
23 # ----- Solution 1 -----
24 # Prime the loop, just in case the first entry is '0'
25 strYear = input ("Enter year group (1 to 13, 0 to exit)")
26 year = int (strYear)
27
28 # Keep looping until user wants to stop
29 while (year != 0):
30     # Validate input as a real year group
31     if (year < 1):
32         print ("Year too small")
33     elif (year > 13):
34         print ("Year too big")
35     elif (year < 7):
36         print ("Primary")
37     elif (year < 12):
38         print ("Secondary")
39     else:
40         print ("College")
41
42     # Get a new input before going to top of loop
43     strYear = input("Enter year group (1 to 13, 0 to exit)")
44     year = int(strYear)
45
```

```
46 # ----- Solution 2 -----
47 # Prime the loop, just in case the first entry is '0'
48 strYear = input ("Enter year group (1 to 13, 0 to exit)")
49 year = int (strYear)
50
51 # Keep looping until user wants to stop
52 while (year != 0):
53     # Validate input as a real year group
54     if (year < 1):
55         print ("Year too small")
56     elif (year < 7):
57         print ("Primary")
58     elif (year < 12):
59         print ("Secondary")
60     elif (year > 13):
61         print("Year too big")
62     else:
63         print ("College")
64
65     # Get a new input before going to top of loop
66     strYear = input("Enter year group (1 to 13, 0 to exit)")
67     year = int(strYear)
68
```

Question number	Answer	Additional guidance	Mark
<b>5</b>	<p>The following assessment objectives are assessed:</p> <ul style="list-style-type: none"> <li>• AO2.1b</li> <li>• AO3.1</li> <li>• AO3.2a</li> <li>• AO3.2b</li> <li>• AO3.2c</li> </ul> <p>Award marks as shown.</p> <ul style="list-style-type: none"> <li>• Import of math library (1)</li> <li>• Two parameters in first line of subprogram definition (1) with names 'pRadius' and 'pHeight', in any order (1)</li> <li>• Accurate translation of the formula to code (1)</li> <li>• Use of math.pi constant in formula translation (1)</li> <li>• Two passed-in parameters ('pRadius' and 'pHeight') used in the calculation (1)</li> <li>• Assignment of calculation to 'theVolume' (1)</li> <li>• One return statement with 'theVolume' in brackets (1)</li> <li>• Parameters in call to subprogram are 'baseRadius' and 'coneHeight', in any order (1)</li> <li>• Order of parameters matches order in first line of subprogram definition (1)</li> <li>• Capture of returned value in main program, in 'coneVolume' (1)</li> <li>• Format volume to three decimal places for outputting only (1)</li> </ul> <p>Levels-based mark scheme to a maximum of 3, from:</p> <ul style="list-style-type: none"> <li>• Functionality (3)</li> </ul>	<p>Considerations:</p> <ul style="list-style-type: none"> <li>• 6.1.1 Be able to use decomposition to analyse requirements</li> <li>• 6.1.2 Be able to write in a high-level language</li> <li>• 6.6.1 Be able to perform generalisations</li> <li>• Default printing will drop trailing 0s, even if rounded, so string formatting should be used</li> </ul>	<b>(15)</b>

## Functionality (levels-based mark scheme)

0	1	2	3	Max.
<i>No rewardable material</i>	<p><b>Functionality (when the code is run)</b></p> <ul style="list-style-type: none"> <li>• The component parts of the program are incorrect or incomplete, providing a program of limited functionality that meets some of the given requirements.</li> <li>• Program outputs are of limited accuracy and/or provide limited information.</li> <li>• Program responds predictably to some of the anticipated input.</li> <li>• Solution is not robust and may crash on anticipated or provided input.</li> </ul>	<p><b>Functionality (when the code is run)</b></p> <ul style="list-style-type: none"> <li>• The component parts of the program are complete, providing a functional program that meets most of the stated requirements.</li> <li>• Program outputs are mostly accurate and informative.</li> <li>• Program responds predictably to most of the anticipated input.</li> <li>• Solution may not be robust within the constraints of the problem.</li> </ul>	<p><b>Functionality (when the code is run)</b></p> <ul style="list-style-type: none"> <li>• The component parts of the program are complete, providing a functional program that fully meets the given requirements.</li> <li>• Program outputs are accurate, informative, and suitable for the user.</li> <li>• Program responds predictably to anticipated input.</li> <li>• Solution is robust within the constraints of the problem.</li> </ul>	<b>3</b>

```
1 # -----
2 # Import libraries
3 # -----
4 # ===> Import a library to use Pi
5 import math
6
7 # -----
8 # Global variables
9 # -----
10
11 # Hard coded for testing
12 coneHeight = 10.7
13 baseRadius = 1.2
14 coneVolume = 0.0
15
16 # -----
17 # Subprograms
18 # -----
19 # ===> Add parameters inside the brackets
20 def calcVolume (pRadius, pHeight):
21
22     print ("The radius is:", pRadius)
23     print ("The height is:", pHeight)
24
25     # ===> Complete the calculation for the volume
26     theVolume = 1/3 * math.pi * math.pow (pRadius, 2) * pHeight
27     # theVolume = 1/3 * math.pi * pRadius**2 * pHeight
28     # theVolume = 1/3 * math.pi * pRadius * pRadius * pHeight
29
30     print ("The volume is:", theVolume)
31
32     # ===> Return the volume to the caller
33     return (theVolume)
34
35 # -----
36 # Main program
37 # -----
38
39 # ===> Call the subprogram, passing parameters,
40 #       and catch the returned value in the correct variable
41 coneVolume = calcVolume (baseRadius, coneHeight)
42
43 # ===> Print the total volume to three decimal places using string.format()
44 # ===> by completing the pattern inside the { }
45 print ("{:.3f}".format(coneVolume))
46
```

Question number	Answer	Additional guidance	Mark
<b>6</b>	<p>The following assessment objectives are assessed:</p> <ul style="list-style-type: none"> <li>• AO2.1b</li> <li>• AO3.1</li> <li>• AO3.2a</li> <li>• AO3.2b</li> <li>• AO3.2c</li> </ul> <p>Award marks as shown.</p> <p>Points-based mark scheme:</p> <p><b>Inputs</b></p> <ul style="list-style-type: none"> <li>• Accepts and responds to user input (1)</li> <li>• Validation with range check using relational operators <math>\geq 1000, \leq 9999</math> (1)</li> </ul> <p><b>Process</b></p> <ul style="list-style-type: none"> <li>• Use of library subprograms <code>len()</code> (1) to work with any number of users in the list</li> <li>• Use of Boolean (1) to stop loop when found or passed over</li> <li>• Use of 2-dimensional indexing (1) in user list</li> </ul> <p><b>Outputs</b></p> <ul style="list-style-type: none"> <li>• Display of appropriate messages (1)</li> </ul> <p>Levels-based mark scheme to a maximum of 9, from:</p> <ul style="list-style-type: none"> <li>• Solution design (3)</li> <li>• Good programming practices (3)</li> <li>• Functionality (3)</li> </ul>	<p>Considerations:</p> <ul style="list-style-type: none"> <li>• 6.1.1 Use decomposition and abstraction to analyse a problem (inputs, outputs, processing, initialisation, design)</li> <li>• 6.6.1 Decompose into subproblems</li> <li>• 6.1.2 Write in a high-level language</li> <li>• 6.2.2 Use sequencing and selection components</li> </ul>	<b>(15)</b>

## Solution design (levels-based mark scheme)

0	1	2	3	Max.
<i>No rewardable material</i>	<ul style="list-style-type: none"> <li>• There has been little attempt to decompose the problem.</li> <li>• Some of the component parts of the problem can be seen in the solution, although this will not be complete.</li> <li>• Some parts of the logic are clear and appropriate to the problem.</li> <li>• The use of variables and data structures, appropriate to the problem, is limited.</li> <li>• The choice of programming constructs, appropriate to the problem, is limited.</li> </ul>	<ul style="list-style-type: none"> <li>• There has been some attempt to decompose the problem.</li> <li>• Most of the component parts of the problem can be seen in the solution.</li> <li>• Most parts of the logic are clear and appropriate to the problem.</li> <li>• The use of variables and data structures is mostly appropriate.</li> <li>• The choice of programming constructs is mostly appropriate to the problem.</li> </ul>	<ul style="list-style-type: none"> <li>• The problem has been decomposed clearly into component parts.</li> <li>• The component parts of the problem can be seen clearly in the solution.</li> <li>• The logic is clear and appropriate to the problem.</li> <li>• The choice of variables and data structures is appropriate to the problem.</li> <li>• The choice of programming constructs is accurate and appropriate to the problem.</li> </ul>	<b>3</b>

## Good programming practices (levels-based mark scheme)

0	1	2	3	Max.
<i>No rewardable material</i>	<ul style="list-style-type: none"> <li>• There has been little attempt to lay out the code into identifiable sections to aid readability.</li> <li>• Some use of meaningful variable names.</li> <li>• Limited or excessive commenting.</li> <li>• Parts of the code are clear, with limited use of appropriate spacing and indentation.</li> </ul>	<ul style="list-style-type: none"> <li>• There has been some attempt to lay out the code to aid readability, although sections may still be mixed.</li> <li>• Uses mostly meaningful variable names.</li> <li>• Some use of appropriate commenting, although may be excessive.</li> <li>• Code is mostly clear, with some use of appropriate white space to aid readability.</li> </ul>	<ul style="list-style-type: none"> <li>• Layout of code is effective in separating sections, e.g. putting all variables together, putting all subprograms together as appropriate.</li> <li>• Meaningful variable names and subprogram interfaces are used where appropriate.</li> <li>• Effective commenting is used to explain logic of code blocks.</li> <li>• Code is clear, with good use of white space to aid readability.</li> </ul>	<b>3</b>

## Functionality (levels-based mark scheme)

0	1	2	3	Max.
<i>No rewardable material</i>	<p><b>Functionality (when the code is run)</b></p> <ul style="list-style-type: none"> <li>• The component parts of the program are incorrect or incomplete, providing a program of limited functionality that meets some of the given requirements.</li> <li>• Program outputs are of limited accuracy and/or provide limited information.</li> <li>• Program responds predictably to some of the anticipated input.</li> <li>• Solution is not robust and may crash on anticipated or provided input.</li> </ul>	<p><b>Functionality (when the code is run)</b></p> <ul style="list-style-type: none"> <li>• The component parts of the program are complete, providing a functional program that meets most of the stated requirements.</li> <li>• Program outputs are mostly accurate and informative.</li> <li>• Program responds predictably to most of the anticipated input.</li> <li>• Solution may not be robust within the constraints of the problem.</li> </ul>	<p><b>Functionality (when the code is run)</b></p> <ul style="list-style-type: none"> <li>• The component parts of the program are complete, providing a functional program that fully meets the given requirements.</li> <li>• Program outputs are accurate, informative, and suitable for the user.</li> <li>• Program responds predictably to anticipated input.</li> <li>• Solution is robust within the constraints of the problem.</li> </ul>	<b>3</b>

```
1 # -----
2 # Global variables
3 # -----
4
5 # User Number, Last Name, First Name, Login Name, Passcode
6 userList = [[110,"Cashin","Bonnie","Cae110",7005],
7             [101,"Cheruit","Madeleine","Che101",1507],
8             [103,"Chanel","Coco","Chol03",7333],
9             [107,"Gres","Madame","Gre107",3054],
10            [114,"Hamnett","Katharine","Hae114",4807],
11            [118,"Herrera","Carolina","Hea118",5567],
12            [111,"Hulanicki","Barbara","Hua111",5125],
13            [116,"Johnson","Betsey","Joy116",8869],
14            [104,"Lanvin","Jeanne","Lae104",8580],
15            [109,"McCardell","Claire","Mce109",5991],
16            [102,"Paquin","Jeanne","Pae102",6495],
17            [112,"Quant","Mary","Quy112",9028],
18            [113,"Rykiel","Sonia","Rya113",1177],
19            [105,"Schiaparelli","Elsa","Sca105",2980],
20            [108,"Schlee","Valentina","Sca108",6801],
21            [106,"Vionnet","Madeleine","Vie106",9042],
22            [117,"Von Furstenberg","Diane","Voe117",2553],
23            [119,"Wang","Vera","Waa119",2004],
24            [115,"Westwood","Vivienne","Wee115",7806]]
25
26 inID = ""                      # String
27 inPass = 0                       # Integer
28 found = False                    # Haven't found the record yet
29 passed = False                  # Haven't gone past where it should be
30 index = 0                       # The current record being looked at
31
```

```
32 # -----
33 # Main program
34 # -----
35
36 # Get user login name
37 inID = input ("Enter your user login name, type X to exit.")
38
39 # Get user passcode
40 inPass = int (input ("Enter your four digit passcode"))
41
42 # Check if passcode is valid
43 if (inPass >= 1000 and inPass <= 9999):
44     # Look through userList to find matching set
45     while (found == False and passed == False and index < len(userList)):
46         # If both parts match (authenticated), display welcome message
47         if (userList[index][3] == inID and userList[index][4] == inPass):
48             found = True
49             print ("Welcome", userList[index][2], userList[index][1])
50         # Check if have passed over where it should be in the list
51         elif (userList[index][3] > inID):
52             passed = True                      # Stops looping
53         else:
54             index = index + 1                # Look at next entry
55         # If not found or passed, display "Invalid Login Credentials"
56         if (found == False):
57             print ("Invalid Login Credentials")
58 else:
59     print ("Passcode must be four digits long")
60
```