JUNE 2023

Question	Answer
1(a)	**One** mark for each correct line. **Program development life cycle description** → **Program development life cycle stage** develop an algorithm to solve the problem using structure diagrams, flowcharts or pseudocode → design detect and fix the errors in the program → testing identify the problem and its requirements → analysis write and implement the instructions to solve the problem → coding
1(b)	**One** mark for naming or describing each component part, max **three** **For example:** inputs // what is put into the system processes // actions taken to achieve a result outputs // what is taken out of the system storage // what needs to be kept for future use

JUNE 2023 (2)

Question	Answer
1	**A**

Question	Answer
3	**One** mark for each correct answer • structure diagram / chart • flowchart • pseudocode

NOVEMBER 2023

Question	Answer
4	One mark for each correct word • array • constant • variable

Question	Answer
5(a)	One mark for each point (max two) • simplifying the problem • removing unnecessary details from the problem // selecting elements required • filtering out irrelevant characteristics from those elements
5(b)	One mark for each point (max three) • inputs • processes • outputs • storage
5(c)	One mark for stage, one mark for matching description (max two) • design (1) details of solution set out (1) • coding (1) program is developed (1) • testing (1) program is tested for errors (1)

JUNE 2024 (22)

3	**One** mark for each correct answer **max three** MP1 abstraction MP2 decomposition MP3 identification of problem MP4 identification of requirements // outline of success criteria

NOVEMBER 2024 (21)

4	**One** mark for each point • analysis • design • testing

Question	Answer
5	**One** mark for each method identified, **one** mark for a further description (**max six)** • structure diagram (1) a hierarchical diagram showing the breakdown of a computer program into sub-programs (1) • flowchart (1) a diagram showing the ordered steps to complete a computer program (1) • pseudocode (1) shows what a program does in plain language (1)

NOVEMBER 2024 (22)

5(a)	**One** mark per mark point (**max one**) • Design • Coding • Testing
5(b)	**One** mark per mark point (**max three**) • Abstraction • Discard/remove irrelevant information / hiding complexities / keeping the key elements of the problem • Decomposition of the problem • Breaking the problem into inputs, processes and outputs • Identification of the problem • Identification of the requirements of the solution to the problem • Research into the problem by data collection • Example of data collection

NOVEMBER 2021 (1)

Question	Answer
3(a)(i)	**one** mark for sample, **one** mark for reason **max four** Normal Sample any positive value with three decimal places e.g. 5.682 Reason to test that normal data is **accepted** and processed correctly Erroneous Sample any value that would be rejected e.g. 5.6 or -1.345 or seven Reason to test that erroneous data is **rejected**
3(a)(ii)	Reason to test that 0.000 / -0.001 / highest possible non-positive is rejected and 0.001 / 0.000 / lowest positive number is accepted Sample 1 0.000 Sample 2 0.001
3(b)	**One mark** To check that values are entered as intended // to prevent incorrect values that meet the validation criteria being accepted **Two marks** Asking the user to enter the value twice and comparing the values // double entry (1) only accepting a value if both entries are identical (1) or Displaying the value as it is entered (1) so the user can put right errors have been made as the value was entered (1)

4 One mark for each correct column

Counter	Distinction	Mark	Award	OUTPUT
0	0			
1	1	88		
2		74		
3		60		
4	2	90		
5	3	84		
6	4	87		
7	5	95		
8		72		
9	6	84		
10		66		
		-1	0.6	Highly Commended

NOVEMBER 2021 (2)

Question	Answer
	Section B
2	**One** mark for **two** correct rows **Two** marks for **three** correct rows **Three** marks for **four** correct rows.

Statement	Validation (✓)	Verification (✓)	Neither (✓)
a check where data is re-entered to make sure no errors have been introduced during data entry		✓	
an automatic check to make sure the data entered has the correct number of characters	✓		
a check to make sure the data entered is sensible	✓		
a check to make sure the data entered is correct			✓

3	**One** mark per bullet point **Normal test data** • Test data e.g. 50 (allow any number between 1 and 100 inclusive) • Reason Data that is within range and should be **accepted** **Extreme test data** • Test data 100 / 1 • Reason Data at the **maximum** / **minimum** end of the range and should be **accepted** **Erroneous test data** • Test data e.g. 300 (allow anything that isn't between 1 and 100 inclusive, including other data types) • Reason Data outside the range that should be **rejected**

JUNE 2022 (1)

Question	Answer
	Section B
2	**Four** marks for five correct rows **Three** marks for four correct rows **Two** marks for three correct rows **One** marks for two correct rows

Description	Data type				
	Boolean	Char	Integer	Real	String
a single character from the keyboard		✓			
multiple characters from the keyboard					✓
only **one** of **two** possible values	✓				
only whole numbers			✓		
any number				✓	

3	**One** mark per mark point, max **four** • Normal test data computerscience@cambridge.org.uk • Reason this is a valid email address (containing the @ symbol) and should be accepted • Erroneous test data computerscienceisgreat • Reason this is just a string, and should be rejected (as an email address needs a single '@')

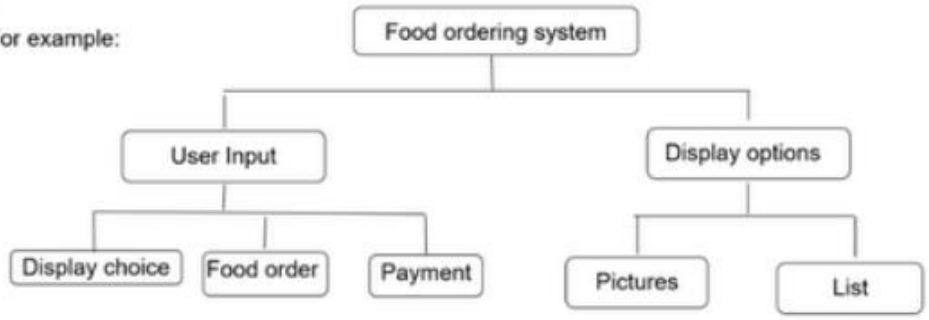
JUNE 2022 (3)

Question	Answer
	Section B
2	**One** mark per row, max **four**

Description	Types of test data			
	Boundary	**Erroneous / Abnormal**	**Extreme**	**Normal**
test data that is always on the limit of acceptability			✓	
test data that is either on the limit of acceptability or test data that is just outside the limit of acceptability	✓			
test data that will always be rejected		✓		
test data that is within the limits of acceptability			✓	✓

3	**One** mark per mark point, max **four** • variables are used to represent values that can change during the execution of a program // variables can be used to store the results of calculations / counting / totalling // can store values entered by the user • variable example – any data that is input into a program such as a date • constants represent values that **must stay the same throughout the execution of a program** • constant example – any value that does not change, such as Pi in mathematical formulae

MARCH 2023

5	**One** mark for a suitable hierarchical structure **One** mark for suitable names for the sub systems for user input and display options **One** mark for sub systems for user inputs, (choice of display,) food order and payment **One** mark for sub systems for display output types, pictures and list For example: 

NOVEMBER 2024 (23)

10(a) MYWORD

Password	Accept	Index	Found	OUTPUT
MYWORD	TRUE			(Please enter password)
	FALSE			
	FALSE			
		1	FALSE	
	FALSE			Rejected

M!word

Password	Accept	Index	Found	OUTPUT
M!word	TRUE			(Please enter password)
	FALSE			
		1	FALSE	
	FALSE			Rejected

10(a) My!Hidden

Password	Accept	Index	Found	OUTPUT
My!Hidden	TRUE			(Please enter password)
		1	FALSE	
		2	FALSE	
		3	TRUE	
		4		
				Accepted

For each trace table, **one mark** Password and output, **one mark** `Accept`, `Index` and `Found` columns

10(b) **One** mark for each correct point

- maximum length 20 characters **and** minimum length 8 characters
- cannot be all upper-case letters or all lower-case letters // must contain upper-case and lower-case letters
- must contain an !

9 **One** mark for each error identified and correction

- first decision box flow line labels `Yes` should be `No` and vice versa
- second decision box `OR` should be `AND`
- Flow line from A to B should be should `A ← B`
- Flowchart symbol for `A ← C` should be a process box / rectangle

JUNE 2022 (3)

4(a) **One** mark per mark point, max **seven**

MP1 correct `In` column
MP2 correct `Logic` column
MP3 correct `Test` column
MP4 correct `Number` column
MP5 correct `Store[Count]` column
MP6 correct `Count` and `Limit` columns
MP7 correct `Out` and OUTPUT columns

In	Logic	Test	Number	Store [Count]	Count	Limit	Out	OUTPUT
					0	5		
1	TRUE	2	9					
		3						
	FALSE							
2	TRUE	2	5					
		3		5	1			
3	TRUE	2	8					
	FALSE							
4	TRUE	2	10					
	FALSE							
5	TRUE	2	7					
		3		7	2		0	5
							1	7

4(b) **One** mark per mark point, max **two**

- to find / output prime numbers
- … store prime numbers in an array

4(c)

One mark per mark point, max **three**

MP1 insert a `WHILE` loop … // pre-condition loop
MP2 … after `Input Number`
MP3 … with a condition to enter the loop `Number < 3`
MP4 an error message included within the loop to ask for a re-entry of `Number`
MP5 …with another input prompt for `Number`
MP6 `ENDWHILE` closes the loop and the program carries on from `REPEAT` in the original algorithm

OR

One mark per mark point, max **three**

MP1 insert a `REPEAT` loop … // post-condition loop
MP2 … before `Input Number`
MP3 a conditional statement should be placed after `Input Number`
MP4 …to check if `Number < 3`
MP5 if the number entered is <3, an error message included within the loop to ask for a re-entry of `Number`
MP6 `UNTIL Number >= 3` closes the loop and the program carries on from `REPEAT` in the original algorithm

JUNE 2023

3(a)

One mark per mark point, max **two**

- Validation is an automated check carried out by a computer
- … to make sure the data entered is sensible/acceptable/reasonable

MARCH 2023

6(a)

One mark for each error identified and correction

- Line `05 OUTPUT UsefulEnergyOut` should be `INPUT UsefulEnergyOut`
- Line `06 IF TotalEnergyIn <> -1 AND UsefulEnergy <> -1` should be:
  `IF TotalEnergyIn <> -1 AND UsefulEnergyOut <> -1`
- Line `11 UNTIL TotalEnergyIn <> -1 OR UsefulEnergyOut <> -1` should be:
  `UNTIL TotalEnergyIn = -1 OR UsefulEnergyOut = -1`

6(b)

One mark for checking for >= 92
One mark for outputting "**A-rated**" only if the condition is met
For example

```
IF Efficiency >= 92
  THEN
    OUTPUT "A-rated"
ENDIF
```

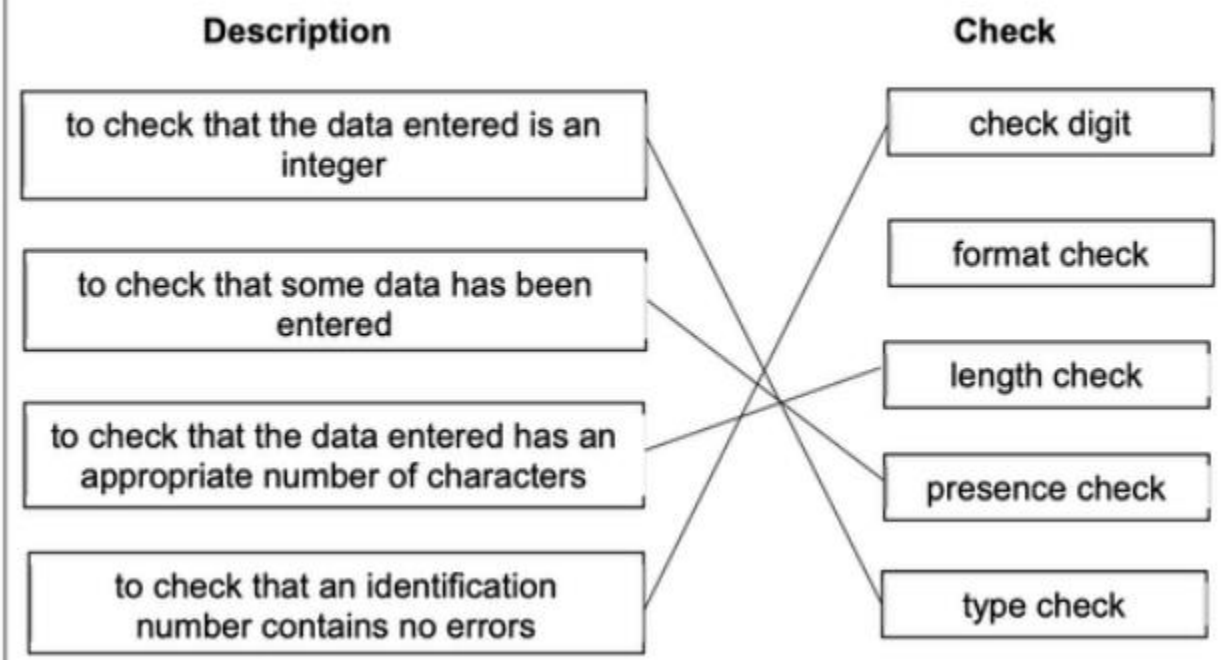
JUNE 2023 (2)

4(a)	**One** mark for each point (**max three**). • range check with acceptable values is (greater than) zero **and** less than 1000 • presence check to ensure the program will not continue until a value has been entered • type/character check to ensure that a number is entered • length check to ensure there are no more than 3 digits entered
4(b)(i)	To verify the data / for verification / as a verification check // to make sure that no changes are made to the data on entry

4(b)(ii)	**One** mark for each point (**max three**).
	• use of iteration
	• use of two inputs
	• to check that the two inputs are the same / different
	• use of the given variable `Measurement`
	
	For example
	`REPEAT`
	`  OUTPUT "Please enter measurement "`
	`  INPUT Measurement`
	`  OUTPUT "Please re-enter measurement "`
	`  INPUT MeasurementCheck`
	`UNTIL Measurement = MeasurementCheck`

JUNE 2023 (3)

5(a) **One** mark for each correct line.



5(b)

One mark per mark point, max **three**

- appropriate `REPEAT` / `WHILE` loop begin and end
- input of `Length`
- appropriate input prompt / error message
- correct loop exit/entry condition / selection

Example answers:

WHILE Loop

```
OUTPUT "Enter a number between 15 and 35 inclusive"
INPUT Length
WHILE Length <15 OR Length > 35 (DO)
    OUTPUT "Your number must be between 15 and 35 inclusive
    INPUT Length
ENDWHILE
```

REPEAT Loop

```
REPEAT
    OUTPUT "Enter a number between 15 and 35 inclusive"
    INPUT Length
UNTIL Length >= 15 AND LENGTH <= 35
```

JUNE 2023 (2)

7(a)

- `07`
- `04/12` or `16/18`
- `02/20`

7(b)

One mark for each error identified and correction

- Line `07 Total ← Total + Number * Counter`
  should be `Total ← Total + Number[Counter] * Counter`
- Line `08 IF Number[Counter] = 0`
  should be `IF Number[Counter] = -1` // should be `IF Number[Counter] < 0`
- Line `16 FOR Counter ← 0 TO 5`
  should be `FOR Counter ← 1 TO 5`

JUNE 2023

7(a) **One** mark per correct column, max **four**

Pointer	Letter	Choice	OUTPUT
1	F		
2			
3			
4			
5			
6			Letter F is represented by Foxtrot
			Another Letter? (Y or N)
		Y	
1	D		
2			
3			
4			Letter D is represented by Delta
			Another Letter? (Y or N)
		N	

7(b) (Linear) search

7(c) **One** mark per mark point, max **two**

- The algorithm would not stop
- … because it would not have found the item it was seeking

Or

- The array would run out of values after the pointer reached 13
- the algorithm will crash

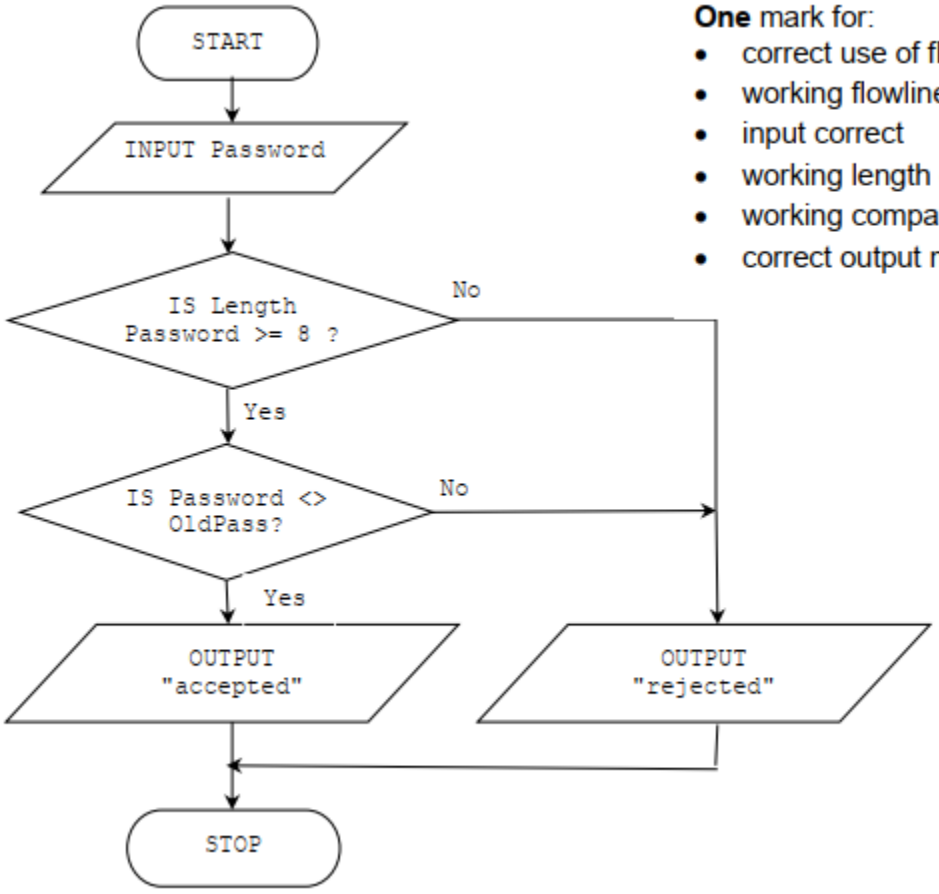
SPECIMEN 2023 – 2A

8 **One** mark for each.

10.00 boundary / abnormal data // the price should be rejected // value is out of range
9.99 boundary / extreme / normal data // the price should be accepted // value is within normal range
ten abnormal data // input should be rejected // value is wrong type

NOVEMBER 2024 (21)

8(a)



One mark for:

- correct use of flowchart symbols
- working flowlines
- input correct
- working length check
- working comparison
- correct output messages

MARCH 2024 (22)

5(a)

One mark for each error identified and correction

- Line `05 OUTPUT` should be `INPUT`
- Line `06 OR` should be `AND`
- Line `10 NEXT` should be `ENDIF`

5(b)(i)

One mark for checking for `< 0` or `>= 0`
One mark for checking of both inputs
One mark for correct repetition of both inputs e.g. use of `REPEAT WHILE` or using existing loop, in separate loops or both in a single loop

Example:

```
REPEAT
    OUTPUT "Enter cost price "
    INPUT Cost
UNTIL Cost >= 0
REPEAT
    OUTPUT "Enter selling price "
    INPUT Sell
UNTIL Sell >= 0
```

or

```
OUTPUT "Enter cost price "
INPUT Cost
WHILE Cost < 0
    OUTPUT "Enter cost price "
    INPUT Cost
ENDWHILE
OUTPUT "Enter selling price "
INPUT Sell
WHILE Sell < 0
    OUTPUT "Enter selling price "
    INPUT Sell
ENDWHILE
```

5(b)(ii)

One mark for identifying validation check and **one** mark for accompanying description **max four**

- presence check (1) to check that values have been input (1)
- type check (1) to check for numerical values (1)

3(a)

One mark for each point **max four**.

- input value, outside loop
- correct use of loop
- checking value input against contents of array …
- … appropriate action
- correct outputs

Example:

```
INPUT MyNumber
Location ← 0
FOR Index ← 1 TO 50
    IF Values[Index] = MyNumber
       THEN
         Location ← Index
    ENDIF
NEXT Index
IF Location = 0
   THEN
     OUTPUT "Not found"
   ELSE
     OUTPUT Location
ENDIF
```

3(b)

One mark for each point **max four**.

- use of inner and outer loop
- correct use of loops
- checking adjacent values in array …
- … swap if required
- correct stopping condition

```
Last ← 50
Repeat
    Swap ← FALSE
    FOR Index ← 1 TO Last - 1
        IF Values[Index] > Values[Index + 1]
          THEN
            Temp ← Values[Index]
            Values[Index] ← Values[Index + 1]
            Values[Index + 1] ← Temp
            Swap ← TRUE
        ENDIF
    NEXT
    Last ← Last - 1
UNTIL NOT Swap or Last = 1
```

4

One mark for each point **max three**.

- Integer
- real
- char
- string
- Boolean

JUNE 2023 (2)

9(a)

One mark for each column `F`, `C` and `T`
Two marks for columns `X[1]` to `X[5]` all entries correct or
One mark for columns `X[1]` to `X[5]` with one error

F	C	X[1]	X[2]	X[3]	X[4]	X[5]	T
		10	1	5	7	11	
0	1						10
1	2	1	10				10
1	3		5	10			10
1	4			7	10		
	5						
0	1						
	2						
	3						
	4						
	5						

9(b)

One mark for each point
- (bubble) sort data in array
- in ascending order

JUNE 2023 (3)

Question	Answer
4(a)	**One** mark per mark point, max **two** • To ensure that data has been accurately copied // to ensure that changes have not been made to the values originally intended when data is copied • … from one source to another

Question	Answer
4(b)	**One** mark for each appropriate verification check, max **two** **One** mark for each correct accompanying use, max **two** **For example:** Verification check 1 – Visual check Use – the user looks through the data that has been entered and confirms that no changes have been made. Verification check 2 – Double data entry Use – data is entered twice, the two entries are compared and if they do not match, a re-entry is requested.

NOVEMBER 2023 (2)

Question	Answer
2(a)	Format check
2(c)	**One** mark per mark point, max **two** MP1 check that there are 10 **characters** in total MP2 if the date is **too long/short** it will be **rejected**

NOVEMBER 2023

6(a)	Displaying/sort 10 names in alphabetical order 9

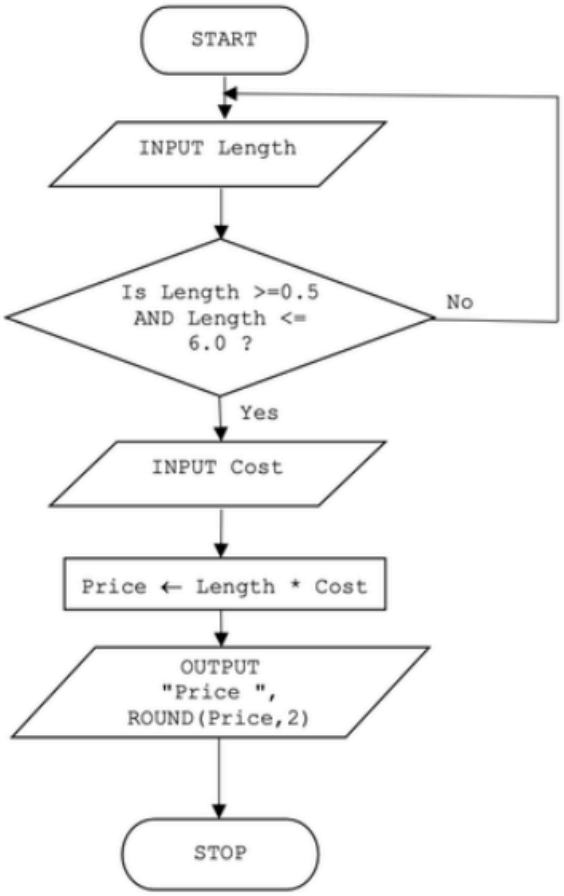
Question	Answer
6(b)	One mark for each point (max four) • Initialisation • inputting 10 names • storing the names in an array • sorting the names in alphabetical order using a bubble sort • displaying the 10 names • iteration

6(c)	**One** mark for a meaningful identifier for the array
	`A Names // ArrayNames`
	
	Two marks for 3 meaningful identifiers for variables
	One marks for 1 or 2 meaningful identifiers for variables
	`T Temp`
	`C Counter`
	`L Length`

6(d)	One mark for each point (max two) • use of comments • use of procedures/functions • use of white space

Question	
8(a)	Range check

8(b)



One mark for each of the following points
- correct use of flowchart symbols
- working flow lines and complete
- both inputs correct
- working range check
- working calculation
- correct output rounded to two decimal places

8(c)	One mark for set of test data, one mark for purpose (max four) Example: 1 and 1 (1) normal data to ensure the algorithm accepts this test data (1) –1 and 1 (1) abnormal data for length to ensure that it is rejected (1)

PUBLISHED

Question	Answer
8(d)	One mark for two correct headings Two marks for three correct headings Three marks for all headings correct and no other headings unless used in 8(b) **Length** | **Cost** | **Price** | **OUTPUT**
8(e)	**One** mark for each point (max two) • validate `Cost` ... • ... with a range/presence check • add another validation check for `Length`

4(a)

One mark per mark point, **max four**

- Line 01 / `DECLARE People : ARRAY[1:50, 1:3] OF REAL`
  should be `DECLARE People : ARRAY[1:50, 1:3] OF STRING`
- Line 10 / `Count ← 100`
  should be `Count ← 1`
- Line 12 / `CASE OF`
  should be `REPEAT`
- Line 27 / `UNTIL NOT Count`
  should be `UNTIL NOT Continue` // `UNTIL Continue = FALSE`

4(b)

- Use of appropriate loop
- Method to check array maximum not exceeded
- Method to check current / next array element not empty
- Output of all three array elements per array row (and no more)

Example algorithm:

```
Count ← 1
WHILE Count <= 50 AND People[Count, 1] <> "" DO
    OUTPUT People[Count, 1]
    OUTPUT People[Count, 2]
    OUTPUT People[Count, 3]
    Count ← Count + 1
ENDWHILE
```

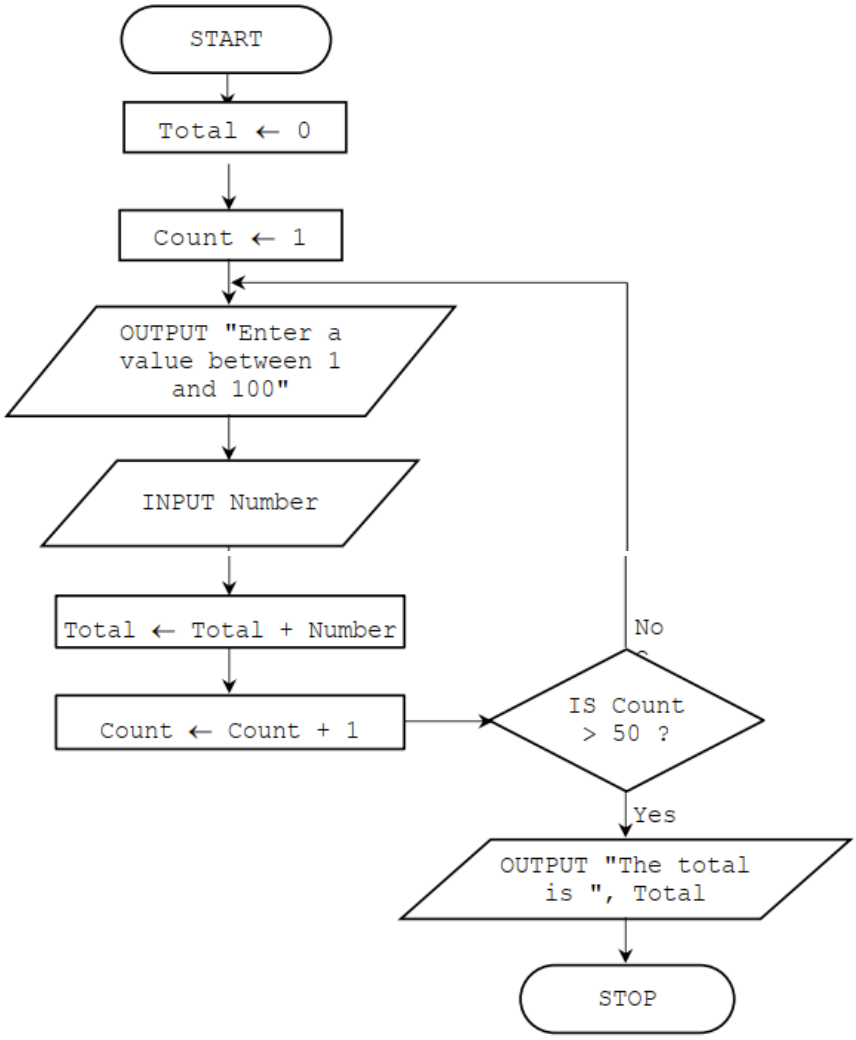
4(c)

One mark per mark point, **max four**

MP1 Declare/use a variable that is set to the maximum size of the array
MP2 … at the start of the program
MP3 After line 18
MP4 … check that the value of the counting variable is not greater than the array maximum variable
MP5 … and if it is do not allow any more entries / set the value of `Response` to `'N'` / add additional condition to `UNTIL` statement that checks if the counting variable is at maximum

(b) **One** mark per mark point

- Two labelled terminators START and STOP
- Loop initialisation, incrementation and termination
- Input with prompt
- `Total` initialised and totalling of inputs
- Output with message
- Fully correct flowchart with all correct arrows, yes/no labels and symbols



4(a)	**One** mark per mark point • Line 02 / `DECLARE Counter : STRING` should be `DECLARE Counter : INTEGER` • Line 09 / `Temp ← TRUE` should be `Swapped ← TRUE` • Line 10 / `WHILE Swapped = TRUE OR Pass <= Limit - 1 DO` should be `WHILE Swapped = TRUE AND Pass <= Limit - 1 DO` • Line 17 / `ItemList[Counter] ← Temp` should be `ItemList[Counter + 1] ← Temp` • Line 19 / `ENDCASE` should be `ENDIF`
4(b)	**One** mark per mark point (**max three**) • The use of a flag (set initially to FALSE) to show if a swap has been made (during the current iteration) • … to stop the loop if it has been sorted • The reduction in the limit of the (inner) loop after each iteration (of the loop) • … to reduce the number of comparisons / iterations required

M/J2025/22

4(a)	To check that the value entered has not changed on input.
4(b)	**One** mark per mark point, **max six** MP1 At least one appropriate working loop MP2 First inputs of numbers all stored in array `Numbers` MP3 Correct second input made for each number MP4 Working selection statement to compare first and second inputs one at a time MP5 If numbers don't match, appropriate output, including both values input and a message requiring re-input MP6 New input stored in current array location (replacing original input) MP7 Final output confirming completion, **outside loop**. **For example,**

```
// variable and array declarations or initialisations
// are not required for this question
FOR Index ← 1 TO 10
    INPUT Numbers[Index]
NEXT Index
FOR Index ← 1 TO 10
    INPUT CheckNumber
    IF CheckNumber <> Numbers[Index]
      THEN
        OUTPUT CheckNumber, " and ", Numbers[Index], " do
               not match.  Please re-enter the number. "
        INPUT Numbers[Index]
    ENDIF
NEXT Index
OUTPUT "The check has been completed."
```

5(a)

One mark per mark point

- Line 01 / `DECLARE Names : ARRAY[1:500] OF REAL`
  should be `DECLARE Names : ARRAY[1:500] OF STRING`
- Line 09 / Total ← `100`
  should be Total ← `0`
- Line 11 / `INPUT Names[Index]`
  should be `INPUT Names[Counter]`
- Line 13 / `Total ← Counter + Heights[Counter]`
  should be `Total ← Total + Heights[Counter]`
- Line 20 / `OUTPUT "The shortest height is ", Heights`
  should be `OUTPUT "The shortest height is ", Shortest`
  or `OUTPUT "The shortest height is ", Heights[Index]`

```
01 DECLARE Names : ARRAY[1:500] OF STRING
02 DECLARE Heights : ARRAY[1:500] OF REAL
03 DECLARE Shortest : REAL
04 DECLARE Total : REAL
05 DECLARE Counter : INTEGER
06 DECLARE Index : INTEGER
07 Shortest ← 500
08 Index ← 0
09 Total ← 0
10 FOR Counter ← 1 TO 500
11     INPUT Names[Counter]
12     INPUT Heights[Counter]
13     Total ← Total + Heights[Counter]
14     IF Heights[Counter] < Shortest
15       THEN
16         Shortest ← Heights[Counter]
17         Index ← Counter
18     ENDIF
19 NEXT Counter
20 OUTPUT "The shortest height is ", Shortest
21 OUTPUT "The shortest person is ", Names[Index]
22 OUTPUT "The average height is ", Total / 500
```

5(b)

One mark per mark point

MP1 Output statement with correct calculation of average, or use of a variable average.
MP2 Correct use of `ROUND` function set to 1 decimal place.

For example,

```
OUTPUT ROUND(Total / 500, 1)
```

5(c)	**One** mark per statement, **max four** MP1 Declaration of new variable for tallest person / largest value at start of algorithm MP2 Initialisation of largest variable to a low number e.g. 0 MP3 New selection statement after input of height to compare it with current largest value MP4 If input of height is larger than current largest variable, it should become the new largest value MP5 **Outside the loop**, output the current value of the largest variable.