****

Copyright

© 2021 PG Online Limited

The contents of this unit are protected by copyright.

This Sample Paper, Sample Paper Mark Scheme and other associated files distributed with it are supplied to you by PG Online Limited under licence and may be used and copied by you only in accordance with the terms of the licence agreement between you and PG Online Limited. Except as expressly permitted by the licence, no part of the materials distributed with this unit may be used, reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic or otherwise, without the prior written permission of PG Online Limited.

Licence agreement

This is a legal agreement between you, the teaching institution, and PG Online Limited. PG Online Limited grants to you a non-exclusive, non-transferable, revocable licence to use this Sample Paper, Sample Paper Mark Scheme and other associated files distributed with it in the course of teaching by your teachers and/or employees.

The materials distributed with this unit may be copied and used by your teachers and/or employees on a single site only in the course of their teaching. You warrant that you shall not, and shall procure that each of your teachers and/or employees shall not, share in any way any of the materials or part of the materials with any third party, including users on another site or individuals who are teachers and/or employees of a separate institution. You acknowledge and agree that the materials must remain with you, the teaching institution, and no part of the materials may be transferred to another institution. You also warrant that you shall not, and shall procure that each of your teachers and/or employees shall not, procure, authorise, encourage, facilitate or enable any third party to reproduce these materials in whole or in part without the prior permission of PG Online Limited.

In consideration of the licence granted to you, you shall indemnify PG Online Limited against all liabilities, costs, expenses, damages and losses (including but not limited to any direct, indirect or consequential losses, loss of profit, loss of reputation and all interest, penalties and legal costs and all other professional costs and expenses) suffered or incurred by PG Online Limited arising out of or in connection with the exercise by you of your rights granted under this licence.

Please write clearly, in BLOCK CAPITALS and black ink

Centre number Candidate number

Forename(s)

Surname

Date of Exam Time allowed: 2 hours

GCSE Computer Science

Paper 1: Computational thinking and programming skills

Total Marks

PAPER 1B

Instructions

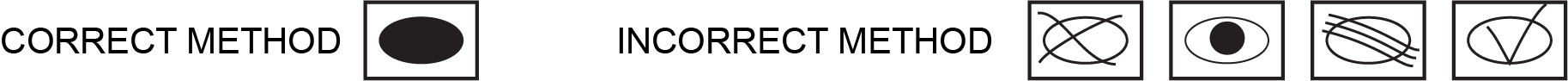
* Write in black ink or black ball-point pen. Use pencil only for drawing.
* Write your answer to each question in the space provided
* Answer all questions
* Do all rough work in this book
* Cross through any work you do not want marked
* **Questions that require a coded solution must be answered in Python 3**
* **You are not allowed to use a calculator**

Information

* The total mark for this paper is **90**
* The student version of this paper has **23** pages

**Advice**

* For multiple-choice questions, completely fill in the lozenge next to the answer you want to select.



* Icon

  Description automatically generatedTo change your answer, cross out your original answer like this: Icon

  Description automatically generated
* If you want to go back to an answer you previously crossed out, circle the answer you now want to select like this:

**Answer ALL questions.**

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **1** | **.** | **1** |

Define the term abstraction.

[2 marks]

Removing irrelevant/unimportant details (1)  
so that attention can be given to the important parts of a problem (1) / to make the problem easier to solve (1).

1 mark for first point, 1 mark for one of the other two points.

Accept different wording with the same meaning.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **2** |  |  |

The pseudo-code in **Figure 1** assigns values to variables.

**Figure 1**

start ← 5  
mult ← start \* start  
mult ← start \* mult

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **2** | **.** | **1** |

Shade **one** lozenge that shows the value stored in the variable start once the program in **Figure 1** has finished running.

[1 mark]

|  |  |  |
| --- | --- | --- |
| **A** | 5 |  |
| **B** | 10 | Icon  Description automatically generated |
| **C** | 25 | Icon  Description automatically generated |
| **D** | 15 | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **2** | **.** | **2** |

Shade **one** lozenge that shows the data type for the variable mult in **Figure 1**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| **A** | Boolean |  |
| **B** | Floating point number | Icon  Description automatically generated |
| **C** | Integer |  |
| **D** | String | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** |  |  |

The algorithm in **Figure 2** is used to calculate the number of car parking spaces left in a car park.

* Line numbers are included but are not part of the algorithm.

**Figure 2**

1 CONSTANT TOTAL\_SPACES ← 200

2 currentSpaces ← TOTAL\_SPACES

3

4 SUBROUTINE CAR\_ENTERS():

5 currentSpaces ← currentSpaces – 1

6 ENDSUBROUTINE

7

8 SUBROUTINE CAR\_EXITS():

9 currentSpaces ← currentSpaces + 1

10 ENDSUBROUTINE

11

12 CAR\_ENTERS()

13 CAR\_ENTERS()

14 CAR\_EXITS()

15

16 IF currentSpaces < 1 THEN

17 OUTPUT "Car park full"

18 ENDIF

19 OUTPUT currentSpaces

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **1** |

Shade **one** lozenge which explains the meaning of the relational operator < in line 16 in **Figure 2**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| **A** | Greater than |  |
| **B** | Greater than or equal to | Icon  Description automatically generated |
| **C** | Less than |  |
| **D** | Less than or equal to | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **2** |

Shade **one** lozenge which shows the number of subroutines present in the program. **Figure 2**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| **A** | 2 |  |
| **B** | 3 | Icon  Description automatically generated |
| **C** | 5 |  |
| **D** | 7 | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **3** |

Shade **one** lozenge which shows the data type of the variable currentSpaces in   
**Figure 2**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| **A** | Character |  |
| **B** | Integer |  |
| **C** | Real |  |
| **D** | String | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **4** |

Shade **one** lozenge which gives the meaning of CONSTANT in line 1 in **Figure 2**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| **A** | The value it stores is able to change while the program is running. |  |
| **B** | The value it stores is not able to change while the program is running. |  |
| **C** | It is not possible to read the value stored while the  program is running |  |
| **D** | The line of code cannot be changed from line 1. | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **5** |

State the line number where selection starts in **Figure 2**.

[1 mark]

Line 16.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **3** | **.** | **6** |

Shade **one** lozenge which gives the output from the program in **Figure 2**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| **A** | Car park full |  |
| **B** | 198 |  |
| **C** | 199 |  |
| **D** | 201 | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **4** |  |  |

A programmer has written a Python program for a hotel’s reception staff to decide whether they should have the front door open or closed.

The door should be closed when either the temperature is less than 15°C or the chance of it raining is more than 10%. Otherwise, the doors should be open. Both the temperature and the chance of rain are entered as a whole number.

Complete the program below by filling in the gaps using the items in **Figure 3**. You will not need to use all the items in **Figure 3**. Each item in **Figure 3** should only be used once.

[5 marks]

**Figure 3**

|  |  |  |  |
| --- | --- | --- | --- |
| if | elif | else: |  |
| and | not | or |  |
| bool | float | int |  |
| < | <= | > | >= |
| : | ) | then |  |

temperature = int(input("Temperature: "))

chanceOfRain = int(input("Chance of rain: "))

if chanceOfRain > 10 or temperature < 15:

print("Close doors")

else:

print("Open doors")

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** |  |  |

The algorithm in **Figure 4** is a searching algorithm.

* Array indexing starts at 0.
* Line numbers are included but are not part of the algorithm.

**Figure 4**

1. arr ← [3, 5, 6, 8, 9, 11, 15, 16, 18]
2. x ← 16
3. mid ← 0
4. low ← 0
5. high ← LEN(arr) - 1
6. found ← False
7. WHILE low <= high AND NOT found:
8. mid ← (high + low) // 2
9. IF arr[mid] < x THEN
10. low ← mid + 1
11. ELSE IF arr[mid] > x:
12. high ← mid - 1
13. ELSE
14. found ← True
15. ENDIF
16. ENDWHILE
17. IF found:
18. OUTPUT mid
19. ELSE
20. OUTPUT "Not found"

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** | **.** | **1** |

State the value of LEN(arr).

[1 mark]

9 (1)

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** | **.** | **2** |

Shade **one** lozenge to show which of the following contains the **false** statement about the algorithm in **Figure 4**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| **A** | The algorithm uses variables |  |
| **B** | The algorithm uses indefinite iteration |  |
| **C** | The algorithm uses a 2D array |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** | **.** | **3** |

Shade **one** lozenge which gives the name of the // operator in line 10 in **Figure 2**.

[1 mark]

|  |  |  |
| --- | --- | --- |
| **A** | Assignment |  |
| **B** | Division |  |
| **C** | Integer division |  |
| **D** | Modulus | Icon  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** | **.** | **4** |

Complete the trace table for the algorithm shown in **Figure 4** some values have already been entered. You do not have to use all rows.

[5 marks]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **mid** | **low** | **high** | **found** | **OUTPUT** |
| 0 | 0 | 8 | False |  |
| 4 | 5 |  |  |  |
| 6 | 7 |  |  |  |
| 7 |  |  | True |  |
|  |  |  |  | 7 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1 mark for each correct column. Allow variable values to be copied down if they are unchanged.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **5** | **.** | **5** |

State the name of the algorithm given in **Figure 4**.

[1 mark]

Binary search (1).

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **6** |  |  |

Write a Python program that calculates the area of a triangle. The formula to calculate the area is:

*area = (base × height) ÷ 2*

The program should:

* Allow the user to enter the base in metres (the user may enter floating point numbers)
* Allow the user to enter the height in metres (the user may enter floating point numbers)
* Validate that both base and height have been entered
* If either base or height have not been entered, output “You must enter numbers for both base and height"
  + The program does not need to ask the user to enter the base and height again if they don’t enter numbers for base or height
* If both base and height are entered, calculate and output the area of the triangle

You should use meaningful variable name(s), correct syntax and indentation in your answer.

The answer grid below contains vertical lines to help you indent your code accurately.

[9 marks]

**Program Design (2 marks)**

Meaningful variable names throughout (even if the logic is incorrect) (1).  
Suitable data types throughout (must be a float for base, height and area) (1).

**Program Logic (7 marks)**

Get user input for base and assign to variable (1).  
Get user input for height and assign to variable (1).  
Check that both base and height are not empty (1).  
Output error message (1).  
Calculate area according to formula (1).  
Convert inputs to floats (1).  
Output the area (1).  
  
Accept different versions of the calculate if correct –   
e.g. area = 0.5 \* base \* height

**Maximum 8 marks** if any errors in code.

Python example (fully correct)

base = input("Enter base: ")

height = input("Enter height: ")

if base == "" or height == "":

print("You must enter numbers for both base and height")

else:

area = (float(base) \* float(height)) / 2

print(area)

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **7** |  |  |

Write a Python program that creates usernames.

The program allows the user to input their last name. It then takes the first four letters of the name and combines it with a random number between 1 and 9. You can assume that all names entered will be four letters or longer.

For example, if the user enters “SINCLAIR”, the program outputs: SINC5 (the program randomly could output any username from SINC1 to SINC9)

Your program should work as follows:

* Gets the user to enter a string and store it in a suitable variable
* Calculate a random number between 1 and 9
* Create a username from the first four characters of the name entered and the random number
* Output the username

You should use meaningful variable name(s), correct syntax and indentation in your answer.

The answer grid below contains vertical lines to help you indent your code accurately.

[8 marks]

**Program Design (1 marks)**

Meaningful variables names (even if the logic is incorrect) (1)

**Program Logic (7 marks)**

Get user input for name correctly (1)  
Store the result in a variable (1)  
Import random library (1)  
Generate random integer between 1 and 9 (1)  
Find the substring of the first four letters of the last name (1)  
Concatenate the random integer to the end of the substring (1)  
Output the username (1)

**Maximum 7 marks** if any errors in code

Python example 1 (fully correct)

import random

lastName = input("Enter your last name: ")

rand = random.randint(1,9)

username = lastName[:4] + str(rand)

print(username)

Accept the substring being created by taking each element of the string – e.g.  
lastName[0] + lastName[1] + lastName[2] + lastName[3]

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **8** |  |  |

A child’s stacking toy consists of 10 wooden blocks that can be stacked onto a pole. The blocks are labelled from smallest (1) to largest (10).

A robot has been made to stack the blocks. The following subroutines control the robot.

|  |  |
| --- | --- |
| ADD(blockNumber) | Adds the block labelled blockNumber to the top of the stack of cubes. |
| REMOVE() | Removes the top cube from the pole and returns the number of the block removed |
| HEIGHT() | Returns the number of blocks currently on the pole. |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **8** | **.** | **1** |

The following shows how the blocks are arranged at the start.

Text, icon

Description automatically generated

Draw the final arrangement of the blocks after the following algorithm has run.

ADD(6)  
ADD(7)  
ADD(8)

[3 marks]

A picture containing icon

Description automatically generated

Three blocks added (1)  
Labelled with 6,7,8 (1)  
6 at the bottom and 8 at the top (1)  
Accept drawings where the size of blocks are all the same or not in perfect proportion.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **8** | **.** | **2** |

The pole is empty at the start.

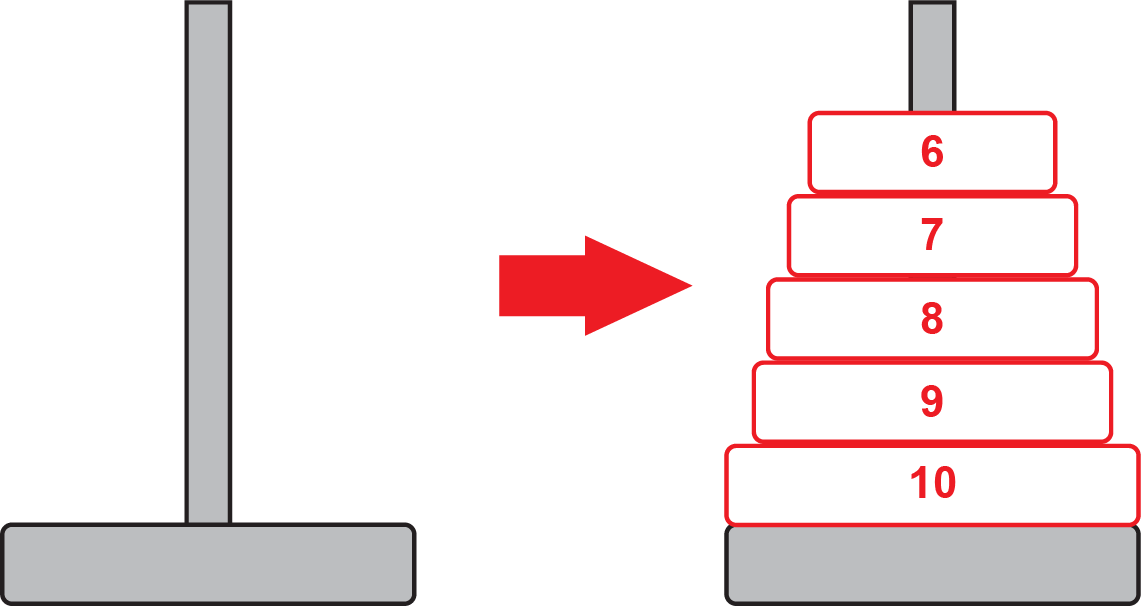
Draw the final arrangement of the blocks on the pole after the following algorithm has run.

FOR i ← 10 TO 6 STEP -1

ADD(i)

ENDFOR

[3 marks]



Blocks start at 10 (1)  
and go in consecutive order to 6 (1)  
With 10 at the bottom and 6 at the top (1)

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **8** | **.** | **3** |

The subroutine HEIGHT() returns the number of blocks that are currently stacked. The pole is able to hold only 7 blocks at a time.

Develop an algorithm using pseudocode for a new subroutine FULL which has no parameters and will return True if the pole is full and False if more blocks are able to be added.

[5 marks]

* Create subroutine with no arguments (indicated by brackets with nothing inside) (1)
* IF structure used (1)
* Correct condition in IF statement (1)
* Return False if there are fewer than 7 items stacked (1)
* Return True if there are 7 items stacked (or more) (1).

**Example 1**

SUBROUTINE FULL()  
 IF HEIGHT() < 7 THEN  
 RETURN False  
 ELSE  
 RETURN True  
ENDSUBROUTINE

**Example 2**

SUBROUTINE FULL()  
 currentHeight ← HEIGHT()  
 IF currentHeight < 7 THEN  
 RETURN False  
 ELSE  
 RETURN True  
ENDSUBROUTINE

**Example 3**

SUBROUTINE FULL()  
 IF HEIGHT() >= 7 THEN  
 RETURN True  
 ELSE  
 RETURN False  
ENDSUBROUTINE

Allow two IF statements to determine full and not full.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **8** | **.** | **4** |

Complete the algorithm below using pseudocode which will determine if the blocks are in order with the smallest at the top and the largest at the bottom.

To be in order, any block can be on the bottom, but all blocks above must be exactly 1 smaller than the block it sits on. The following three arrangements are given as examples:

Diagram

Description automatically generated with medium confidence

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outputs True This is because all blocks are in order |  | Outputs False This is because Block 7 is missing |  | Outputs False This is because Block 7 is larger than Block 6, but above it |

Your algorithm will assume that the pole is already stacked. The algorithm only needs to make use of the subroutines REMOVE and HEIGHT. When remove is called, the block is removed from the pole and the number on it is returned.

The algorithm currently removes the top block if there is one and stores the number on the block in the variable currentBlock. Complete the algorithm by:

* Using iteration to check all the other blocks in the stack are in order
* Output True or False depending on whether the blocks are in order

inOrder ← True  
currentBlock ← 0  
  
IF HEIGHT() > 0 THEN  
 currentBlock ← REMOVE()  
ENDIF

[7 marks]

* Indefinite loop (WHILE loop) used (1)
* With correct condition (1)
* REMOVE() used to find the next block (1)
* Comparison made of the next block with currentBlock to check it is one greater (1)
* inOrder changed to False if blocks don’t match (1)
* currentBlock changed to next block number (1)
* inOrder output (1)

**Example**

inOrder ← True  
currentBlock ← 0  
  
IF HEIGHT() > 0 THEN  
 currentBlock ← REMOVE()  
ENDIF  
  
WHILE HEIGHT() > 0 and inOrder  
 nextBlock ← REMOVE()  
 IF nextBlock != currentBlock + 1  
 inOrder ← False  
 ELSE  
 currentBlock ← nextBlock  
  
OUTPUT inOrder  
  
Checking inOrder is not necessary – this just saves checking further blocks if one is found out of place.  
Accept a FOR loop being used.

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **9** | **.** | **1** |

The following are computer science terms (labelled A-E).

**A** output  
**B** syntax error  
**C** abstraction  
**D** concatenation  
**E** string

For each of the definitions in the table, write the label of the most suitable Computer Science term. Use a label only once.

[3 marks]

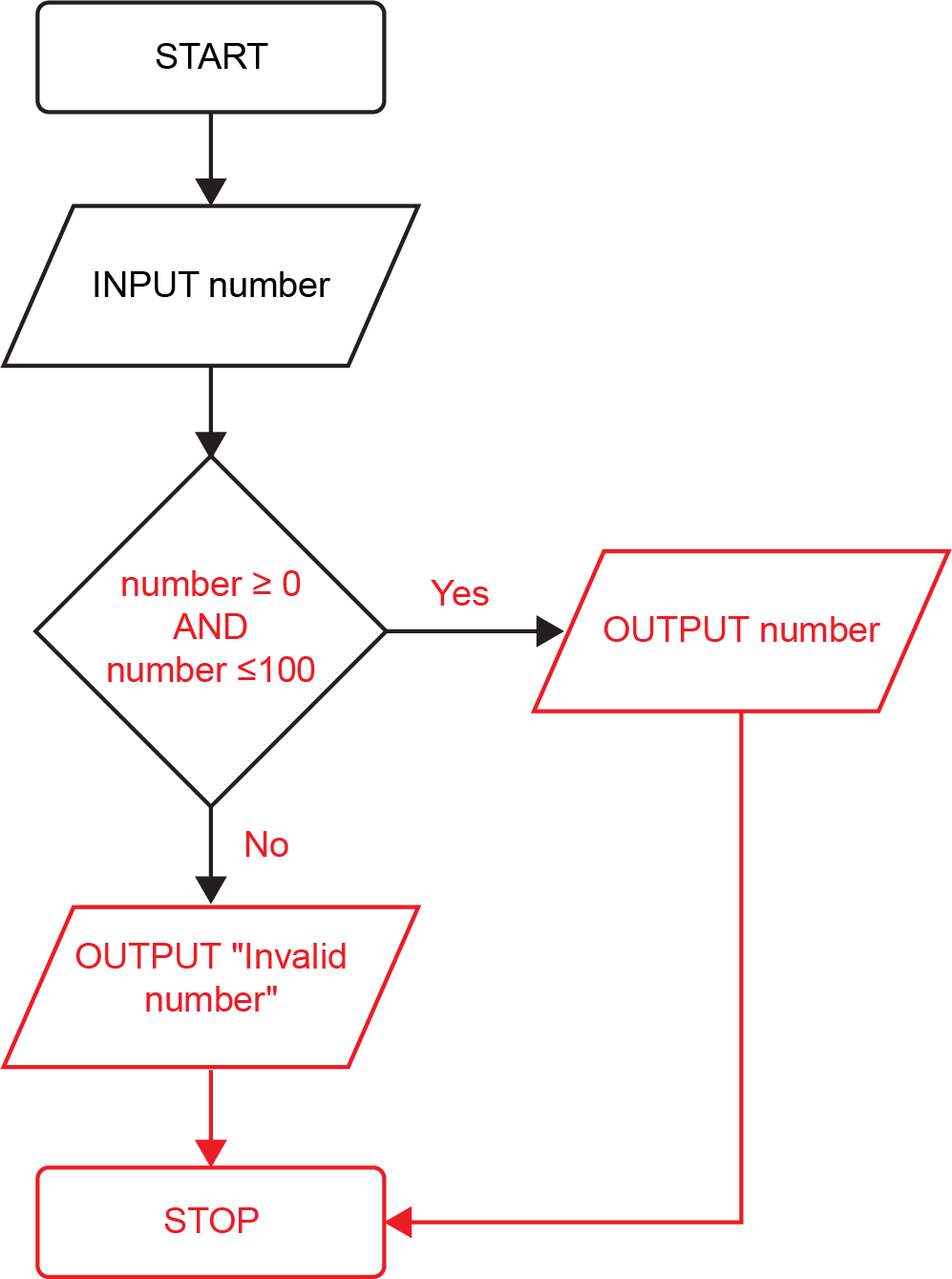
|  |  |
| --- | --- |
|  | **Label** |
| Combining two strings together | D |
| A series of characters | E |
| Ignoring or removing unnecessary details from a problem | C |

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **9** | **.** | **2** |

An algorithm will determine if a number is between 0 and 100 inclusive. If the number input is between 0 and 100, the algorithm will output the number. Otherwise, the algorithm will output “Invalid number”.

Complete the flowchart below for the algorithm described.

[5 marks]



* Condition looks for number greater than or equal to 0 (1)
* Condition looks for number also being less than or equal to 0. Boolean AND operator needed. Accept a second decision symbol with correct condition (1)
* Correct labels (Yes/No or True/False) from condition (1)
* Correct outputs of number and “Invalid number” (1)
* Correct symbol used for outputs (1)
* Stop symbol (1)

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **0** |  |  |

Fill in the blank arrays to show the steps involved in applying the bubble sort algorithm to the array ["Sam", "Dan", "Ivy", "Ed", "Eva"].

[5 marks]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| "Sam" | "Dan" | "Ivy" | "Ed" | "Eva" |
|  |  |  |  |  |
| "Dan" | "Sam" | "Ivy" | "Ed" | "Eva" |
|  |  |  |  |  |
| "Dan" | "Ivy" | "Sam" | "Ed" | "Eva" |
|  |  |  |  |  |
| "Dan" | "Ivy" | "Ed" | "Sam" | "Eva" |
|  |  |  |  |  |
| "Dan" | "Ivy" | "Ed" | "Eva" | "Sam" |
|  |  |  |  |  |
| "Dan" | "Ed" | "Ivy" | "Eva" | "Sam" |
|  |  |  |  |  |
| "Dan" | "Ed" | "Eva" | "Ivy" | "Sam" |
|  |  |  |  |  |

1 mark for each correct change (allow follow on)

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **1** |  |  |

The following Python program asks the user to enter the quantity of a product they would like to buy in an online shop’s promotion. The customer will only be allowed to order between 1 and 10 of the product. The program is shown in **Figure 5**.

**Figure 5**

MAX\_PRODUCTS = 10

quantity = 0

addProducts = int(input("Quantity: "))

if addProducts >= 1 and addProducts <= 10:

quantity = addProducts

print(quantity, "products added")

else:

print("You can only order up to 10 items")

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **1** | **.** | **1** |

The test data in the following test plan is for three different types of test.  
Complete the test plan for the code shown in **Figure 5**.

[2 marks]

|  |  |  |
| --- | --- | --- |
| **Test type** | **Test data** | **Expected result** |
| Normal data | 6 | Output “6 products added” |
| Erroneous data | -5 | Output “You can only order up to 10 items” |
| Boundary data | 11 | Output “You can only order up to 10 items” |

1 mark for each correct row

Accept extreme data instead of boundary data  
Accept invalid data instead of erroneous data

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **1** | **.** | **2** |

The program currently only asks the user to enter the quantity once.

Using Python only, change the program code in **Figure 5** so that if the user enters an invalid number (outside the range 0 to 10), the program will explain that an invalid number has been entered and ask them to enter the quantity. The program will repeatedly do this until a valid number has been entered.

[4 marks]

MAX\_PRODUCTS = 10

quantity = 0

validQuantity = False

while not validQuantity:

addProducts = int(input("Quantity: "))

if addProducts >= 1 and addProducts <= 10:

quantity = addProducts

validQuantity = True

else:

print("You can only order up to 10 items")

print(quantity, "products added")

Flag create such as validQuantity (1)  
Use of indefinite iteration (WHILE loop) (1)   
Correct condition in the WHILE loop (1)  
Flag changed if the quantity is valid (1)

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** |  |  |

A Python program has been written to calculate the number of 500 gram bags required to store an amount of flour entered by the user. The program also calculates the amount of left-over flour once all the bags have been filled. The program is shown in **Figure 6**.

**Figure 6**

BAG\_SIZE = 500

flourWeight = int(input("How much flour delivered: "))

bagsNeeded = flourWeight // 500

flourLeft = flourWeight % 500

print(bagsNeeded, "bags required")

print(flourLeft, "g of flour left over")

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** | **.** | **1** |

Explain why the data type of flourLeft will be an integer.

[2 marks]

flourWeight is an integer (1)  
Therefore the remainder calculated / result of performing modulus will result in an integer (1).

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** | **.** | **2** |

State the output from the program when the user enters 2100 into the program.

[2 mark]

4 bags required  
100 g of flour left over

Accept no space between 100 and g.

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **2** |  | **3** |

State the name of the % operator used in the line of code:

flourLeft = flourWeight % 500

[1 mark]

Modulus (operator)

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **3** |  |  |

A palindrome is a word or number that reads the same backwards as it does forwards.

For example:   
racecar  
is a palindrome as it reads the same if you reverse it.

hello  
is not a palindrome as when reversed, it reads olleh.

The program should:

* Ask the user to enter a word or number
* Calculate whether the word or number is a palindrome
* If the word or number is a palindrome, output “word is a palindrome”
* If the word or number is not a palindrome, output “word is not a palindrome”

You should use meaningful variable name(s), correct syntax and indentation in your answer.

The answer grid below contains vertical lines to help you indent your code accurately.

[7 marks]

**Program Design (1 mark)**

Meaningful variable names throughout **and** suitable data types throughout (1).

**Solution 1 Program Logic (6 marks)**

Input requested and stored in variable (1).  
FOR loop that goes through each letter in the string (or half the string) (1).  
comparison of the current letter with the corresponding letter in the other half of the string (1).  
Correctly set a flag or calculate if the word is a palindrome (1).  
Output “word is a palindrome” if it is, or if using a flag (1).  
Output “word is not a palindrome” if it isn’t, or if using a flag (1).

**Alternative Solution 2 Program Logic (6 marks)**

Input requested and stored in variable (1).  
Reverse the word (1).  
Comparison of reversed word with the original word (1).  
IF and ELSE statements correctly used (1).  
Output “word is a palindrome” if it is (1).  
Output “word is not a palindrome” if it isn’t (1).

**Maximum 6 marks** if any errors in code

Accept alternative solutions that work correctly.

Python example 1 (fully correct)

isPalindrome = True

word = input("Enter word or number: ")

for i in range(0, len(word)//2):

if word[i] != word[len(word)-i-1]:

isPalindrome = False

if isPalindrome:

print("word is a palindrome")

else:

print("word is not a palindrome")

Python example 2 (fully correct)

word = input("Enter word or number: ")

if word == word[::-1]:

print("word is a palindrome")

else:

print("word is not a palindrome")

Acknowledgements

Artwork



*Change of Heart*

© Karen Stamper (30 cm × 30 cm)

Paper collage and acrylic on board

www.karenstampercollage.com