

Example 1.1 – Pseudocode

iGCSE Computer Science, SAM Paper 1, Q7(a)

- 7 Algorithms can be designed using pseudocode or flowcharts. Then, they need to be translated into code that a computing device can execute.

Figure 2 shows the pseudocode for an algorithm.

```

1 # This is the pseudocode for an algorithm
2 SET inNum TO 0
3 SET result TO 1
4 SET i TO 0
5
6 SEND "Enter a number: " TO DISPLAY
7 RECEIVE inNum FROM (INTEGER) KEYBOARD
8
9 IF (inNum < 0) THEN
10     SEND "Invalid input" TO DISPLAY
11 ELSE
12     IF (inNum = 0) THEN
13         SEND "Answer is 1" TO DISPLAY
14     ELSE
15         FOR i FROM 1 TO inNum DO
16             SET result TO result * i
17         END FOR
18         SEND "The answer is " & result TO DISPLAY
19     ENDIF
20 ENDIF

```

Figure 2

- (a) Use the information in Figure 2 to answer these questions.

- (i) Complete the table to show the output for the given input.

(3)

Input	Output message
0	
-12	
5	

- (ii) State the purpose of this algorithm.

(1)

.....

.....

.....

Example 1.2 – Flowcharts

GCSE Computer Science (2016), SAM Paper 2, Q10(a)

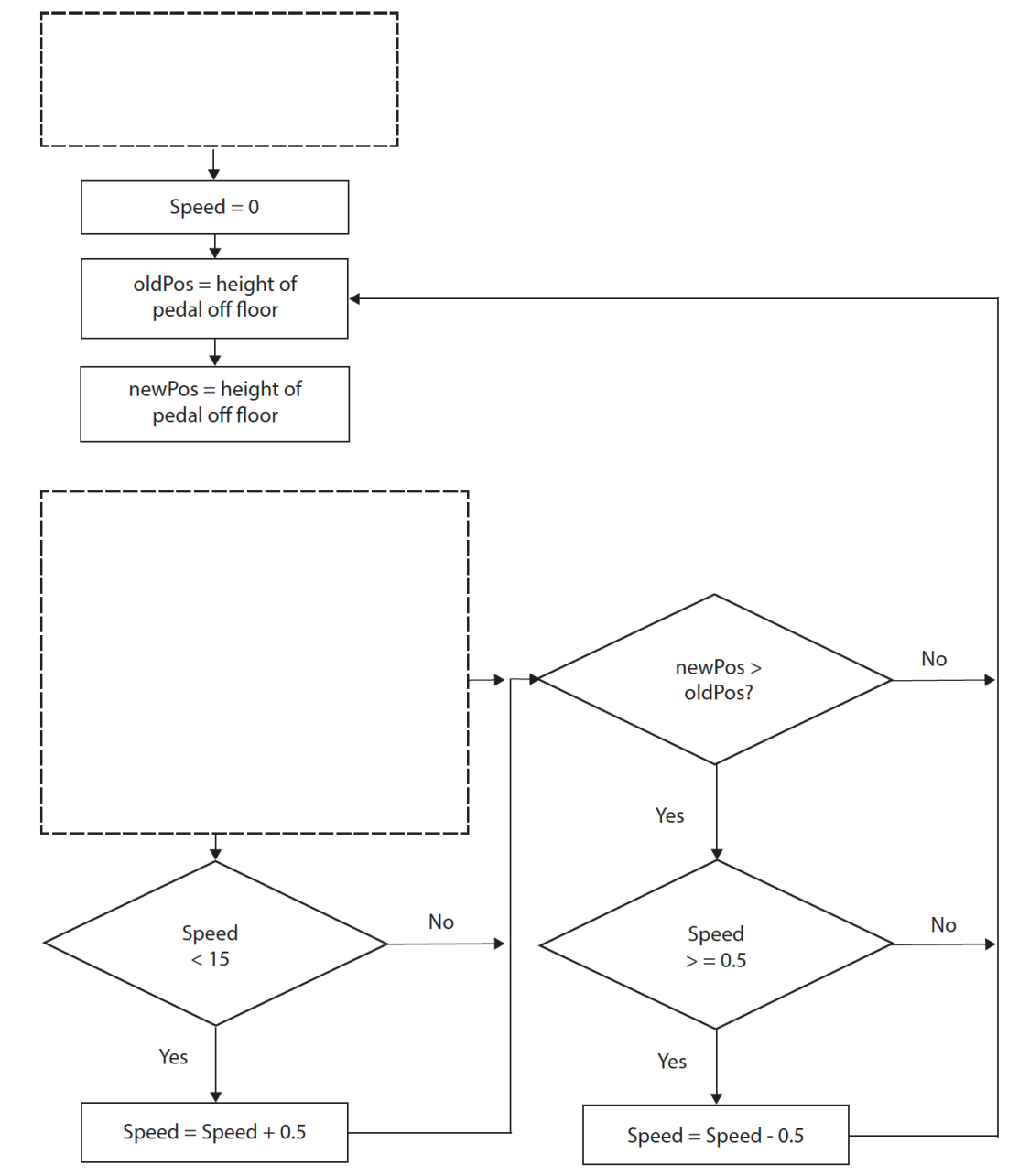
10 Each car at Sparky Autos has a pedal to make the car go forward.

(a) The drivers can make the car go forward by using the pedal.

- The closer the pedal is to the floor, the faster the car goes.
- The further the pedal is from the floor, the slower the car goes.
- Each car has its speed limited to 15 kilometres per hour.

Complete the flow chart to show this process.

(6)



Example 1.3 – Flowcharts

iGCSE Computer Science, SAM Paper 1, Q7(b)

(b) A bus company sets fares for different groups of passengers.

The fares are:

- a child fare for passengers 15 years old and younger
- a senior fare for passengers 65 years old and older
- a full fare for all other passengers.

Construct a flowchart of an algorithm that will determine the fare for one passenger when an age is input.

No validation of input is required.

(5)

Example 1.4 - Written descriptions

GCSE Computer Science (2016), Specimen Paper 2, Q6(b)

- 6 The HappyPetBox Company uses a software application to calculate staff wages and produce payslips. Sample input data for this system is shown.

National insurance (NI) number	Standard format LL123456L
Full time	Y or N Full-time = minimum of 40 hours per week Part-time = maximum of 20 hours per week
Hours worked	Integer Hours worked in current week Maximum of 10 hours overtime in one week for full-time only
Pay rate	Real Hourly pay rate

- (b) Create a **written description** of a function for the wages system. The function should receive hours worked and pay rate from the main program and return total pay due. Overtime rate is 1.5 x pay rate.

Do not use pseudo-code, program code, or a flowchart.

(4)

check if hours worked is > 40
 If no overtime → total = hours x pay rate
 If overtime worked → total = (hours - 40) x
 (pay rate x 1.5) + (40 x pay rate)

Example 1.5 - Program code
iGCSE Computer Science, SAM Paper 2, Q5(a)

5 Data, stored as numbers, is very easily processed using computer algorithms.

(a) Open the file **Q05a** in the code editor.

Complete the trace table to show the execution of the code.

You may not need to fill in all the rows in the table.

(5)

target	rs	rm	r

```
target = 4
r = 1
rs = 0
rm = 0

while (r <= target):
    rs = r ** 2
    print(rs)
    rm = r % 4
    print(rm)
    r = r + 1
```

Example 1.6 – A trace table for an algorithm expressed in pseudocode

GCSE Computer Science (2016), Specimen Paper 2, Q4(b)(ii)

- (b) In the summer, Sparky Autos is open more hours each day. This means additional members of staff are needed.

Pseudo-code that works out the number of days that additional members of staff are needed is shown.

```

2  # This program is owned by Sparky Autos
3
4  SET dates[] TO ["01/06/2014", "02/06/2014", "03/06/2014"]
5  SET extra[] TO ['Y', 'N', 'Y']
6  SET visitors[] TO [122, 51, 147]
7  SET staff[] TO [12, 6, 10]
8
9  SET length TO LENGTH (extra)
10 SET count TO 0
11 SET index TO 0
12
13 REPEAT
14     IF extra[index] = 'Y' THEN
15         SET count TO count + 1
16     END IF
17     SET index TO index + 1
18 UNTIL index = length
19

```

- (ii) Complete the trace table showing the execution of the pseudo-code. You may not need to fill in all the rows in the table.

(4)

Length	Count	Index	Extra (index)

Example 1.7 – Bubble sort

iGCSE Computer Science, SAM Paper 2, Q4(b)

- (b) Here is a list of numbers that need to be sorted in **ascending** order.

28	7	26	21	34	18	16	9
----	---	----	----	----	----	----	---

Perform the first pass of a bubble sort.

Use this space for working to help you answer the questions.

- (i) Complete the table to show how the list will have changed at the end of the first pass.

(1)

--	--	--	--	--	--	--	--

- (ii) State the number of comparisons made in the first pass.

(1)

-
- (iii) State the number of swaps made in the first pass.

(1)

Sample questions for Topic 1

(c) A bubble sort is only one type of sorting algorithm.

(i) Give **one** reason why a bubble sort is inefficient when sorting a large dataset.

(1)

.....

.....

(ii) State the position in a list that will always remain unchanged after the first pass of any ascending order bubble sort.

(1)

.....

Example 1.8 – Merge sort

GCSE Computer Science (2016), SAM Paper 1, Q6(e)

(e) A list is made up of the numbers 84, 52, 4, 6, 68, 39, 53, 1.

Show the steps involved when sorting this list of numbers using a merge sort algorithm.

(2)

Question Number	Answer	Additional Guidance																																						
6(e)	<div><ul style="list-style-type: none">Award 1 mark for each row of the process.</div> <div><table><tr><td>84</td><td>52</td><td>4</td><td>6</td><td>68</td><td>39</td><td>53</td><td>1</td></tr></table> <table><tr><td>52</td><td>84</td><td></td><td>4</td><td>6</td><td></td><td>39</td><td>68</td><td></td><td>1</td><td>53</td><td>(1)</td></tr></table> <table><tr><td>4</td><td>6</td><td>52</td><td>84</td><td></td><td>1</td><td>39</td><td>53</td><td>68</td><td>(1)</td></tr></table><p>This leads to:</p><table><tr><td>1</td><td>4</td><td>6</td><td>39</td><td>52</td><td>53</td><td>68</td><td>84</td></tr></table></div>	84	52	4	6	68	39	53	1	52	84		4	6		39	68		1	53	(1)	4	6	52	84		1	39	53	68	(1)	1	4	6	39	52	53	68	84	<div><ul style="list-style-type: none">Award 1 mark for each step.Marks should not be awarded for the last row.Any notation showing distinct lists at each stage is acceptable.</div>
84	52	4	6	68	39	53	1																																	
52	84		4	6		39	68		1	53	(1)																													
4	6	52	84		1	39	53	68	(1)																															
1	4	6	39	52	53	68	84																																	

Example 1.9 – Binary search

GCSE Computer Science (2013), Paper 1, June 2016, Q4(c)(ii)

Manuel is writing a binary search routine to search for an individual pupil in a list of all pupil numbers.

Here is the list of pupil numbers.

837, 1529, 1683, 2245, 3901, 3921, 4524

- (ii) Complete the table showing the pupil numbers visited and the associated sublists when using a binary search to locate the pupil number 1683.

(5)

Pupil number visited	Sublist

Pupil number visited	Sublist
2245	2245 > 1683

Examples 1.10 – Efficiency of algorithms

GCSE Computer Science (2016), Specimen Paper 2, Q5(a)

- 5 The accounts department at the HappyPetBox Company processes sales, subscriptions, and invoices.

The accounts office has a list of paid invoices, but they are all jumbled up. A bubble sort can be used to sort the list of paid invoice numbers.

The pseudo-code for the sort algorithm is shown.

```
1
2 # Bubble sort paid invoices
3
4 PROCEDURE sort()
5 BEGIN PROCEDURE
6     SET size TO LENGTH (invoices)
7     FOR i FROM 0 TO size - 1 DO
8
9         FOR j FROM 0 TO size - (i + 1) DO
10             IF (invoices[j] > invoices[j + 1]) THEN
11
12                 SET temp TO invoices[j]
13                 SET invoices[j] TO invoices[j + 1]
14                 SET invoices[j + 1] TO temp
15
16             END IF
17         END FOR
18
19     END FOR
20 END PROCEDURE
21
22
```

- (a) Explain an improvement that could be made to the algorithm to increase the efficiency of the sort.

(3)

.....

.....

.....

Example 1.11 – Decomposition

iGCSE Computer Science, SAM Paper 2, Q6

- 6 Open the file named **Q06** in the code editor.

In file **Q06**, the names and years of birth of artists are stored in a 2-dimensional data structure.

Labels for their work need to be created by joining the first letter of their last name, the first letter of their first name and their year of birth.

For example, a label for ('Andy', 'Warhol', 1928) would be 'WA1928'.

Write a program to:

- process each artist to create a label
- store all the labels in the data structure named 'theLabels'
- display the labels for all the artists
- find and display the name and year of birth of the youngest artist.

Your program should function correctly, even if 'theArtists' data structure has more, fewer, or different artists.

You **must** use the data structures in file **Q06**.

Save your amended code as **Q06FINISHED** with the correct file extension for the programming language.

(Total for Question 6 = 20 marks)

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

theLabels = []    # Put the new user labels into this structure

# Add your code here
```