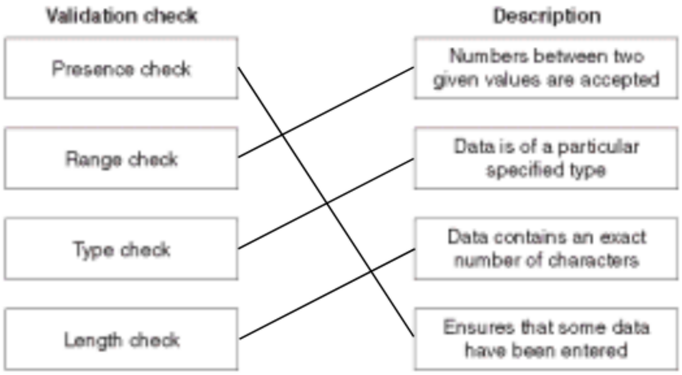
7. Algorithm design and problem-solving

Marking scheme

Q1)

1 mark for each correct link up to maximum of 3 marks


Validation check
Description
Presence check
Numbers between two given values are accepted
Range check
Data is of a particular specified type
Type check
Data contains an exact number of characters
Length check
Ensures that some data have been entered

[3]

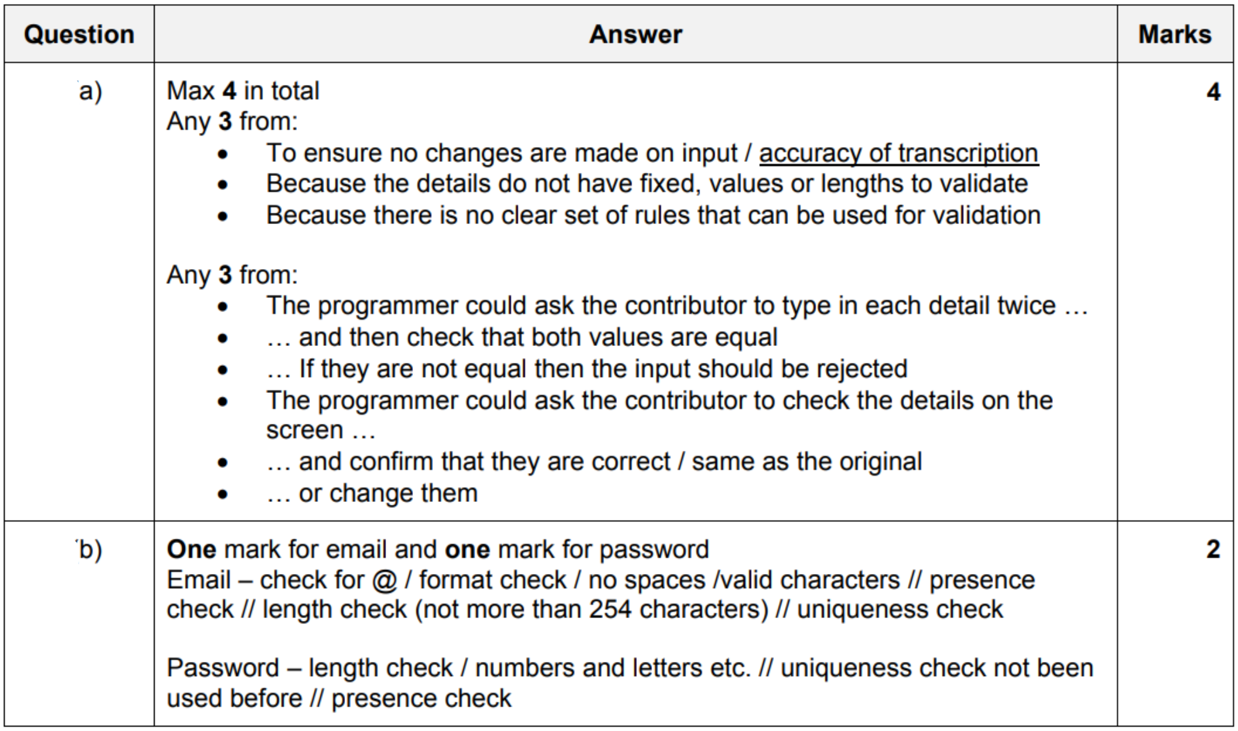
Q2)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td>1 mark for each correct answer
<table>
<tr><th>Statements</th><th>Validation</th><th>Verification</th></tr>
<tr><td>To automatically check the accuracy of a bar code</td><td>✓</td><td></td></tr>
<tr><td>To check if the data input is sensible</td><td>✓</td><td></td></tr>
<tr><td>To check if the data input matches the data that has been supplied</td><td></td><td>✓</td></tr>
<tr><td>To automatically check that all required data fields have been completed</td><td>✓</td><td></td></tr>
</table></td><td>4</td></tr>
</table>

Q3)

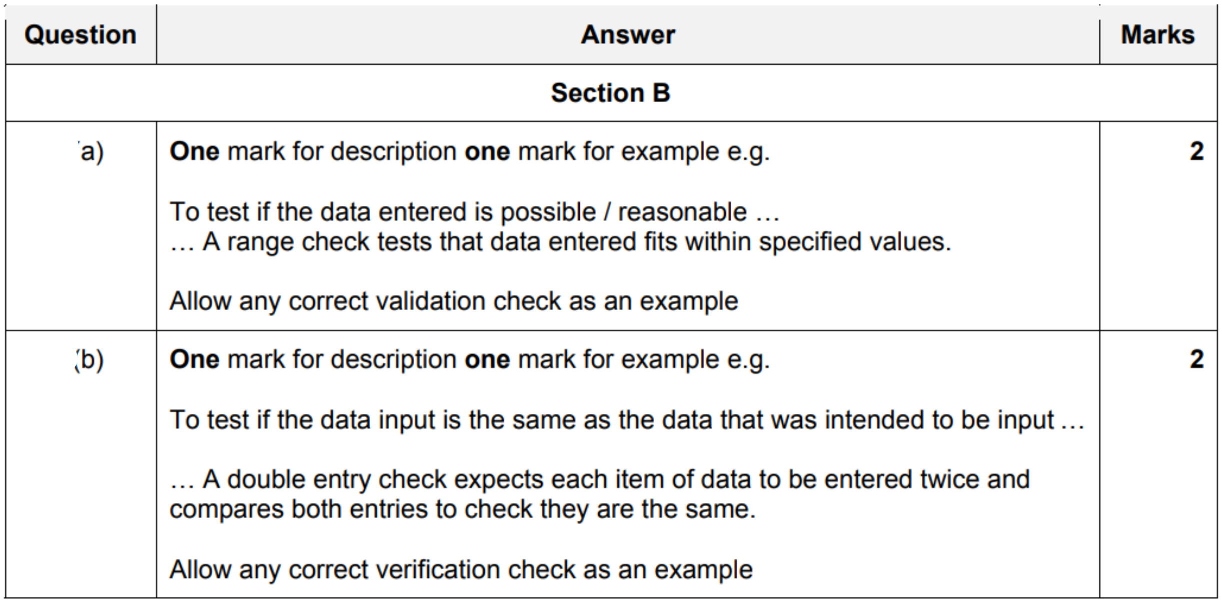
Question	Answer	Marks
(a)	1 mark per bullet: ∞ Validation checks whether data to be entered is possible/sensible // computer check ∞ Verification checks that data entered is the data that was intended to be entered // can be a human check // matches the source	**2**
(b)	1 mark for each valid point **Either** ∞ Double Entry // suitable practical example ∞ the data will be entered twice ∞ compared by the computer or by a human ∞ if a discrepancy is found, the data entry operator is asked to re-enter the data **Or** ∞ Visual Verification // suitable practical example ∞ the data will be compared to the source 'document' ∞ compared by a human ∞ if a discrepancy is found, the data is re-entered	**2**
(c)	1 mark for explanation and 1 mark for an expansion ∞ Library routine is a list of instructions // block of code // subroutine ∞ … that is used often … ∞ … which is given a name ∞ … and which can be called from other programs ∞ Library routines make writing programs easier and faster as the code is already written ∞ Library routines make program testing easier as the code has already been tested and debugged	**2**

Q4)



Question	Answer	Marks
a)	Max **4** in total Any **3** from: • To ensure no changes are made on input / accuracy of transcription • Because the details do not have fixed, values or lengths to validate • Because there is no clear set of rules that can be used for validation Any **3** from: • The programmer could ask the contributor to type in each detail twice … • … and then check that both values are equal • … If they are not equal then the input should be rejected • The programmer could ask the contributor to check the details on the screen … • … and confirm that they are correct / same as the original • … or change them	**4**
b)	**One** mark for email and **one** mark for password Email – check for @ / format check / no spaces /valid characters // presence check // length check (not more than 254 characters) // uniqueness check Password – length check / numbers and letters etc. // uniqueness check not been used before // presence check	**2**

Q5)



Question	Answer	Marks
	Section B	
a)	**One** mark for description **one** mark for example e.g. To test if the data entered is possible / reasonable … … A range check tests that data entered fits within specified values. Allow any correct validation check as an example	**2**
(b)	**One** mark for description **one** mark for example e.g. To test if the data input is the same as the data that was intended to be input … … A double entry check expects each item of data to be entered twice and compares both entries to check they are the same. Allow any correct verification check as an example	**2**

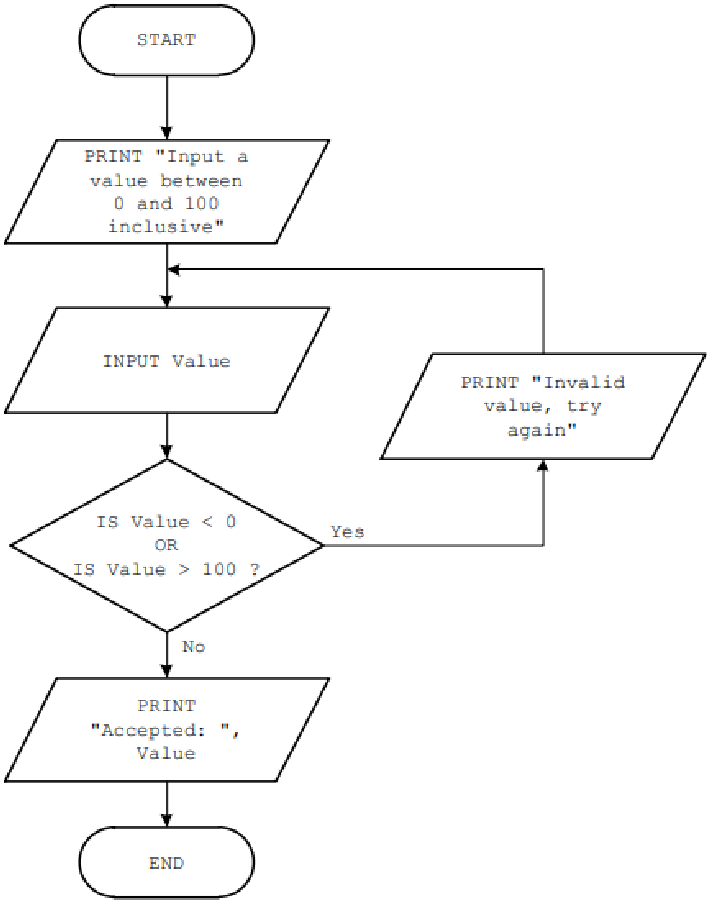
Q6)

Question	Answer	Marks
	1 mark for description 1 mark for example, e.g. To test if the value input falls between a given upper bound and a given lower bound … If a month has to be input using an integer, it must be between 1 and 12 inclusive. 1 mark for description 1 mark for example, e.g. To test if the data input is over/under a certain number of characters … An international telephone number can be no longer than 15 digits. 1 mark for description 1 mark for example, e.g. To test if the input is of the correct data type … If the input is expecting integer(s) to be entered, it will not permit a string to be entered.	**6**

Q7)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>Range check</td><td><b>1</b></td></tr>
<tr><td>(b)</td><td><b>Two</b> from:
• The entered number (<code>Value</code>) is being checked to see that it is not &lt; 0 or not &gt; 100
• If it is, it is rejected and the user has to enter another number / an error message is displayed
• Otherwise the number is accepted, the word ‘Accepted’ is output along with the <code>Value</code></td><td><b>2</b></td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(c)</td><td>
<table>
<tr><th><code>Value</code></th><th>OUTPUT</th></tr>
<tr><td></td><td><code>Input a value between 0 and 100 inclusive</code></td></tr>
<tr><td>200</td><td><code>Invalid value, try again</code></td></tr>
<tr><td>300</td><td><code>Invalid value, try again</code></td></tr>
<tr><td>–1</td><td><code>Invalid value, try again</code></td></tr>
<tr><td>50</td><td><code>Accepted: 50</code></td></tr>
<tr><td></td><td></td></tr>
</table>
1 mark – <code>Value</code> column
1 mark – OUTPUT column first line
1 mark – OUTPUT column lines two to five</td><td><b>3</b></td></tr>
</table>

Question	Answer	Marks
(d)	 START PRINT "Input a value between 0 and 100 inclusive" INPUT Value IS Value < 0 OR IS Value > 100 ? Yes PRINT "Invalid value, try again" No PRINT "Accepted: ", Value END 1 mark – Input prompt and input value 1 mark – Correct decision box labelled sufficiently (Yes/No) – allow 2 decision boxes 1 mark – Remaining outputs correct 1 mark – All connecting lines and arrows to be complete and correct 1 mark – Standard flowchart symbols used	5

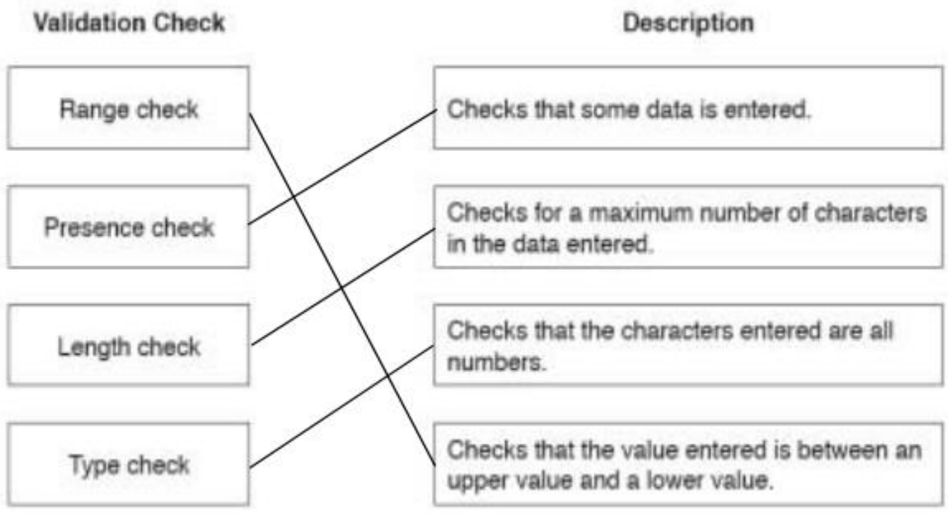
Q8)

Question	Answer	Marks
	Validation **Two** from: ∞ automated checking ∞ checking that data is reasonable / of a certain type ∞ checking that data meets certain criteria Example 1 mark ∞ range check // length check // type check // check digit etc. **Verification** **Two** from: ∞ checking that data has not changed… ∞ … during input to a computer ∞ … during transfer between computers / devices Example 1 mark ∞ double entry // checking against original // visual check // use of checksum etc.	**6**

Q9)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td>
<table>
<tr><th>Statements</th><th>Validation</th><th>Verification</th></tr>
<tr><td>Range check</td><td>✓</td><td></td></tr>
<tr><td>Double entry</td><td></td><td>✓</td></tr>
<tr><td>Check digit</td><td>✓</td><td></td></tr>
<tr><td>Presence check</td><td>✓</td><td></td></tr>
</table>
1 mark for each correct row</td><td><b>4</b></td></tr>
</table>

Q10)

Question	Answer	Marks
	 Validation Check Range check Presence check Length check Type check Description Checks that some data is entered. Checks for a maximum number of characters in the data entered. Checks that the characters entered are all numbers. Checks that the value entered is between an upper value and a lower value. One mark for each correct line, up to maximum of three marks.	3

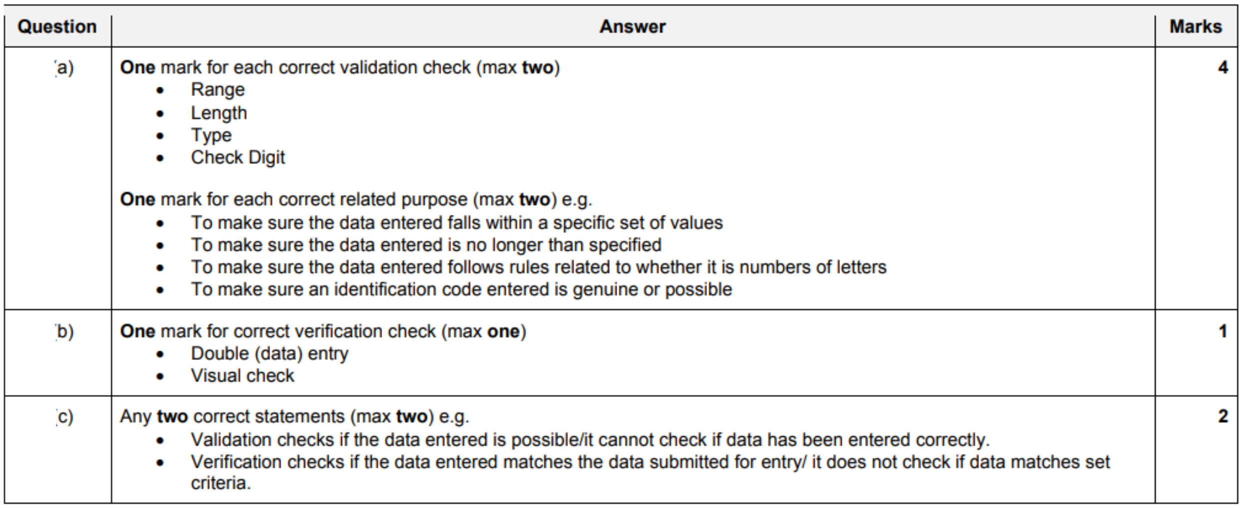
Q11)

Question	Answer	Marks
	Two from • Sub-program / system not the whole program / system • To perform a frequently used operation within a program • That can be called when needed • That can be reused by another program	2

Q12)

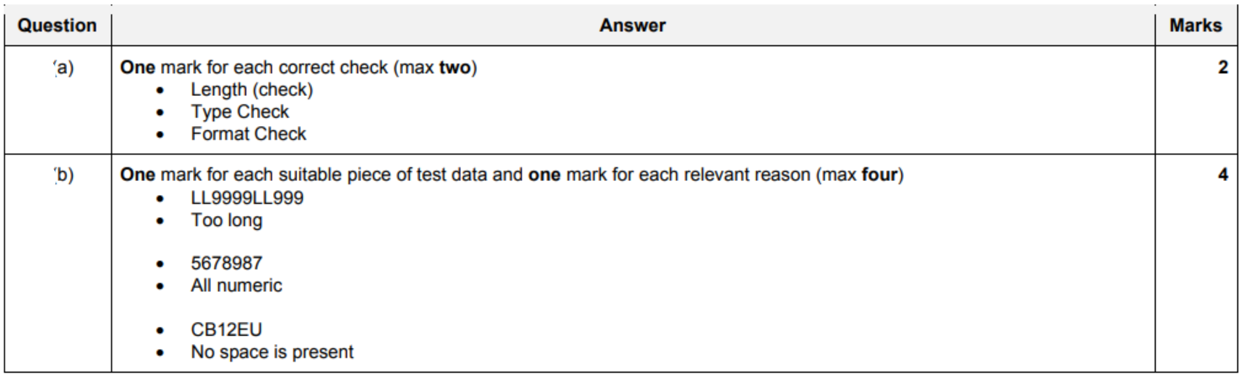
Question	Answer			Marks
	Statement	**True (✓)**	**False (✓)**	2
	A structure diagram is a piece of code that is available throughout the structure of a program.		✓	
	A structure diagram shows the hierarchy of a system.	✓		
	A structure diagram is another name for an array.		✓	
	A structure diagram shows the relationship between different components of a system.	✓		
	Two marks for four correct rows. **One** mark for three correct rows.			

Q13)



Question	Answer	Marks
a)	**One** mark for each correct validation check (max **two**) • Range • Length • Type • Check Digit **One** mark for each correct related purpose (max **two**) e.g. • To make sure the data entered falls within a specific set of values • To make sure the data entered is no longer than specified • To make sure the data entered follows rules related to whether it is numbers of letters • To make sure an identification code entered is genuine or possible	**4**
b)	**One** mark for correct verification check (max **one**) • Double (data) entry • Visual check	**1**
c)	Any **two** correct statements (max **two**) e.g. • Validation checks if the data entered is possible/it cannot check if data has been entered correctly. • Verification checks if the data entered matches the data submitted for entry/ it does not check if data matches set criteria.	**2**

Q14)



Question	Answer	Marks
a)	**One** mark for each correct check (max **two**) • Length (check) • Type Check • Format Check	**2**
b)	**One** mark for each suitable piece of test data and **one** mark for each relevant reason (max **four**) • LL9999LL999 • Too long • 5678987 • All numeric • CB12EU • No space is present	**4**

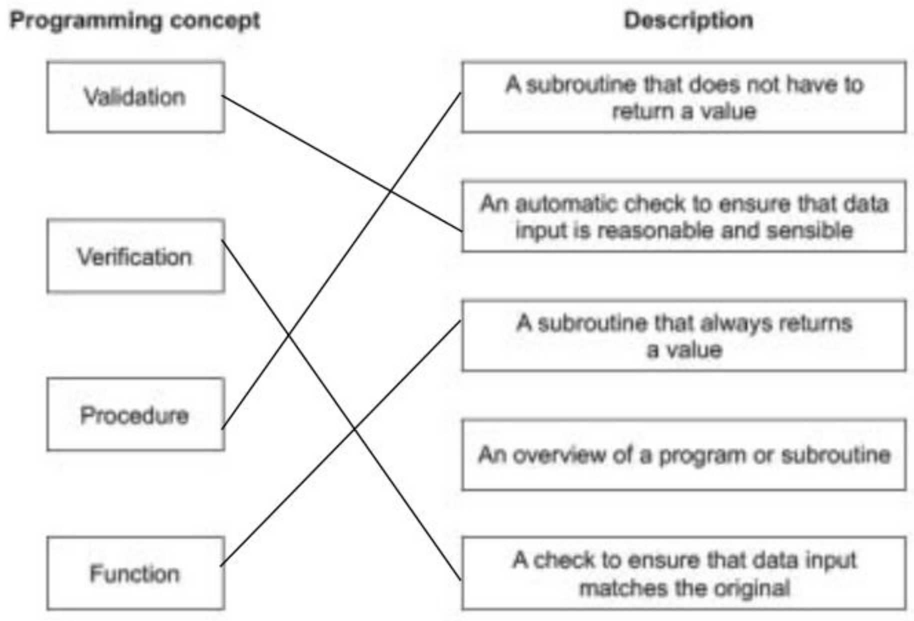
Q15)

Statement	true (✓)	false (✓)
A subroutine is called from within a program.	✓	
A subroutine is **not** a complete program.	✓	
A subroutine is a self-contained piece of code.	✓	
A subroutine must return a value to the code from which it was called.		✓

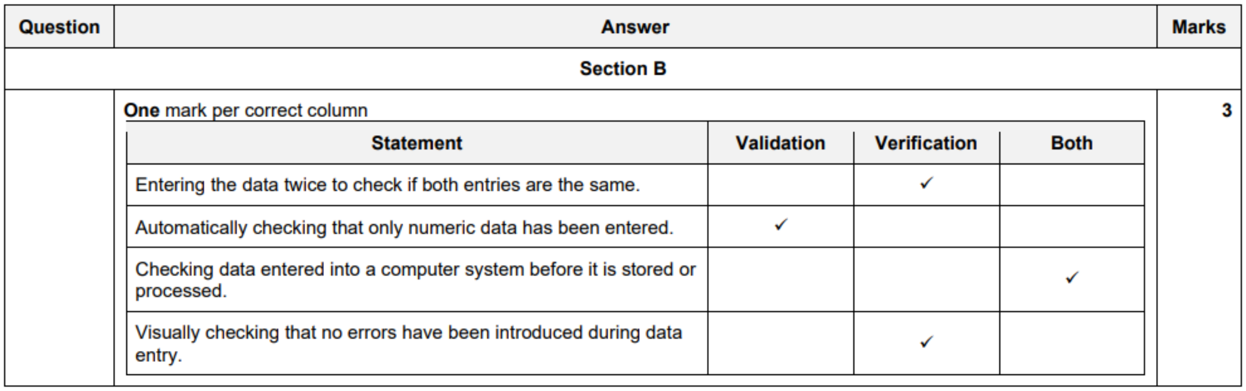
Two marks for four correct rows
One mark for any two correct rows

2

Q16)

Question	Answer	Marks
	One mark for each correct line  Programming concept Description Validation Verification Procedure Function A subroutine that does not have to return a value An automatic check to ensure that data input is reasonable and sensible A subroutine that always returns a value An overview of a program or subroutine A check to ensure that data input matches the original	4

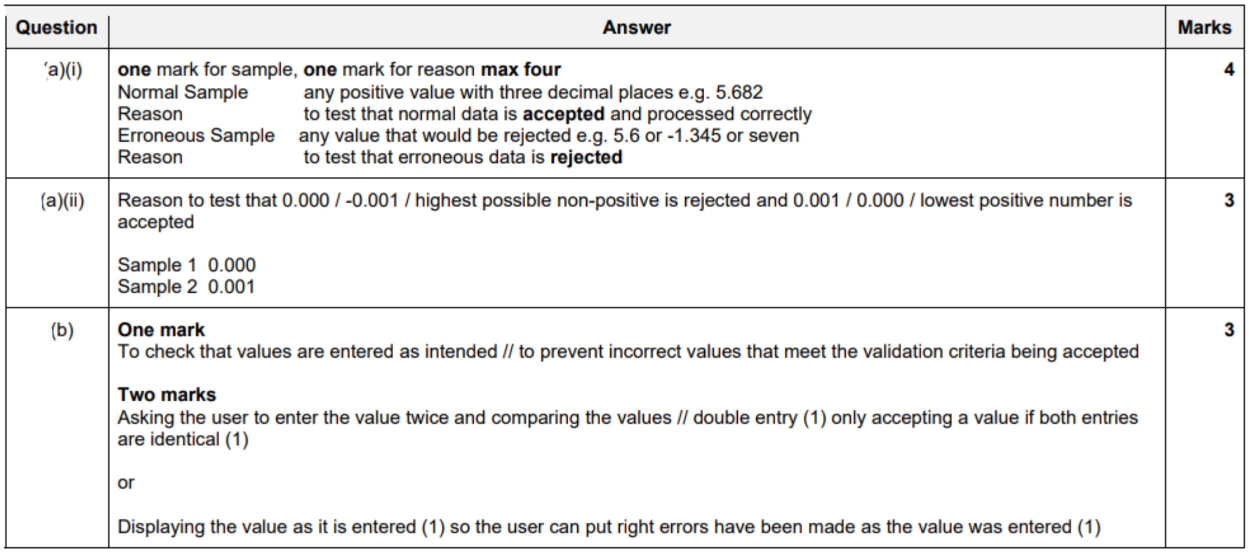
Q17)



Question	Answer	Marks
	Section B	
	One mark per correct column	**3**

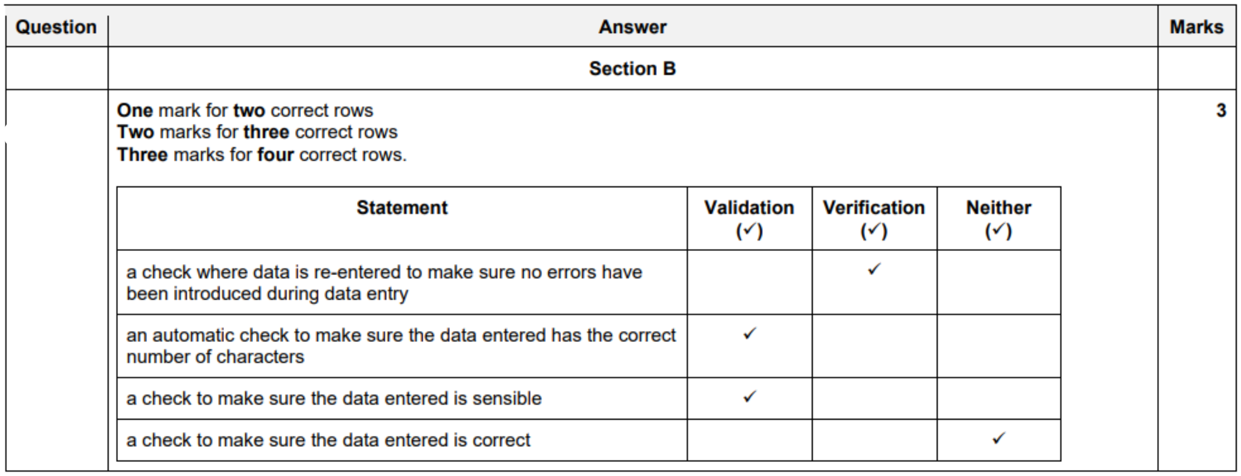
Statement	Validation	Verification	Both
Entering the data twice to check if both entries are the same.		✓	
Automatically checking that only numeric data has been entered.	✓		
Checking data entered into a computer system before it is stored or processed.			✓
Visually checking that no errors have been introduced during data entry.		✓	

Q18)



Question	Answer	Marks
(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**	**4**
(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)	**3**

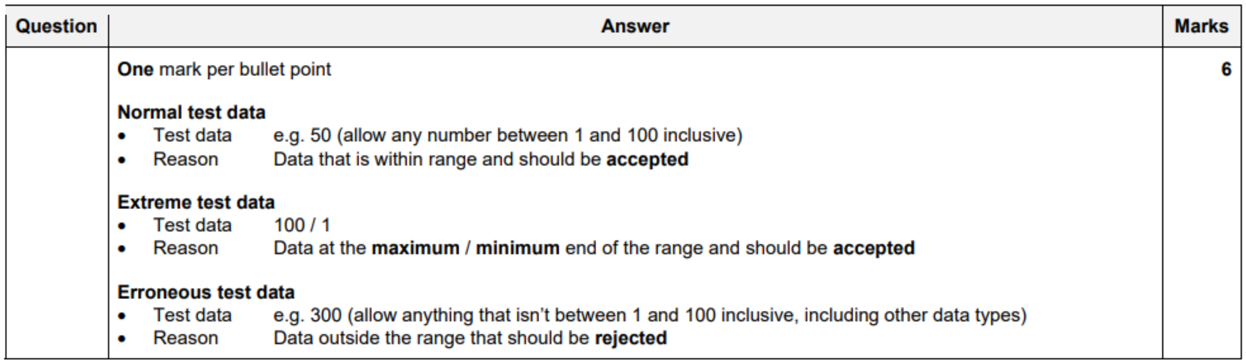
Q19)



Question	Answer	Marks
	Section B	
	One mark for **two** correct rows **Two** marks for **three** correct rows **Three** marks for **four** correct rows.	**3**

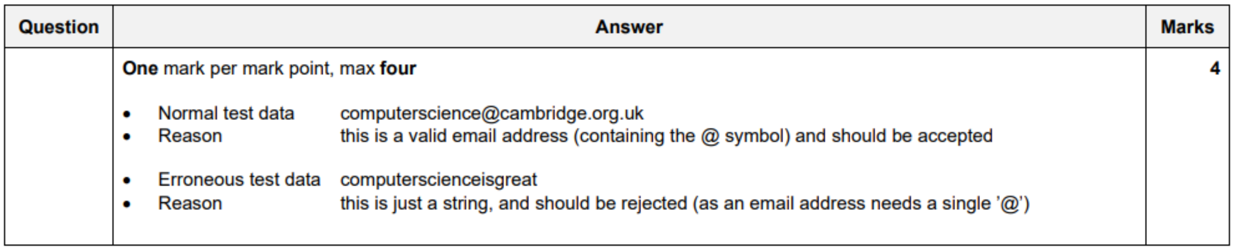
Statement	Validation (✓)	Verification (✓)	Neither (✓)
a check where data is re-entered to make sure no errors have been introduced during data entry		✓	
an automatic check to make sure the data entered has the correct number of characters	✓		
a check to make sure the data entered is sensible	✓		
a check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

Q20)



Question	Answer	Marks
	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**	**6**

Q21)



Question	Answer	Marks
	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 '@')	**4**

Q22)

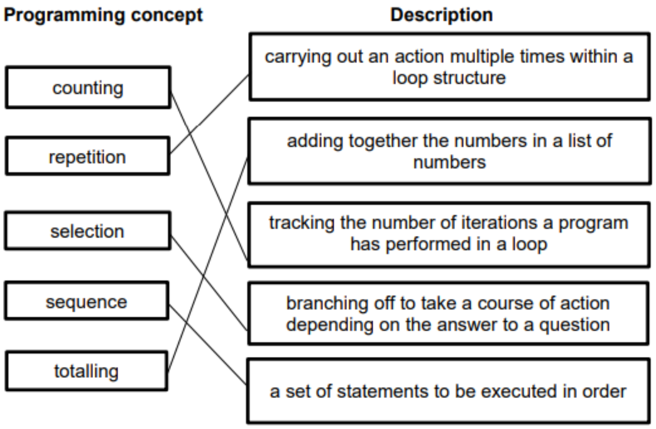
Question	Answer	Marks
Section B		
	One mark per row, max **four**	4

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			✓	✓

Q23)

Question	Answer	Marks
(a)	**Two** marks per check, description must match name of check if given, max **six** • Check 1 use a type check ○ to ensure that the value is a number / integer • Check 2 use a length check ○ to ensure that there are only 4 characters / digits • Check 3 use a range check ○ to ensure that the value is **>= 1000 and <=9999**	6
(b)	**One** mark per mark point, max **three** MP1 input the new PIN MP2 input the new PIN again // ask the user to check the number on screen MP3 check that both PINs are the same // confirm that it is the PIN to use MP4 check that the new PIN is not the same as the old PIN	3

Q24)

Question	Answer	Marks
	Section B	
	One mark for each correct line, max **four**  Programming concept Description counting repetition selection sequence totalling carrying out an action multiple times within a loop structure adding together the numbers in a list of numbers tracking the number of iterations a program has performed in a loop branching off to take a course of action depending on the answer to a question a set of statements to be executed in order	**4**

Q25)

Question	Answer	Marks
	One mark per mark point, max **three** MP1 verification is used to make sure the items in stock do not change from the original when they are input // verification is used to make sure the items in stock do not change from what was intended to be input // verification is used to make sure the items are **accurately copied** MP2 enter each item in stock twice / double entry // visual check MP3 matching description of the type of check stated in MP2 **Example answers** Double entry [1] enter data twice **and** only accept identical values [1] Visual check [1] look at the data that has been entered **and** confirm it matches [1]	**3**

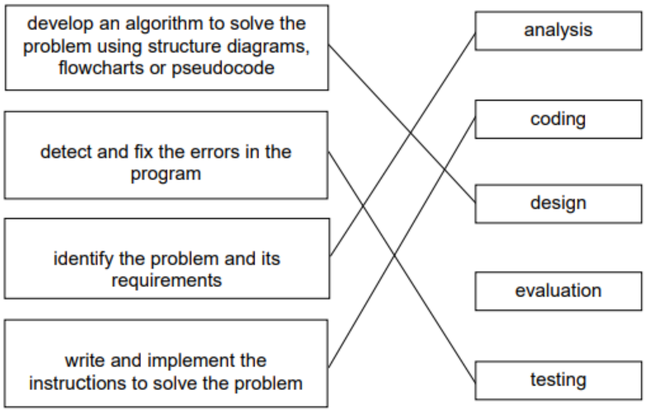
Q26)

Question	Answer	Marks
	One mark per mark point, max **two** • type of test data … • … description of test data **Example answers** Normal data (1) data that would be accepted by the program (1) Boundary / extreme data (1) data that is **on** the acceptable limits (1)	**2**

Q27)

Question	Answer	Marks
(a)(i)	- 1	1
(a)(ii)	**One** mark for each correct letter seen, max **two** - C - D	2
(b)(i)	**One** mark per mark point, max **two** - two or more digits - transposed	2
`(b)(ii)	**One** mark per mark point, max **two** - multiply each digit by a different number / its place value - before adding them together and dividing by a number	2
(c)	**One** mark per mark point, max **two** - length check - type check - presence check - format check	2

Q28)

Question	Answer	Marks
(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 detect and fix the errors in the program identify the problem and its requirements write and implement the instructions to solve the problem analysis coding design evaluation testing	4
`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	3

Q29)

Question	Answer	Marks
	A	1

Q30)

Question	Answer	Marks
(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	2
(b)	**One** mark for each appropriate test data, max **three** **One** mark for each correct accompanying reason, max **three** **For example:** Normal – 75 Reason – the data lies within the required range **and** should be accepted Abnormal – Sixty Reason – this is the wrong data type **and** should be rejected Extreme – 200 Reason – the highest value in the required range that should be accepted	6

Q31)

Question	Answer	Marks
	A	1

Q32)

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

Q33)

Question	Answer	Marks
(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	**3**
(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	**1**

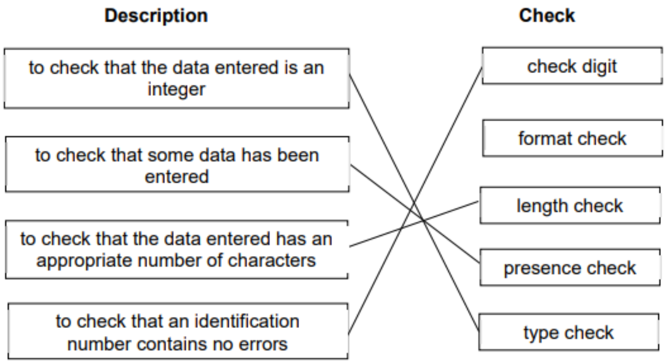
(b)(ii)	**One** mark for each point (**max three**).
	• use of iteration
	• use of two inputs
	• to check that the two inputs are the same / different
	• use of the given variable `Measurement`
	
	For example
	`REPEAT`
	`    OUTPUT "Please enter measurement "`
	`    INPUT Measurement`
	`    OUTPUT "Please re-enter measurement "`
	`    INPUT MeasurementCheck`
	`UNTIL Measurement = MeasurementCheck`
		3

Q34)

Question	Answer	Marks
(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	**2**

Question	Answer	Marks
(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.	**4**

Q35)

Question	Answer	Marks
(a)	**One** mark for each correct line.  **Description** to check that the data entered is an integer to check that some data has been entered to check that the data entered has an appropriate number of characters to check that an identification number contains no errors **Check** check digit format check length check presence check type check	**4**

Question	Answer	Marks
(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	**3**

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
```

Q36)

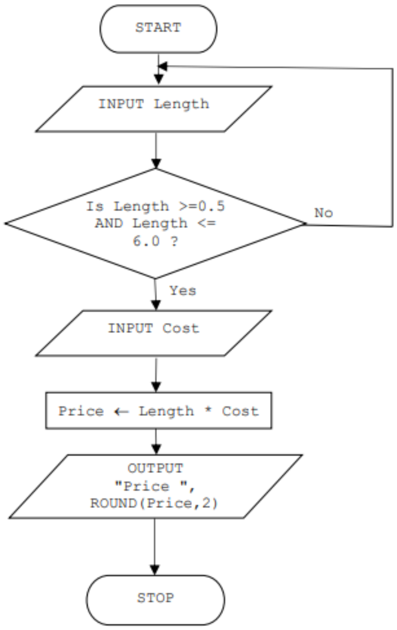
Question	Answer	Marks
	One mark for each correct word • array • constant • variable	**3**

Q37)

Question	Answer	Marks
(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	**2**
(b)	One mark for each point (max three) • inputs • processes • outputs • storage	**3**
(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)	**2**

Q38)

Question	Answer	Marks
(a)	Range check	1

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(b)</td><td>
START
INPUT Length
Is Length >=0.5 AND Length <= 6.0 ?
No
Yes
INPUT Cost
Price ← Length * Cost
OUTPUT "Price ", ROUND(Price,2)
STOP

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</td><td>6</td></tr>
<tr><td>(c)</td><td>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)</td><td>4</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(d)</td><td>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)
<table><tr><th>Length</th><th>Cost</th><th>Price</th><th>OUTPUT</th></tr></table></td><td>3</td></tr>
<tr><td>(e)</td><td>**One** mark for each point (max two)
• validate Cost …
• … with a range/presence check
• add another validation check for Length</td><td>2</td></tr>
</table>

Q39)

Question	Answer	Marks
	A	**1**

Q40)

Question	Answer	Marks
(a)	Format check	**1**
(b)	**One** mark for each appropriate test data, max **two** **One** mark for each correct accompanying reason, max **two** **For example:** Normal – 30/12/1960 … Reason – … (the date is written in the correct format and) **should be accepted**. Abnormal – 30/Dec/1960 … Reason – … (the month is not written in the correct format and) **should be rejected**.	**4**
(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**	**2**

Q41)

Question	Answer	Marks
	A	**1**

Q42)

Question	Answer	Marks
(a)	**One** mark for each point • type check • range check	**2**

(b)	**One** mark for each point (max five)
	• use of loop for check
	• checking for whole number
	• checking for number greater than or equal to one
	• … and less than or equal to six
	• Appropriate error/reinput message
	• ability to reinput value
	
	Example:
	`WHILE Seats < 1 OR Seats > 6 OR Seats <> ROUND(Seats, 0) DO`
	`    OUTPUT "Please enter a valid number of seats "`
	`    INPUT Seats`
	`ENDWHILE`
		5

(c)	**One** mark for correct test data, **one** mark for corresponding reason Example: 7, abnormal data to show that this value would be rejected	**2**

Q43)

Question	Answer	Marks
	D	**1**

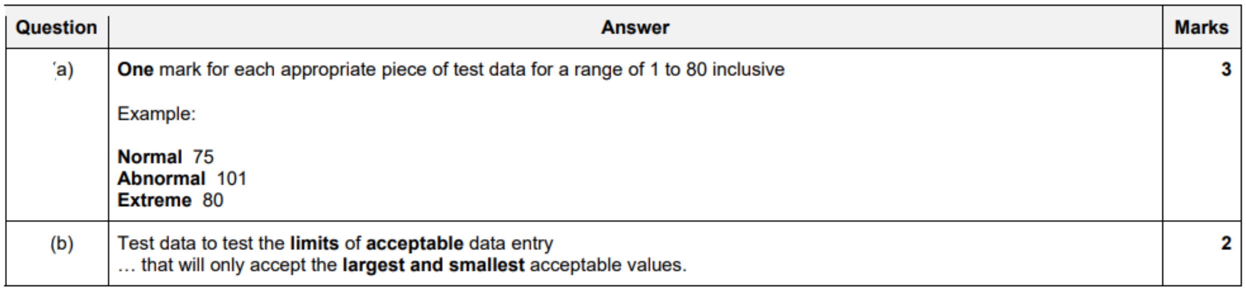
Q44)

Question	Answer	Marks
	C	**1**

Q45)

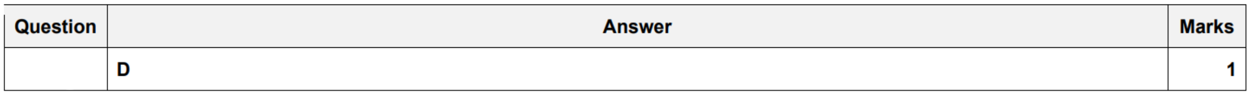
Question	Answer	Marks
	One mark for each correct answer **max three** MP1 abstraction MP2 decomposition MP3 identification of problem MP4 identification of requirements // outline of success criteria	**3**

Q46)



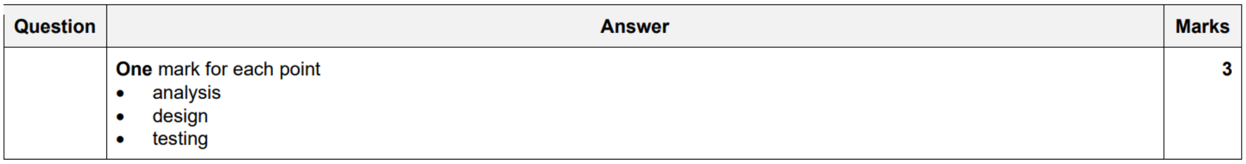
Question	Answer	Marks
(a)	**One** mark for each appropriate piece of test data for a range of 1 to 80 inclusive Example: **Normal** 75 **Abnormal** 101 **Extreme** 80	3
(b)	Test data to test the **limits** of **acceptable** data entry … that will only accept the **largest and smallest** acceptable values.	2

Q47)



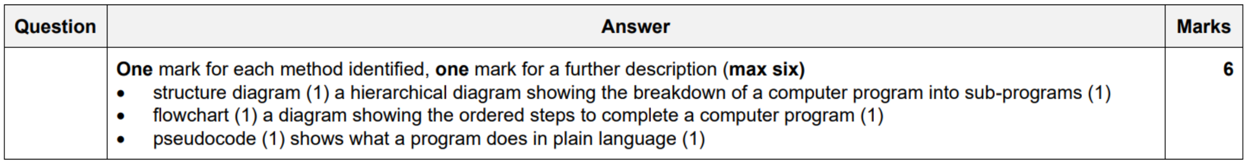
Question	Answer	Marks
	D	**1**

Q48)



Question	Answer	Marks
	One mark for each point • analysis • design • testing	**3**

Q49)



Question	Answer	Marks
	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)	**6**

Q50)

Question	Answer	Marks
	C	**1**

Q51)

Question	Answer	Marks
	C	**1**

Q52)

Question	Answer	Marks
(a)	**One** mark per mark point (**max one**) • Design • Coding • Testing	**1**
(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	**3**

Q53)

Question	Answer	Marks
	One mark for naming the type of check and **one** mark for an expansion (**max two**) • Visual check • … looking at the data that has been entered and either confirming it is correct or showing / correcting errors. **OR** • Double entry check // Data entered twice • … data is entered twice and the two sets of data are **compared** (by the computer). If they don't match, an error has been input, so re-entry is requested.	**2**

Q54)

Question	Answer	Marks
	One mark per mark point (**max four**) • 80 • The largest whole number that would **be accepted / at the very limit** / Boundary/Extreme data that would be **accepted / at the very limit** • 81 • The smallest whole number that would be **rejected / is greater than the limit** / Boundary/Abnormal/Erroneous data that would be **rejected / is greater than the limit**	**4**

Q55)

Question	Answer	Marks
	C	**1**

Q56)

Question	Answer	Marks
	B	**1**

Q57)

Question	Answer	Marks
	One mark for stage, **one** mark for matching description and **one** mark for matching expansion (**max six**) For example: • design (1) construction of a solution (1) using standard methods e.g. flowcharts (1) • coding (1) program is written (1) iterative testing takes place (1) • testing (1) program is tested for errors (1) program is tested that it meets its requirements (1)	**6**

Pseudocode

Q58)

1 mark for each error identified + suggested correction

Line 1 or `Small = 0`: this should read **`Small = 999`**

line 5 or `IF`...: this should read **`IF Num < Small THEN Small = Num`**

line 8 or `UNTIL`: this should read **`UNTIL Counter = 10`** or
`UNTIL Counter > = 10` or
`UNTIL Counter > 9`

line 7 or `PRINT`...: **`PRINT Small`** should come after the end of the repeat loop

or

line 8 or `UNTIL`: this should come before line 7 [4]

Q59)

Any **two** points from

- a variable is used to store data that can change during the running of a program
- a constant is used to store data that will not be changed during the running of a program [2]

Q60)

1 mark for each error identified + suggested correction

Line 1 or `Large =9999`: this should read **`Large = 0`**

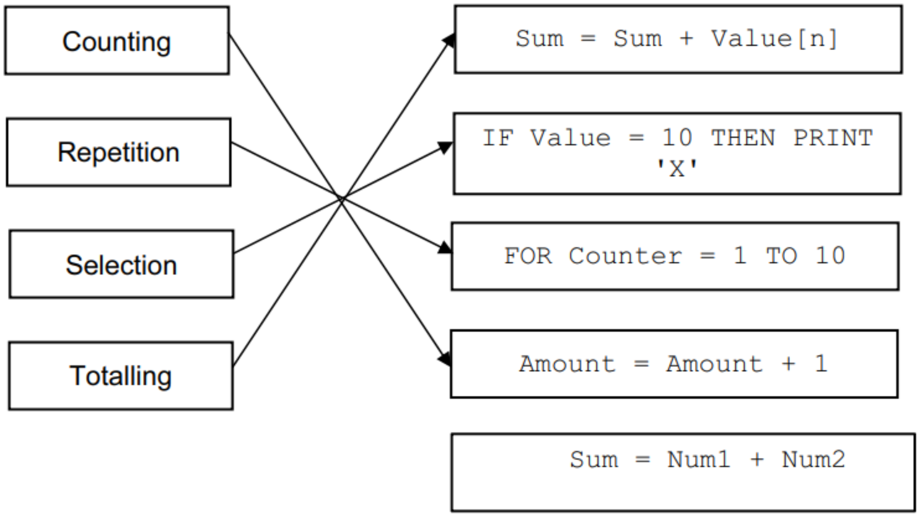
Line 3 or `WHILE`: this should read **`WHILE Counter < 30`**

line 6 or `IF`: this should read **`IF Num > Large THEN Large = Num`**

line 7 or `Counter =`...: this should read **`Counter = Counter + 1`** [4]

Q61)

1 mark for each correct line, two lines from one box not allowed



[4]

Q62)

One mark for each error identified + suggested correction
line 4 or (`Total =) Total + 1`: this should read (`Total =) Total + Num`
line 5 or `Counter = Counter + 1`: delete this line
line 6 or (`Average = )Total / Counter`: swap lines 6 and 7
line 6 or (`Average = )Total / Counter` : this should read (`Average =) Total / 50`

[4]

Q63)

There are many possible correct answers this is an example only.

Normal	e.g. 1.7
Extreme	0.5 **or** 2.0 only
Abnormal	e.g. one

[3]

Q64)

– `IF (… THEN … ELSE … ENDIF)`
– `CASE (… OF … OTHERWISE … ENDCASE)` [2]

Q65)

One mark for each error identified + suggested correction
line 5 or `IF Num < 0`: this should read `IF Num > 0 (THEN Total = Total + Num)`

line 6 or (`IF Num > 0`) `THEN Counter = Counter + 1`:
this should read (`IF Num > 0 THEN`) `Poscount = Poscount + 1`

line 7 `Average = Total/Poscount`: this should come after the end of the repeat loop

line 9 or `PRINT Num`: this should read `PRINT Average` [4]

Q66)

(a) **(i)** Normal

(ii) Acceptable data to test that the results are as expected. [2]

(b) **One** mark for the data set, **one** mark for the type and **one** mark for the matching reason
There are many possible correct answers this is an example only.
Set 1 – Age 4, height 0.9
Type – Boundary/Extreme
Reason – Data to test the validation that is just within the limits of acceptability
Set 2 – Age 10, height 1.4
Type – Abnormal
Reason – Data that should be rejected and produce an error message [6]

Q67)

(i) **1** mark for each change

Change variable name in every instance as needs to be meaningful e.g. `Large`
Set this variable to a low value
line 5: change comparison from < to > [3]

(ii) **3** marks maximum, 1 mark for each change correctly included.

```
1   Large = 0
2   Counter = 0
3   REPEAT
4     INPUT Num
5     IF Num > Large THEN Large = Num
6     Counter = Counter + 1
7   UNTIL Counter = 10
8   PRINT Large
```

[3]

Q68)

(i) **1** mark for each improvement

use `FOR … NEXT` instead of `REPEAT … UNTIL`
Move `PRINT` to after the end of the loop
Add error checking to check that the value input is positive [3]

(ii) **3** marks maximum, **1** mark for each improvement correctly included.

Sample answer below

```
Total = 0
  FOR Counter = 1 To 10
   REPEAT
    INPUT Num
  UNTIL Num >0
    Total = Total + Num
NEXT Counter
PRINT Total
```

[3]

Q69)

(a) 1 mark for each change

- Line 2: `OutRange = 0`
- Line 6: should be `OutRange = OutRange + 1`
- Line 7: not needed
- Line 8: `NEXT X` should be `NEXT Count` / Line 3: `FOR Count = 1 TO 10` should be `FOR X = 1 TO 10` [4]

(b)

Number	Within range (✓)	Outside range (✓)	Reason
10		✓	Range greater than 10, so 10 not included
20		✓	Range less than 20, so 20 not included

[4]

Q70)

1 mark for identifying each error, 1 mark for the corresponding change

– line 2 or `Counter = 100`
– `Counter = 0`

– line 6 or `UNTIL Num < 0`
– `UNTIL Num >= 0`

– line 7 or `Total = Total + 1`
– `Total = Total + Num`

– line 8 or `Counter = Counter + Num`
– `Counter = Counter + 1` [8]

Q71)

Question	Answer	Marks
	1 mark for each error identified and suggested correction (the corrected code must be written in full) *Line 2 Correct code* `Counter = 0` (1) *Line 7 Correct code* `Total = Total + Number` // `Number + Total` (1) *Line 8 Correct code* `Counter = Counter + 1` // `1 + Counter` (1) *Line 10 Correct code* `Average = Total / Counter` // `Average = Total / 50` (1)	**4**

Q72)

Question	Answer	Marks

(a)	award full marks for any working solution
	- Input three numbers (1)
	- Attempt to select largest number (1)
	- Working method (1)
	- print out largest number (1)
	Sample algorithm
	`INPUT Num1, Num2, Num3`
	`IF (Num1 > Num2) AND (Num1 > Num3) THEN PRINT Num1`
	`ENDIF`
	`IF (Num2 > Num1) AND (Num2 > Num3) THEN PRINT Num2`
	`ENDIF`
	`IF (Num3 > Num1) AND (Num3 > Num2) THEN PRINT Num3`
	`ENDIF`
	or
	`INPUT Num1`
	`Big ← Num1`
	`INPUT Num2, Num3`
	`IF Num2 > Big THEN Big ← Num2 ENDIF`
	`IF Num3 > Big THEN Big ← Num3 ENDIF`
	`PRINT Big`
		4

(b)	1 mark for each data set and 1 mark for the matching reason. There are many possible correct answers, these are examples only. *Test data set 1:* 30, 29, 28 *Reason:* first number is the largest Test data set 2: x, y, z Reason: abnormal data, should be rejected Max 4 marks	4

Q73)

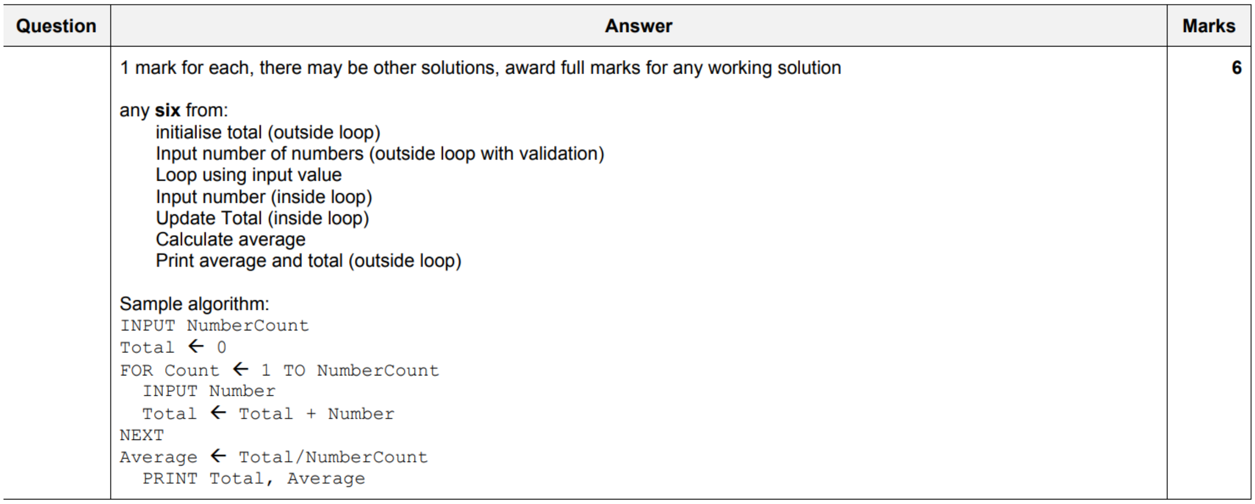
Question	Answer	Marks
(a)	Error - `Count ← 0` Correction - `Count ← 1` or Error - `UNTIL Count > 100` Correction - `UNTIL Count >= 100` or `UNTIL Count = 100` or `UNTIL Count > 99`	**2**

(b)	- use of `FOR` with correct start and end values …
	- … use of `NEXT`
	- … removal of increment for `Count`
	Sample algorithm
	`Sum ← 0`
	`FOR Count ← 1 TO 100`
	`  INPUT Number`
	`  Sum ← Sum + Number`
	`NEXT // NEXT Count`
	`PRINT Sum`
		3

Q74)

Question	Answer	Marks
	1 mark for each error identified plus suggested correction (the corrected lines must be written in full) Line 4 **correct line** `WHILE Number <= 99 OR Number > 1000` Line 7 **correct line** `Num[Index] = Number` Line 9 **correct line** `NEXT (Index)` Line 10 **correct line** `PRINT Count`	**4**

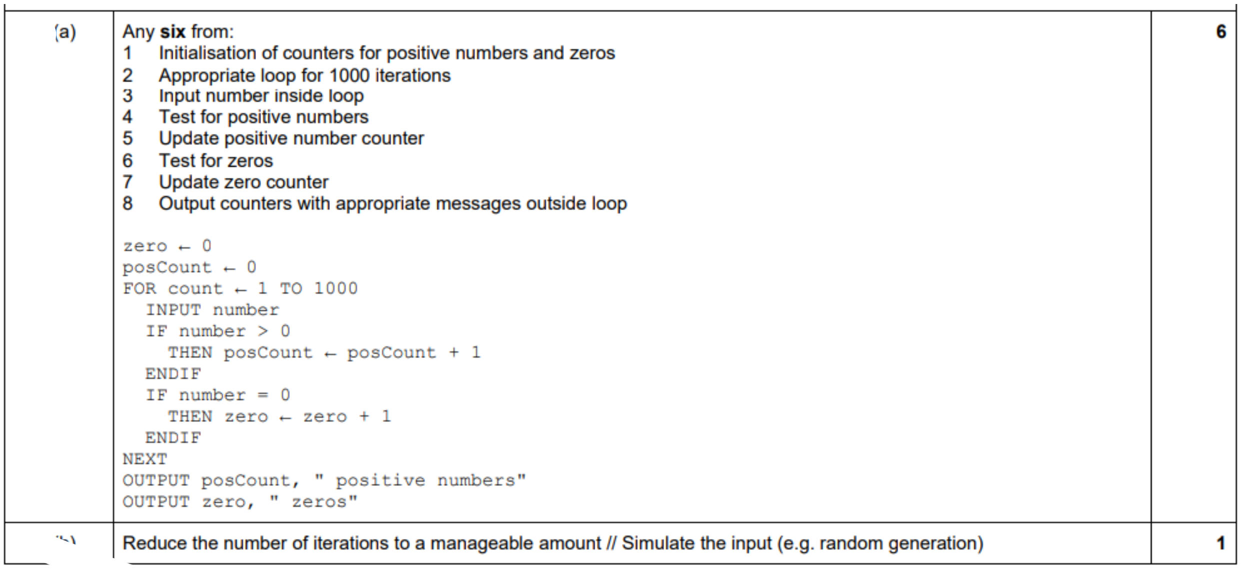
Q75)



Question	Answer	Marks

	1 mark for each, there may be other solutions, award full marks for any working solution
	
	any **six** from:
	initialise total (outside loop)
	Input number of numbers (outside loop with validation)
	Loop using input value
	Input number (inside loop)
	Update Total (inside loop)
	Calculate average
	Print average and total (outside loop)
	
	Sample algorithm:
	`INPUT NumberCount`
	`Total ← 0`
	`FOR Count ← 1 TO NumberCount`
	`  INPUT Number`
	`  Total ← Total + Number`
	`NEXT`
	`Average ← Total/NumberCount`
	`  PRINT Total, Average`
		6

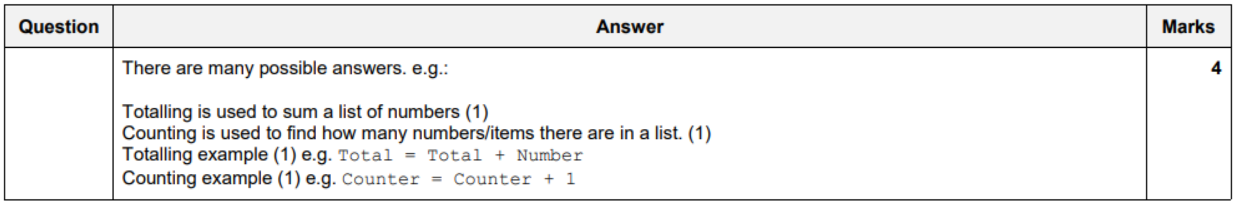
Q76)



(a)	Any **six** from:
	1 Initialisation of counters for positive numbers and zeros
	2 Appropriate loop for 1000 iterations
	3 Input number inside loop
	4 Test for positive numbers
	5 Update positive number counter
	6 Test for zeros
	7 Update zero counter
	8 Output counters with appropriate messages outside loop
	
	`zero ← 0`
	`posCount ← 0`
	`FOR count ← 1 TO 1000`
	`  INPUT number`
	`  IF number > 0`
	`    THEN posCount ← posCount + 1`
	`  ENDIF`
	`  IF number = 0`
	`    THEN zero ← zero + 1`
	`  ENDIF`
	`NEXT`
	`OUTPUT posCount, " positive numbers"`
	`OUTPUT zero, " zeros"`
		6

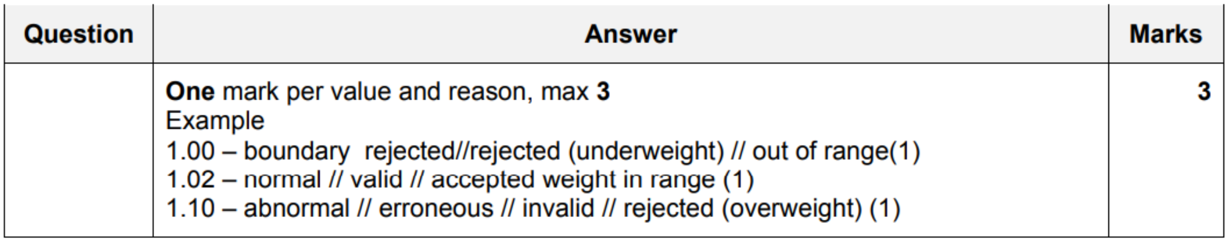
(b)	Reduce the number of iterations to a manageable amount // Simulate the input (e.g. random generation)	**1**

Q77)



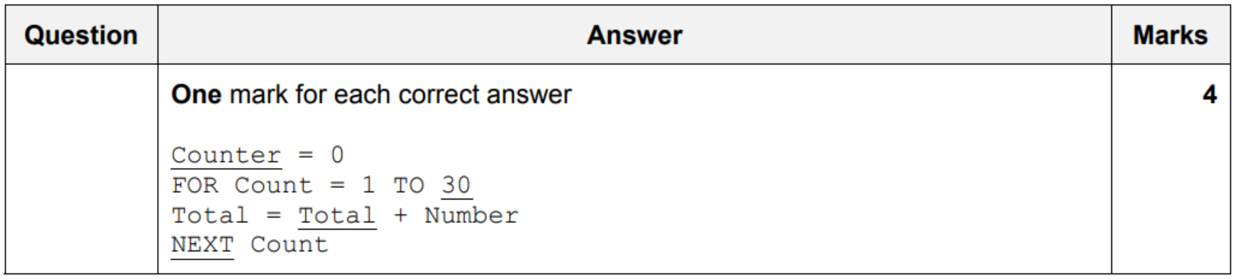
Question	Answer	Marks
	There are many possible answers. e.g.: Totalling is used to sum a list of numbers (1) Counting is used to find how many numbers/items there are in a list. (1) Totalling example (1) e.g. `Total = Total + Number` Counting example (1) e.g. `Counter = Counter + 1`	**4**

Q78)



Question	Answer	Marks
	One mark per value and reason, max **3** Example 1.00 – boundary rejected//rejected (underweight) // out of range(1) 1.02 – normal // valid // accepted weight in range (1) 1.10 – abnormal // erroneous // invalid // rejected (overweight) (1)	**3**

Q79)



Question	Answer	Marks

	One mark for each correct answer
	
	`Counter = 0`
	`FOR Count = 1 TO 30`
	`Total = Total + Number`
	`NEXT Count`
		4

Q80)

Question	Answer	Marks
(a)	Any **two** from: - Expects a number to be input - Checks if the number is greater than 100 - Outputs the result of the test - Specific output example	**2**
(b)(i)	**One** mark for correct answer e.g. Use a (condition controlled) loop	**1**

(b)(ii)	**One** mark for each point
	- Initialisation of Number variable
	- Correct loop statements
	- Correct INPUT and OUTPUT
	e.g.
	`INPUT Number`
	`WHILE Number > 100 DO`
	`  OUTPUT "The number is too large"`
	`  INPUT Number`
	`ENDWHILE`
	`OUTPUT "The number is acceptable"`
	or
	`INPUT Number`
	`REPEAT`
	`  IF Number > 100`
	`    THEN`
	`      OUTPUT "The number is too large"`
	`  ENDIF`
	`  INPUT Number`
	`UNTIL Number <= 100`
	`OUTPUT "The number is acceptable"`
		3

Q81)

Question	Answer	Marks
(a)	1 mark for each point: ∞ Expects 50 numbers to be input ∞ Totals the numbers as they are entered / carries out a running total ∞ Outputs the result after the numbers have all been entered	3

(b)	1 mark for each point (max 3 marks):
	∞ Correct initialisation of counter for REPEAT or WHILE loop …
	∞ … Correct loop statements and counter increment
	∞ … Correct statements inside loop
	∞ … Correct statements outside loop
	e.g.
	`Total ← 0`
	`Count ← 1`
	`WHILE Count <= 50 DO`
	`  INPUT Num`
	`  Total ← Total + Num`
	`  Count ← Count + 1`
	`ENDWHILE`
	`OUTPUT Total`
	Accept alternative correct ranges e.g.
	`Count ← 0`
	`WHILE Count < 50 DO`
	`Total ← 0`
	`Count ← 0`
	`REPEAT`
	`  INPUT Num`
	`  Total ← Total + Num`
	`  Count ← Count + 1`
	`UNTIL Count = 50`
	`OUTPUT Total`
	Accept alternative correct ranges e.g.
	`Count ← 1`
	`UNTIL Count > 50`
		3

(c)	1 mark for each correct point in description, e.g. ∞ Use a variable for the counter upper limit… ∞ … that is input by the user. or ∞ loop using a condition control … ∞ … until condition is met	2

Q82)

(a)	Any **five** from:
	1 Use of correct variables
	2 Input 3 numbers
	3 Check all 3 input numbers are different
	4 Attempt to find the largest two numbers input
	5 Correctly finding the largest two numbers
	6 Multiply their two largest numbers together and assign to variable
	7 Output the result of the multiplication
	
	Sample answer:
	`REPEAT`
	`   OUTPUT "Enter three different numbers"`
	`   INPUT Number1, Number2, Number3`
	`UNTIL Number1 <> Number2 AND Number2 <> Number3 AND Number3 <> Number1`
	`IF Number3 < Number2 AND Number3 < Number1`
	`   THEN Answer ← Number1 * Number2`
	`ENDIF`
	`IF Number2 < Number3 AND Number2 < Number1`
	`   THEN Answer ← Number1 * Number3`
	`ENDIF`
	`IF Number1 < Number2 AND Number1 < Number3`
	`   THEN Answer ← Number2 * Number3`
	`ENDIF`
	`OUTPUT "Answer = ", Answer`
		5

(b)	There are many correct answers. E.g.: 7, 7, 7 (1 mark) … should be rejected as numbers are equal (1 mark) 7, 8, 9 (1 mark) … normal data answer should be 72 (1 mark)	**4**

Q83)

Question	Answer	Marks
(a)	1 mark for each error identified plus suggested correction Line 1 or `Total = 100.00`: **correction** Total = `0(.00)` Line 8 or `Count = Count + 1`: **correction** This line should be removed (not required in a FOR loop) // use of `REPEAT…UNTIL` or `WHILE…DO…ENDWHILE` Line 7 or `PRINT Total /30`: **correction** This line should be outside the loop (or it will print each iteration)	**3**
(b)	1 mark for correct purpose: Find/output average height	**1**

Q84)

(a) 1 mark for each error identified + suggested correction

∞ `Low ← Count` should be `Low ← Number`

∞ `Number > Low` should be `Number < Low`

∞ `UNTIL Count = 99` should be `UNTIL Count > 99` or `UNTIL Count = 100` or `UNTIL Count >= 100` // `Count ← 1` should be `Count ← 0`

∞ `PRINT "Largest Number is ", Number` should be `PRINT "Largest Number is ", High`

4

(b)

MP1 Add `Total ← 0` // `Total ← Number`
MP2 Add `Total ← Total + Number`
MP3 Add `PRINT "Total is ", Total`
MP4 All positioning explained / seen

```
Count ← 1
INPUT Number
High ← Number
Low ← Number
Total ← Number
REPEAT
    INPUT Number
    Total ← Total + Number
    IF Number > High
    THEN
        High ← Number
    ENDIF
    IF Number < Low
    THEN
        Low ← Number
    ENDIF
    Count ← Count + 1
UNTIL Count > 99
PRINT "Largest Number is ", High
PRINT "Smallest Number is ", Low
PRINT "Total is ", Total
```

4

Q85)

(a)	1 mark for each error identified + suggested correction
	Count ← 1 should be Count ← 0 or Count >= 500 should be Count > 500
	AND should be OR
	Reject ← Reject – 1 should be Reject ← Reject + 1
	Reject ← Reject/100 should be Reject ← Reject/5 or Reject * 100 / 500
		4

(b)	MP1 Add Accept ← 0 at start
	MP2 Add ELSE Accept ← Accept + 1 after THEN AND Over and Under defined/position described
	OR
	Add Accept ← Accept + 1 after THEN AND Replace IF statement with …<= Over AND …>= UNDER… /position described
	MP3 Add Accept ← Accept/5 after UNTIL AND correct loop/position described
	MP4 ADD IF Accept < 50 THEN PRINT "Less than 50% accepted" at end
	Accept ← 0
	Count ← 1 // 0
	Reject ← 0
	Over ← 62
	Under ← 58
	REPEAT
	    INPUT ItemWeight
	    IF ItemWeight > Over OR ItemWeight < Under // IF ItemWeight <= Over AND
	    ItemWeight >= Under
	        THEN
	            Reject ← Reject + 1
	        ELSE
	            Accept ← Accept + 1//ELSE not required
	    ENDIF
	    Count ← Count + 1
	UNTIL Count > 500 // >= 500
	Accept ← Accept / 5
	
	IF Accept < 50
	    THEN
	        PRINT "Less than 50% accepted"
	ENDIF
		4

Q86)

Question	Answer	Marks
(a)	• Inputs a series of values • Finds the total • Prints out the average	**3**
(b)	**Three** from: • Use of loop structure • Allow input to define the limit of the loop / use sentinel value • Keeping a count of the number of values • It could use a totalling process to keep a running total	**3**

(c)	Marks awarded as follows (maximum five marks):
	• Initialise Total
	• Enter limit
	• Suitable loop structure
	• Correct input
	• Correct totalling
	• Correct output
	e.g.
	
	`Total ← 0`
	`INPUT CounterLimit`
	`FOR LoopCounter ← 1 To CounterLimit`
	`    INPUT Number`
	`    Total ← Total + Number`
	`NEXT LoopCounter`
	`OUTPUT "The average equals ", Total / CounterLimit`
		5

Q87)

Question	Answer	Marks
	One mark for each error identified and suggested correction: ∞ `IF TotalTry > Number` should be `IF Guess > Number` ∞ `IF Guess > Number` should be `IF Guess < Number` ∞ `TotalTry ← Guess + 1` should be `TotalTry ← TotalTry + 1` ∞ `UNTIL Guess <> Number` should be `UNTIL Guess = Number`	**4**

Q88)

Question	Answer	Marks
	Totalling: ∞ Adding the weight of each basket to the total weight as each weight is entered ∞ `Total = Total + Weight` Counting: ∞ Adding one to/incrementing the number of baskets as each weight is entered ∞ `BasketCount = BasketCount + 1`	**4**

Q89)

Question	Answer	Marks
(a)	• Conditional / selection statement	**1**

(b)	**Four** from:
	MP1 CASE statement with identifier Response
	MP2 Correct structure used for choices…
	MP3 …. correct statements used for choices
	MP4 OTHERWISE and correct statement
	MP5 Single ENDCASE included
	e.g.
	`CASE OF Response // CASE Response OF`
	`    1 : X ← X + Y`
	`    2 : X ← X – Y`
	`    3 : X ← X * Y`
	`    4 : X ← X / Y`
	`    OTHERWISE OUTPUT "No response"`
	`ENDCASE`
		4

Q90)

Question	Answer	Marks
(a)	Any **two** correct statements (max **two**) e.g. - The value of the variable Count begins as 0 … - … and is incremented by 1 before it is tested by the loop condition - Count will never be 0 at the end of the loop	**2**

Question	Answer	Marks

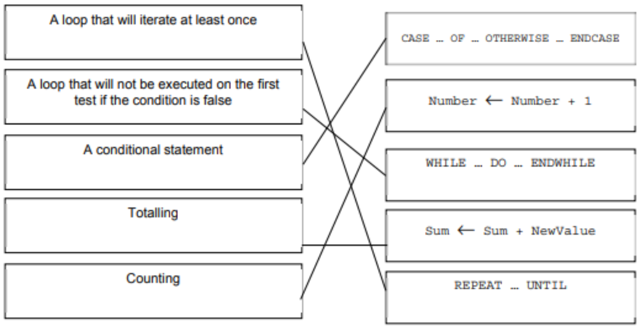
(b)	`Count ← 0`
	`REPEAT`
	`  INPUT Number`
	`  IF Number >= 100`
	`    THEN`
	`      Values[Count] ← Number`
	`  ENDIF`
	`Count ← Count + 1`
	`UNTIL Count = 50`
	
	One mark – separate INPUT statement
	One mark – IF statement attempted
	One mark – IF statement completely correct
	One mark – termination of loop updated
		4

(c)	Any **two** correct statements (max **two**) e.g. - Alter the IFstatement/add a second IF statement/comparison that's already there … - … so that additional criteria set an upper limit of <=200	**2**

Q91)

Question	Answer	Marks
(a)	**One** mark for error and correction Line 1 `HighestMark ← 0` Line 7 `INPUT Mark[Count]` Line 10 `HighestMarkStudents ← HighestMarkStudents + 1` Line 14 `HighestMark ← Mark[Count]`	**4**
(b)	Any **six** from: Add variable `LowestMark` … … Set this to a high value for example 100 Add variable `LowestMarkStudents` … … Set this to zero Check if `Mark[Count] = LowestMark` … … `True –` add 1 to `LowestMarkStudents` Check if `Mark[Count] < LowestMark` … … True – set `LowestMarkStudenta to 1` and set `LowestMark` to `Mark[Count]` Add extra output statement	**6**

Q92)

Question	Answer	Marks
	Description **Pseudocode example**  A loop that will iterate at least once A loop that will not be executed on the first test if the condition is false A conditional statement Totalling Counting CASE … OF … OTHERWISE … ENDCASE Number ← Number + 1 WHILE … DO … ENDWHILE Sum ← Sum + NewValue REPEAT … UNTIL **One** mark – **one** correct link **Two** marks – **two** correct links **Three** marks – **three** correct links **Four** marks – **four/five** correct links	**4**

Q93)

Question	Answer	Marks
	Line 2 and Line 4 errors - **One** mark for each correct identification **and** correction of error Error 1 line number: Line 2 Correction: `REPEAT` Error 2 line number: Line 4 Correction: `IF Number < 0 OR Number > 499` Line 8 error - **One** mark for correct identification of error and **one** mark for **each** correction of error Error 2 line number: Line 8 Correction: `UNTIL Number >= 0 AND Number <= 499`	**6**

Q94)

	One mark for description of variable **One** mark for description of constant **One** mark for inclusion of an example variable **One** mark for inclusion of an example constant Example answer: A value that can change during the execution of a program A named value that cannot change during the execution of a program Variable example - using a counter for example counter ← counter + 1 Constant example – a static value that can be used for checking for example < maxAge	**4**

Q95)

Question	Answer	Marks
(a)	Line 1/2/3/4/8/9 Lines 5–11 Line 9 Line 8	**4**

Question	Answer	Marks
(b)	**One** mark for error and correction Line 1 TotalWeight ← 0 Line 10 **move** OUTPUT "Number of bags in the load is ", BagCount **to end / after line 11** Line 11 UNTIL TotalWeight > MaxWeight **OR** BagCount >= MaxBag Line 12 OUTPUT "Total weight of the load is ", **TotalWeight**	**4**
(c)	Any **four** from: After line 11 Divide TotalWeight by … BagCount Assign a new variable AverageWeight ← TotalWeight / BagCount Output the result OUTPUT AverageWeight With a message "Average weight of a bag of firewood is "	**4**

Q96)

Question	Answer	Marks
	One mark for each correct type of test and **one** mark for each correct accompanying example of test data and reason (max **six**) e.g. • Extreme data • 5000 • to check it is accepted • Normal data • 300 • To check it is accepted • Abnormal data • 10000 • To check it is rejected	**6**

Q97)

(a)	Line 1/2/8/12 Line 3 and/or 14 Line 8/12 Line 6/10/15/19	**4**
(b)	**One** mark for error and correction Line 02 `TooCold ← 0` Line 08 `TooCold ← TooCold + 1` Line 15 `IF TooHot > 5` Line 17 `OUTPUT "Alarm!!"`	**4**
(c)	Any **four** from: Add a new variable `inRange` … … set to zero at start of algorithm Add extra `IF` statement `IF temperature >= -25 AND temperature <= -18` Update `inRange` by 1 if true	**4**

Q98)

Question	Answer	Marks
(a)	**One** mark per mark point **(Max 3)** MP1 Marks input are **stored in the array** `Score[]` MP2 Marks are checked against a range of boundaries // allow example MP3 … and a **matching grade** is assigned to each mark that has been input MP4 … then **stored in the array** `Grade[]` … MP5 … at the same index as the mark input MP6 The algorithm finishes after 30 marks have been input // allows 30 scores to be entered	**3**

(b)	**One** mark per mark point **(Max 3)**
	MP1 Correct loop, including counter if not a FOR loop
	MP2 Correct output of `Score[]`
	MP3 Correct output of `Grade[]`
	MP4 Suitable messages/text in output for both arrays
	
	Example answers
	`Count ← 0`
	`REPEAT`
	`  PRINT "Student: ", Count, " Mark: ", Score[Count], " Grade: ",Grade[Count]`
	`  Count ← Count + 1`
	`UNTIL Count = 30`
	
	`Count ← 0`
	`WHILE Count < 30 DO`
	`  PRINT "Student: ", Count, " Mark: ", Score[Count], " Grade: ",Grade[Count]`
	`  Count ← Count + 1`
	`ENDWHILE`
	
	`FOR Count ← 0 TO 29`
	`  PRINT "Student: ", Count, " Mark: ", Score[Count], " Grade: ", Grade[Count]`
	`NEXT`
		3

Question	Answer	Marks
(c)	Any **three** correct statements **(Max 3)** e.g. MP1 Add an input facility to allow teachers to enter the class size MP2 Add a variable to store the input class size MP3 Use the class size variable as the terminating condition for the loop MP4 Make sure the arrays are sufficiently large to accommodate the largest possible class size	**3**

Q99)

<table>
<tr>
<td>(a)</td>
<td>

Any **six** from:
MP1 Initialisation of large **and** small variables e.g. `Large ← 0 Small ← 1000`
MP2 Use of a loop for 500 entries // or 499 if initialisation done on first correct entry
MP3 Input with prompt
MP4 **Attempt** at checking the range of 1 to 999 for input
MP5 … working range check
MP6 Checking for a whole number
MP7 Selecting largest number
MP8 Selecting smallest number
MP9 Calculating the range
MP10 Outputting the largest, smallest and range with message

```
Large ← 0
Small ← 1000
FOR Count ← 1 TO 500
  REPEAT
    OUTPUT "Enter a whole number between 1 and 999"
    INPUT Number
  UNTIL Number >= 1 AND Number < 1000 AND Number = Number DIV 1
  IF Number < Small
    THEN
      Small ← Number
  ENDIF
  IF Number > Large
    THEN
      Large ← Number
  ENDIF
NEXT
Range ← Large - Small
OUTPUT "Largest number is ", Large, " Smallest number is ", Small, " Range of numbers is
      ", Range
```

</td>
<td>6</td>
</tr>
</table>

Question	Answer	Marks
(b)	**One** mark for action required and **one** mark for method used Reduce the amount of numbers entered By decreasing the final value of the loop or Remove the need to input values By using random numbers / a previously populated array	2

Q100)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td><b>Correct Algorithm 2</b>
<pre>
1   Count ← 0
2   REPEAT
3   FullScore ← 0
4   INPUT Number
         StoreLoop ← 0
5   REPEAT
6   INPUT Score
7   FullScore ← FullScore + Score
           StoreLoop ← StoreLoop + 1
8   UNTIL StoreLoop = Number
9   OUTPUT "The full score is ", FullScore
10   OUTPUT "Another set of scores (Y or N)?"
11   INPUT Another
12   IF Another = "N"
13   THEN
14   Count ← 1
15   ENDIF
16 UNTIL Count = 1
</pre></td><td></td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>`(b)</td><td><b>One</b> mark per mark point (<b>Max 4</b>)

MP1 After line 6 // replace line 6
MP2 ScoreArray[StoreLoop] ← Score //
INPUT ScoreArray[StoreLoop]

MP3 between lines 8 and 10
MP4 AverageScore ← FullScore/Number
MP5 OUTPUT "The average score is ", AverageScore

Example correct algorithm for reference from part 4(a)
<pre>
1   Count ← 0
2   REPEAT
3   FullScore ← 0
4   INPUT Number
5   FOR StoreLoop ← 1 TO Number
6   INPUT Score
7   FullScore ← FullScore + Score
8   NEXT
9   OUTPUT "The full score is ", FullScore
10   OUTPUT "Another set of scores (Y or N)?"
11   INPUT Another
12   IF Another = "N"
13   THEN
14   Count ← 1
15   ENDIF
16 UNTIL Count = 1
</pre></td><td>4</td></tr>
</table>

Q101)

(a)	Line 1 should be `Counter ← 0`
	Line 3 `RandNum[Counter] ← Rand(1, 100)` should be `RandNum[Counter] ← Rand(1, 101)`
	Line 4 `Counter ← Counter + 2` should be `Counter ← Counter + 1`
	Line 5 `UNTIL Count <= 50` should be `UNTIL Counter >= 50 // UNTIL Counter = 50`
	
	`1  Counter ← 0`
	`2  REPEAT`
	`3      RandNum[Counter} ← Rand(1, 100)`
	`4      Counter ← Counter + 1`
	`5  UNTIL Counter >= 50`
	Or
	
	Line 3 `RandNum[Counter]` should be `RandNum[Count]`
	Line 3 `Rand(1, 100)` should be `Rand(1, 101)`
	Line 4 `Counter ← Counter + 2` should be `Count ← Count + 1`
	Line 5 `UNTIL Count <= 50` should be `UNTIL Count >= 50 // UNTIL Count = 50`
	
	`1  Count ← 0`
	`2  REPEAT`
	`3      RandNum[Count} ← Rand(1, 100)`
	`4      Count ← Count + 1`
	`5  UNTIL Count >= 50`
		4

(b)	One mark for each correct line
	`FOR Count ← 0 TO 49 // FOR Count ← 1 TO 50`
	`  RandNum[Count] ← Rand(1, 101) / Rand(0, 101)`
	`NEXT // NEXT Count`
		3

(c)	Precondition loop `// WHILE … DO … ENDWHILE`	**1**

Q102)

Question	Answer	Marks
(a)	**One** mark for error identified and suggested correction (**Max three**) Line 8 `OUTPUT Value2` – should be `INPUT Value2` Line 9 `IF Operator` – should be `CASE OF Operator` Line 15 `OUTPUT "The answer is ", Value1` – should be `Answer` The loop may be corrected using a number of alternative methods: **One** mark for error identified and suggested correction (**Max two**) **Method 1** Line 1 `Continue ← 1` should be `Continue ← 0` Line 22 `UNTIL Continue = 0` should be `ENDWHILE` // Line 2 `WHILE Continue = 0` should be `REPEAT` **and** Line 22 `UNTIL Continue = 0` should be `Until Continue = 1` **OR** **Method 2** Line 2 `WHILE Continue = 0` should be `REPEAT` Line 20 `Continue ← 1` should be `Continue ← 0` // Line 1 `Continue ← 1` should be `Continue ← 0` **and** Line 22 `UNTIL Continue = 0` should be `Until Continue = 1` **OR** **Method 3** Line 2 `WHILE Continue = 0` should be `WHILE Continue = 1` Line 20 `Continue ← 1` should be `Continue ← 0` **and** Line 22 `UNTIL Continue = 0` should be `ENDWHILE`	**5**

Question	Answer	Marks
`(a)	**Corrected algorithm example 1**	

```
1 Continue ← 0
2 WHILE Continue = 0 (DO)
3   OUTPUT "Enter 1 for +, 2 for -, 3 for * or 4 for /"
4   INPUT Operator
5   OUTPUT "Enter the first value"
6   INPUT Value1
7   OUTPUT "Enter the second value"
8   INPUT Value2
9   CASE OF Operator
10    1: Answer ← Value1 + Value2
11    2: Answer ← Value1 - Value2
12    3: Answer ← Value1 * Value2
13    4: Answer ← Value1 / Value2
14  ENDCASE
15  OUTPUT "The answer is ", Answer
16  OUTPUT "Do you wish to enter more values (Yes or No)?"
17  INPUT MoreValues
18  IF MoreValues = "No"
19    THEN
20      Continue ← 1
21  ENDIF
22 ENDWHILE
```

Question	Answer	Marks

(a)	**Corrected algorithm example 2**
	`1 Continue ← 1`
	`2 REPEAT`
	`3   OUTPUT "Enter 1 for +, 2 for -, 3 for * or 4 for /"`
	`4   INPUT Operator`
	`5   OUTPUT "Enter the first value"`
	`6   INPUT Value1`
	`7   OUTPUT "Enter the second value"`
	`8   INPUT Value2`
	`9   CASE OF Operator`
	`10    1: Answer ← Value1 + Value2`
	`11    2: Answer ← Value1 - Value2`
	`12    3: Answer ← Value1 * Value2`
	`13    4: Answer ← Value1 / Value2`
	`14  ENDCASE`
	`15  OUTPUT "The answer is ", Answer`
	`16  OUTPUT "Do you wish to enter more values (Yes or No)?"`
	`17  INPUT MoreValues`
	`18  IF MoreValues = "No"`
	`19    THEN`
	`20      Continue ← 0`
	`21  ENDIF`
	`22 UNTIL Continue = 0`	

Question	Answer	Marks

(b)	**One** mark per bullet
	MP1 Appropriate loop (begin and end) / otherwise selection
	MP2 Testing both ends of condition
	MP3 Suitable message
	MP4 Input/re-input
	
	`WHILE Operator < 1 OR Operator > 4 (DO)`
	`  OUTPUT "Enter 1, 2, 3 or 4"`
	`  INPUT Operator`
	`ENDWHILE`
	
	Alternative answer
	`REPEAT`
	`  IF Operator < 1 OR Operator > 4`
	`    THEN`
	`      OUTPUT "Enter 1, 2, 3 or 4"`
	`      INPUT Operator`
	`  ENDIF`
	`UNTIL Operator >= 1 AND Operator <= 4`
	
	One mark
	
	After line 4 / between lines 2 and 5
		5

Q103)

(a)	**One** mark per mark point, max **three** • line 8 / `PassCheck ← TRUE` correction `PassCheck ← FALSE` • line 12 / `IF Password <> Password` correction `IF Password2 <> Password` // `IF Password <> Password2` • line 18 / `UNTIL PassCheck OR Attempt <> 3` correction `UNTIL PassCheck OR Attempt = 3 / UNTIL PassCheck OR Attempt >= 3`	**3**
(b)	**One** mark check, **one** mark matching description, max **four** Check: validation // length check Description length check // checks number of characters in **password** Check: **verification** // **double entry** Description double entry // comparison that two inputs are the same	**4**
(c)	**One** mark per set, **one** mark matching reason, max **four** Set 1 – any appropriate example e.g. "small" Reason must follow through from the password given e.g. abnormal data will be rejected Set 2 – any different appropriate example e.g. "password" and "password" Reason must be different and follow through from the password given e.g. normal data will be accepted	**4**

Q104)

Question	Answer	Marks
	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	**4**

Q105)

(a)	**One** mark per mark point, max **six** • Line 1 `100` • Line 7 `Value > 100 // Value >= 101` • Line 11 `Reading[Value] + 1` • Line 14 `INPUT Value` • Line 18 `Reading[Count]` • Line 19 `Count - 1`	6

(b)	**One** mark per mark point, max **three**
	• use an IF/conditional statement
	• to check if Reading[Count] not equal to zero
	• before outputting the value // between statements 17 and 18 // code sample showing position
	IF Reading[Count] <> 0
	  THEN
	    OUTPUT
	ENDIF
		3

Q106)

Question	Answer	Marks
(a)	**One** mark per mark point, max **four** • Line 09 / `Higher[HighList] ← MarksEntry` should be `Higher[HighList] ← Mark` • Line 15 / `MidList ← MidList` should be `MidList ← MidList + 1` • Line 17 / `Lower[HighList] ← Mark` should be `Lower[LowList] ← Mark` • Line 22 / `NEXT MarksEntry = 500` should be `UNTIL MarksEntry = 500`	4

Corrected algorithm

```
01 HighList ← 0
02 MidList ← 0
03 LowList ← 0
04 MarksEntry ← 0
05 REPEAT
06   INPUT Mark
07   IF Mark >= 80
08     THEN
09       Higher[HighList] ← Mark
10       HighList ← HighList + 1
11     ELSE
12       IF Mark >= 50
13         THEN
14           Middle[MidList] ← Mark
15           MidList ← MidList + 1
16         ELSE
17           Lower[LowList] ← Mark
18           LowList ← LowList + 1
19       ENDIF
20   ENDIF
21   MarksEntry ← MarksEntry + 1
22 UNTIL MarksEntry = 500
23 OUTPUT "You entered ", HighList, " higher marks"
24 OUTPUT "You entered ", MidList, " middle marks"
25 OUTPUT "You entered ", LowList, " lower marks"
```

Question	Answer	Marks

(b)	**One** mark per mark point, max **four**
	
	MP1 Set up a condition to end the input
	MP2 The correct placement of the condition
	MP3 Set up the test
	MP4 The correct placement of the test
	MP5 Removal of `MarksEntry` counter from the original algorithm
	
	Example answers
	
	Testing at the end of the algorithm
	`OUTPUT "Do you want to enter another mark?"`
	`INPUT AnotherMark`
	`UNTIL AnotherMark = "No"`
	should replace line 22 at end of loop
	The `MarksEntry` counter can be removed // Lines 4 and 21 are not required / can be removed
	
	Testing at the beginning of the algorithm
	`AnotherMark = "Yes"`
	`WHILE AnotherMark = "Yes" DO`
	should replace line 05 at the start of the loop
	`OUTPUT "Do you want to enter another mark?"`
	`INPUT AnotherMark`
	`ENDWHILE`
	should replace line 22 at end of loop
	The `MarksEntry` counter can be removed // Lines 4 and 21 are not required / can be removed
	
	Terminal condition
	`OUTPUT "Enter -1 to end the program"`
	should be placed before the loop and / or before the input in 06
	`IF MARK <> -1 THEN`
	should be placed between lines 06 and 07
	The `MarksEntry` counter can be removed // Lines 4 and 21 are not required / can be removed
	`UNTIL Mark = -1` should be placed at line 22
		4

Q107)

(a)	**One** mark per mark point in the correct position, max **six** • line 01 `50` • line 08 `Value > 50` • line 12 `Reading[Value] + 1` • line 18 `INPUT Value` • line 23 `Reading[Count]` • line 24 `Count + 1`	**6**

(b)	**One** mark per place in code **and** action, max **three**
	• line 21 set Count to 50 / `Count ← 50`
	• line 24 subtract 1 from Count / `Count ← Count -1`
	• line 25 check for Count equal to 34 / check for Count less than 35 / `UNTIL Count = 34` / `UNTIL Count < 35`
	or
	One mark per place in code **and** action, max **three**
	• line 21 set up FOR loop stating at 50 and finishing at 35 / `FOR Count ← 50 TO 35 STEP -1`
	• Remove lines 22 and 24
	• line 25 End FOR loop / `NEXT Count`
	
	Examples
	`21 Count ← 50`
	`22 REPEAT`
	`23 OUTPUT "There are ", Reading[Count], " readings, " of ", Count`
	`24 Count ← Count -1`
	`25 UNTIL Count = 34`
	
	`21 FOR Count ← 50 TO 35 STEP -1`
	`23 OUTPUT "There are ", Reading[Count], " readings, " of ", Count`
	`25 NEXT Count`
		3

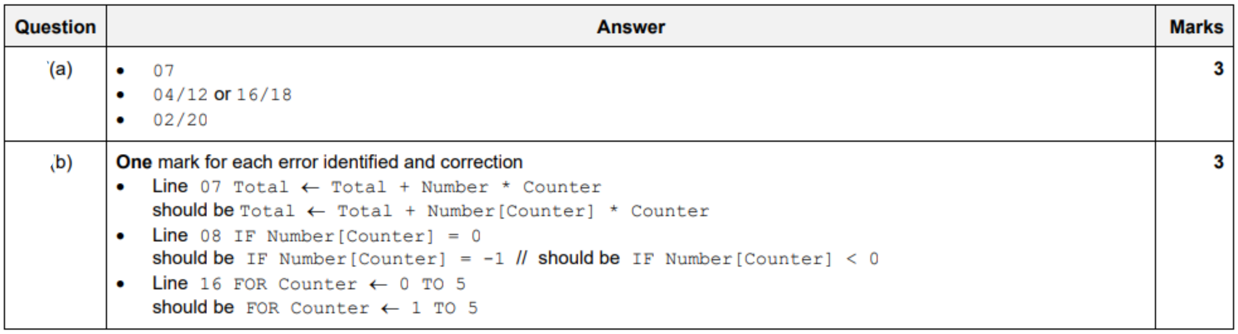
Q108)

Question	Answer	Marks

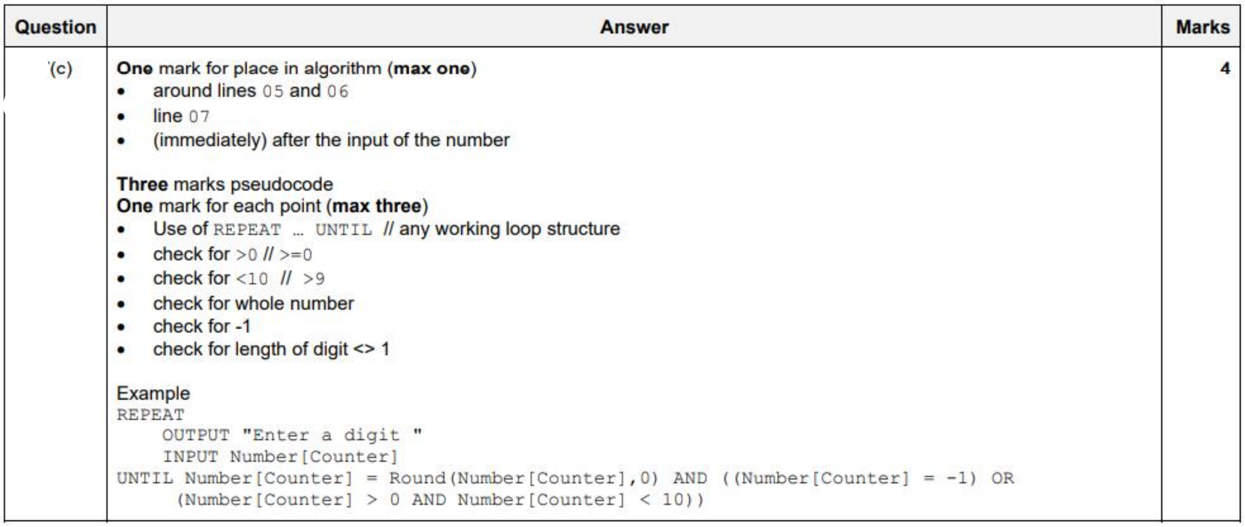
a)	**One** mark per mark point, max **four**
	• Line 04 / `IF Number < 0`
	should be `IF Number > 0`
	• Line 10 / `Exit ← 1` // Line 01/ `Exit ← 1` **and** Line 02 / `WHILE Exit <> 0`
	should be `Exit ← 0` // should be `Exit ← 0` **and** `WHILE Exit = 0`
	• Line 13 / `ENDIF`
	should be `ENDWHILE`
	• Line 14 / `OUTPUT "The total value of your numbers is ", Number`
	should be `OUTPUT "The total value of your numbers is ", Total`
	Correct algorithm:
	`01 Exit ← 1`
	`02 WHILE Exit <> 0 DO`
	`03     INPUT Number`
	`04     IF Number > 0`
	`05       THEN`
	`06         Total ← Total + Number`
	`07       ELSE`
	`08         IF Number = 0`
	`09           THEN`
	`10             Exit ← 0`
	`11         ENDIF`
	`12     ENDIF`
	`13 ENDWHILE`
	`14 OUTPUT "The total value of your numbers is ", Total`
		4

Question	Answer	Marks
(b)	**One** mark per mark point, max **four** • Initialise a new (counting) variable • … `Count ← 0` // to count the acceptable numbers • Insert a counting statement between lines 05 and 07 • … `Count ← Count + 1` • Add a new output after the loop/after line 13 / at the end (of the program) • … `OUTPUT Count`	4

Q109)



Question	Answer	Marks
(a)	• 07 • 04/12 or 16/18 • 02/20	3
(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	3



Question	Answer	Marks
(c)	**One** mark for place in algorithm (**max one**) • around lines 05 and 06 • line 07 • (immediately) after the input of the number **Three** marks pseudocode **One** mark for each point (**max three**) • Use of REPEAT … UNTIL // any working loop structure • check for >0 // >=0 • check for <10 // >9 • check for whole number • check for -1 • check for length of digit <> 1 Example REPEAT OUTPUT "Enter a digit " INPUT Number[Counter] UNTIL Number[Counter] = Round(Number[Counter],0) AND ((Number[Counter] = -1) OR (Number[Counter] > 0 AND Number[Counter] < 10))	4

Q110)

Question	Answer	Marks

(a)	**One** mark per mark point, max **four**
	• Line 01 / `Counter ← 100` should be `Counter ← 0`
	• Line 03 / `While Counter > 100 DO` should be `While Counter < 100 DO`
	• Line 07 / `Total ← Total + Counter` should be `Total ← Total + Number`
	• Line 09 / `ENDCASE` should be `ENDIF`
	Correct algorithm
	`01 Counter ← 0`
	`02 Total ← 0`
	`03 WHILE Counter < 100 DO`
	`04     INPUT Number`
	`05     IF Number > 0`
	`06       THEN`
	`07         Total ← Total + Number`
	`08         Counter ← Counter + 1`
	`09     ENDIF`
	`10 ENDWHILE`
	`11 OUTPUT "The total value of your numbers is ", Total`
	`12 OUTPUT "The average value of your numbers is ", Total / 100`
		4

Question	Answer	Marks
(b)	**One** mark per mark point, max **five** MP1 replace line 03 MP2 with `FOR` MP3 … with limits 0 to 99 / 1 to 100 MP4 replace line 05 to check if `Number` is not positive MP5 … (if `Number` is not positive) insert a validation and re-input routine between lines 06 and 07 … MP6 … that will repeat until a positive value is entered MP7 remove the counter update / line 08 MP8 replace line 10 / `ENDWHILE` with `NEXT`	**5**

Q111)

Question	Answer	Marks
(a)	Displaying/sort 10 names in alphabetical order 9	**1**

Question	Answer	Marks
(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	**4**

(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`
		3

(d)	One mark for each point (max two) • use of comments • use of procedures/functions • use of white space	**2**

Q112)

Question	Answer	Marks

(a)	**One** mark per mark point, max **four**
	MP1 Line 01 / `DECLARE City ARRAY[1:50, 1:2] OF BOOLEAN` should be `DECLARE City : ARRAY[1:50, 1:2] OF STRING` Line 05 / `IF` should be `REPEAT`
	MP2 Line 07 / `INPUT City[Count, 2]`should be `INPUT City[Count, 1]`
	MP3 Line 11 / `UNTIL Count = 50` // Line 04 / `Count ← 1` **AND** Line 10 / `Count ← Count + 1` should be `UNTIL Count = 51` / `UNTIL Count > 50` // Line 04 / `Count ← 0` **AND** move Line 10 to beginning of loop / Line 06
	MP4 Line 12 / `FOR Out ← 1 TO 1` should be `FOR Out ← 1 TO 50`
	Correct algorithm:
	`01 DECLARE City : ARRAY[1:50, 1:2] OF STRING`
	`02 DECLARE Count : INTEGER`
	`03 DECLARE Out : INTEGER`
	`04 Count ← 1`
	`05 REPEAT`
	`06     OUTPUT "Enter the name of the city"`
	`07     INPUT City[Count, 1]`
	`08     OUTPUT "Enter the name of the country"`
	`09     INPUT City[Count, 2]`
	`10     Count ← Count + 1`
	`11` **`UNTIL Count > 50`**
	`12 FOR Out ← 1 TO 50`
	`13     OUTPUT "The city ", City[Out, 1], " is in ",`
	`               City[Out, 2]`
	`14 NEXT Out`
		4

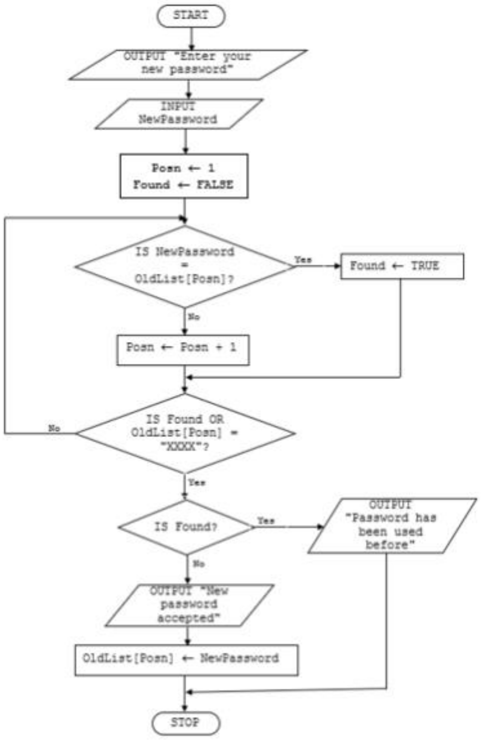
Question	Answer	Marks
(b)	**One** mark per mark point, max **five** MP1 add an input (and prompt to ask) for the country to be searched MP2 ...between lines 11 and 12 MP3 ...using a new variable for the input MP4 Add an `IF` statement to check if the current `Country` array element matches the country being searched MP5 ...between lines 12 and 13 MP6 ...if it does, allow the output in line 13 // the output in line 13 should be after a `THEN` MP7 If it does not, check the next element.	**5**

Q113)

Question	Answer	Marks
	One mark per mark point, max **three** MP1 variables and constants should have meaningful identifiers MP2 …so that programmers/future programmers are able to understand their purpose MP3 they are **both** used for data storage MP4 constants store values that never change during the execution of a program // by example MP5 variables contain values that have been calculated within the program / can change during the execution of the program // by example	**3**

Q114)

Question	Answer	Marks
(a)	**One** mark for each error identified and correction given • Line `06` `Password` should be `NewPassword` • Line `11` `AND` should be `OR` • Line `16` `INPUT` should be `OUTPUT`	3

<table>
<tr><th>Question</th><th colspan="2">Answer</th><th>Marks</th></tr>
<tr><td>(b)</td><td>

START
OUTPUT "Enter your new password"
INPUT NewPassword
Posn ← 1
Found ← FALSE
IS NewPassword = OldList[Posn]?
Yes
Found ← TRUE
No
Posn ← Posn + 1
IS Found OR OldList[Posn] = "XXXX"?
No
Yes
IS Found?
Yes
OUTPUT "Password has been used before"
No
OUTPUT "New password accepted"
OldList[Posn] ← NewPassword
STOP
</td><td>
<b>Max six</b> marks from:

<b>Max four</b> from:
<b>one</b> mark for data entry with message
<b>one</b> mark for initialisation
<b>one</b> mark for checking list // decision box comparing input with array
<b>one</b> mark for updating // updating the two variables position and found
<b>one</b> mark for loop control // second decision box
<b>one</b> mark for setting new password to position in list
<b>one</b> mark for outputs // two outputs

<b>Two marks</b>:
<b>one mark</b> for correct use of flow chart symbols
<b>one mark</b> for correct use arrows and labels
</td><td>6</td></tr>
</table>

Q115)

Question	Answer	Marks

(a)	**One** mark per mark point
	• Line 01 / `DECLARE Loop : STRING` should be `DECLARE Loop : INTEGER`
	• Line 07 / `IF Loop ← 1 TO Limit` should be `FOR Loop ← 1 TO Limit`
	• Line 09 / `INPUT Loop` should be `INPUT Value`
	• Line 10 / `Total ← Total * Value` should be `Total ← Total + Value`
	Correct algorithm:
	`01 DECLARE Loop : INTEGER`
	`02 DECLARE Limit : INTEGER`
	`03 DECLARE Value : REAL`
	`04 DECLARE Total : REAL`
	`05 Total ← 0`
	`06 Limit ← ROUND(RANDOM() * 19,0) + 1`
	`07 FOR Loop ← 1 TO Limit`
	`08     OUTPUT "Enter a number"`
	`09     INPUT Value`
	`10     Total ← Total + Value`
	`11 NEXT Loop`
	`12 OUTPUT "The total of the numbers entered is ", Total`
	`13 OUTPUT "The average number entered is ", Total / Limit`
		4

b)	**One** mark per mark point
	• Correct use of `ROUND` with 2 arguments separated by comma, for example ROUND(5, 2) (in statement 13) …
	• … `(Total / Limit,1)` correct arguments
	For example:
	`OUTPUT "The average of the numbers entered is ",`
	`ROUND(Total / Limit,1)`
		2

(c)	**One** mark per mark point, **max four** • After line 09 / after the input • Insert a `WHILE` / pre-condition loop… • … to check if the value entered is between 1 and 500 inclusive • If the value is not in range, output an error message • … and insert another input statement for re-input. **Or** • Before line 08 / before the input message • start a `REPEAT` / post-condition loop • After line 09 / after the input • … close the `REPEAT` loop by checking if the value entered was between 1 and 500 inclusive • If it wasn't, (the loop will repeat) for the number to be re-input / If it was, the program continues.	4

Q116)

Question	Answer	Marks

	One mark per mark point
	MP1 Correct input statement with appropriate variable
	MP2 Elements of selection statement present – `CASE OF ENDCASE`
	MP3 At least one correct branch in the case statement
	MP4 All branches from 1 to 4 correct
	MP5 Correct use of `OTHERWISE` with correct output.
	For example:
	`INPUT Number`
	`CASE OF Number`
	`  1 : OUTPUT Number`
	`  2 : OUTPUT Number`
	`  3 : OUTPUT Number`
	`  4 : OUTPUT Number`
	`  OTHERWISE OUTPUT "ERROR"`
	`ENDCASE`
	Or
	`INPUT Number`
	`CASE OF Number`
	`  1 : OUTPUT 1`
	`  2 : OUTPUT 2`
	`  3 : OUTPUT 3`
	`  4 : OUTPUT 4`
	`  OTHERWISE OUTPUT "ERROR"`
	`ENDCASE`
		5

Q117)

Question	Answer	Marks
(a)	**One** mark per mark point, **max two** • Data stored in a file gives a permanent copy / prevents the data from being lost • … so it can be recalled / used again in a program / in another program • can be stored elsewhere / transferred to another computer	**2**

(b)	**One** mark per mark point, **max four**
	MP1 Declaration of a string variable
	MP2 Correct use of `OPENFILE` keywords for reading
	MP3 Correct use `READFILE` with string variable
	MP4 Correct use of `UCASE` and `LENGTH` functions **and** output of each
	MP5 Correct use of `CLOSEFILE`
	Example:
	`DECLARE Words : STRING`
	`OPENFILE Quotation.txt FOR READ`
	`READFILE Quotation.txt, Words`
	`OUTPUT UCASE(Words), LENGTH(Words)`
	`CLOSEFILE Quotation.txt`
		4

Q118)

Question	Answer	Marks
(a)	**One** mark per mark point MP1 length check … MP2 … to ensure the product code entered is 6 characters in length MP3 format check … MP4 … to ensure the first two characters of the product code entered are "PD" MP5 range check … MP6 … to ensure that the value of the last four figures of the product code entered is between 1000 and 9999	**6**

(b)(i)	**One** mark for correct use of `LENGTH` operation, **one** mark for appropriate test
	Example:
	`REPEAT`
	`   INPUT Product`
	`UNTIL LENGTH(Product) = 6`
		2

(b)(ii)	**One** mark for correct use of `SUBSTRING` operation, **one** mark for appropriate test
	Example:
	`REPEAT`
	`   INPUT Product`
	`UNTIL SUBSTRING(Product, 1, 2) = "PD"`
		2

Q119)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>

One mark for:
MP1 adding current value to total

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

MP2 input more than one number
MP3 setting total to zero before loop
MP4 correct use of loop including terminal condition
MP5 output total **after loop**

Example:

```
Total ← 0
INPUT Value
WHILE Value <> 9999.9
    Total ← Total + Value
    INPUT Value
ENDWHILE
OUTPUT Total

Value ← 0
Total ← 0
REPEAT
    Total ← Total + Value
    INPUT Value
UNTIL Value = 9999.9
OUTPUT Total
```

</td><td>4</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(b)</td><td>

One mark for each point

MP1 adding one to counter
MP2 **correct** use of selection, if current value > 100 THEN … ENDIF

One mark for each point, **max two**

MP3 input more than one number
MP4 setting counter to zero before loop
MP5 correct use of loop including terminal condition
MP6 output value of counter **after loop**

Example:

```
Counter ← 0
INPUT Value
WHILE Value <> 9999.9
    IF Value > 100
      THEN
        Counter ← Counter + 1
    ENDIF
    INPUT Value
ENDWHILE
OUTPUT Counter
```

</td><td>4</td></tr>
</table>

Q120)

Question	Answer	Marks
(a)	01//02//06//10 04(07) and/or 08 03(12)	3

Question	Answer	Marks

(b)	**One** mark for each error identified and corrected
	Line 04 < should be >
	Line 08 Count should be Counter
	Line 11 ENDWHILE should be ENDIF
	`01 Max ← List[1]`
	`02 Min ← List[1]`
	`03 FOR Counter ← 2 TO 1000`
	`04     IF List[Counter] > Max`
	`05       THEN`
	`06         Max ← List[Counter]`
	`07     ENDIF`
	`08     IF List[Counter] < Min`
	`09       THEN`
	`10         Min ← List[Counter]`
	`11     ENDIF`
	`12 NEXT Counter`
	`13 OUTPUT "Maximum value is ", Max`
	`14 OUTPUT "Minimum value is ", Min`
		3

Q121)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>

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`

Correct algorithm:

```
01 DECLARE People : ARRAY[1:50, 1:3] OF STRING
02 DECLARE Count : INTEGER
03 DECLARE Response : CHAR
04 DECLARE Continue : BOOLEAN
05 FOR I ← 1 TO 50
06     FOR J ← 1 TO 3
07         People[I, J] ← ""
08     NEXT J
09 NEXT I
10 Count ← 1
11 Continue ← TRUE
12 REPEAT
13     OUTPUT "Enter the last name"
14     INPUT People[Count, 1]
15     OUTPUT "Enter the first name"
16     INPUT People[Count, 2]
17     OUTPUT "Enter the city"
18     INPUT People[Count, 3]
19     OUTPUT "Do you want to enter another name (Y or N)?"
20     INPUT Response
21     IF Response = 'N'
22       THEN
23         Continue ← FALSE
24       ELSE
25       Count ← Count + 1
26     ENDIF
27 UNTIL NOT Continue   // UNTIL Response = 'N'
```

</td><td>4</td></tr>
<tr><td>(b)</td><td>

- 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
```

</td><td>4</td></tr>
<tr><td>(c)</td><td>

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

</td><td>4</td></tr>
</table>

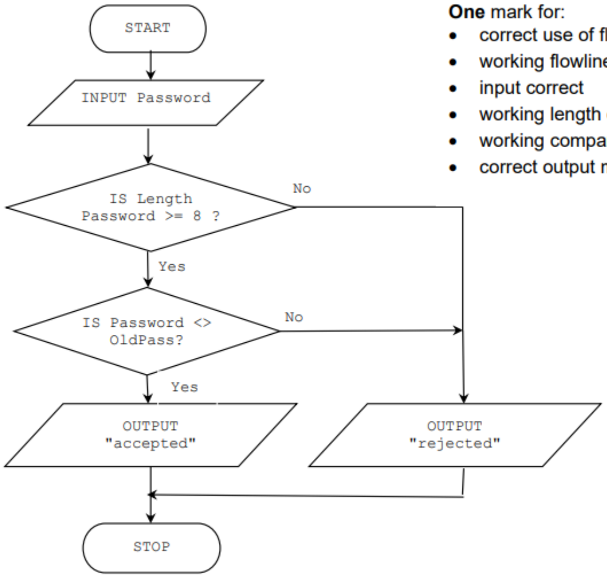
Q122)

Question	Answer	Marks
(a)	**One** mark for each point • `06 C ← 1` • `08 W ← W + A[C]` • `11 X ← W / (C – 1) // ROUND(W / (C – 1),0)`	**3**

(b)	**One** mark for outputting `X`
	One mark for outputting `C – 1`
	One mark for suitable messages
	
	Example:
	
	`12 OUTPUT "Number of values stored in the array is ", C - 1`
	`13 OUTPUT "Average of non-zero elements in the array is ", X`
		3

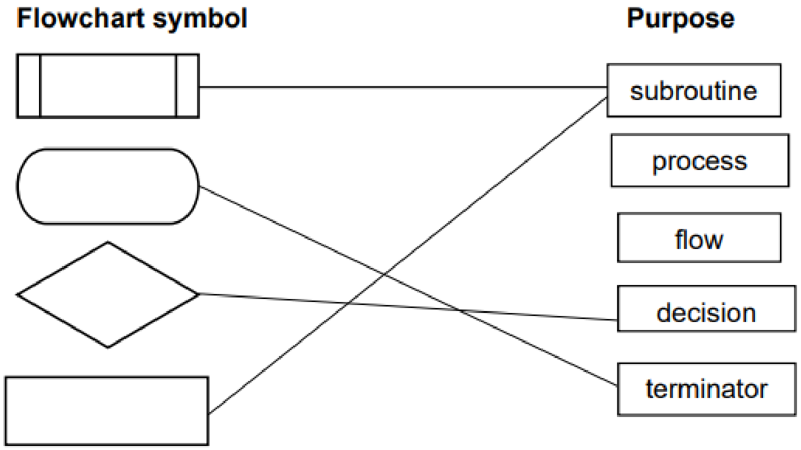
(c)	**One** mark for meaningful identifier for the array
	`A` **`Values`**
	
	Two marks for 3 meaningful identifiers for the variables
	or
	One mark for 1 to 2 meaningful identifiers for the variables
	
	`C` **`Index`**
	`X` **`Average`**
	`W` **`Total`**
		3

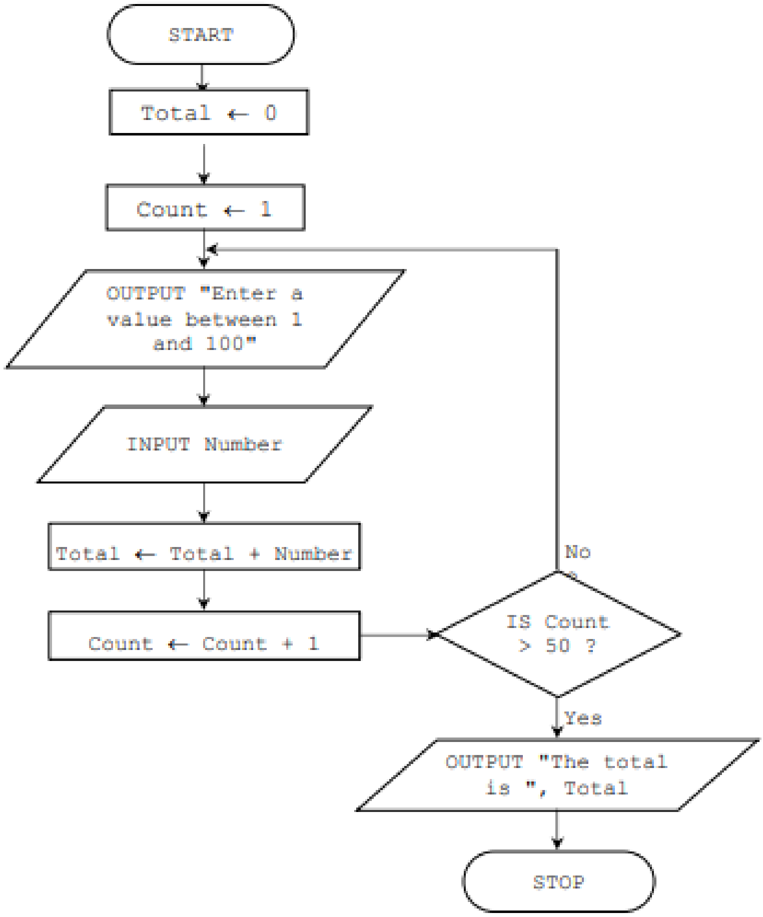
Q123)

Question	Answer	Marks
(a)	 START INPUT Password IS Length Password >= 8 ? No Yes IS Password <> OldPass? No Yes OUTPUT "accepted" OUTPUT "rejected" STOP **One** mark for: • correct use of flowchart symbols • working flowlines • input correct • working length check • working comparison • correct output messages	6
(b)	**One** mark for each point • `OPENFILE MyPassword.txt FOR WRITE` • `WRITEFILE MyPassword.txt, Password` • `CLOSEFILE MyPassword.txt`	3

Question	Answer	Marks
(c)	**One** mark for each point • needs to be retrieved on demand // saved for a later date • storage must be non-volatile	2

Q124)

Question	Answer	Marks
(a)	**One** mark for each correct line  Flowchart symbol Purpose subroutine process flow decision terminator	**4**

Question	Answer	Marks
(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  START Total ← 0 Count ← 1 OUTPUT "Enter a value between 1 and 100" INPUT Number Total ← Total + Number Count ← Count + 1 IS Count > 50 ? No Yes OUTPUT "The total is ", Total STOP	**6**

Q125)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>

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`

Correct algorithm:

<pre>
01 DECLARE ItemList : ARRAY[1:100] OF STRING
<b>02 DECLARE Counter : INTEGER</b>
03 DECLARE Limit : INTEGER
04 DECLARE Pass : INTEGER
05 DECLARE Swapped : BOOLEAN
06 DECLARE Temp : STRING
07 Limit ← 100
08 Pass ← 1
<b>09 Swapped ← TRUE</b>
<b>10 WHILE Swapped = TRUE AND Pass <= Limit - 1 DO</b>
11   Swapped ← FALSE
12   FOR Counter ← 1 TO Limit - Pass
13   IF ItemList[Counter] > ItemList[Counter + 1]
14   THEN
15   Temp ← ItemList[Counter]
16   ItemList[Counter] ← ItemList[Counter + 1]
<b>17   ItemList[Counter + 1] ← Temp</b>
18   Swapped ← TRUE
<b>19   ENDIF</b>
20   Pass ← Pass + 1
21   NEXT Counter
22 ENDWHILE
</pre>

</td><td>5</td></tr>
<tr><td>(b)</td><td>

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

</td><td>3</td></tr>
</table>

Q126)

Question	Answer	Marks
	One mark for test data type, **one** mark for matching example and one mark for matching outcome (**Max nine**) • normal (1) e.g. 25 (1) accepted (1) • extreme (1) 1/100 (1) accepted (1) • abnormal (1) e.g. 125 (1) rejected (1) • boundary (1) 1 and 0 // 100 and 101 (1) first value is accepted, and second value rejected (1)	**9**

TRACETABLE

Q127)

(a) **Number 1 Trace Table**

X	T1	T2	Output
37	2	5	5
2			2

← (1 mark) →← (1 mark) →

Number 2 Trace Table

X	T1	T2	Output
191	11	15	F
11			B

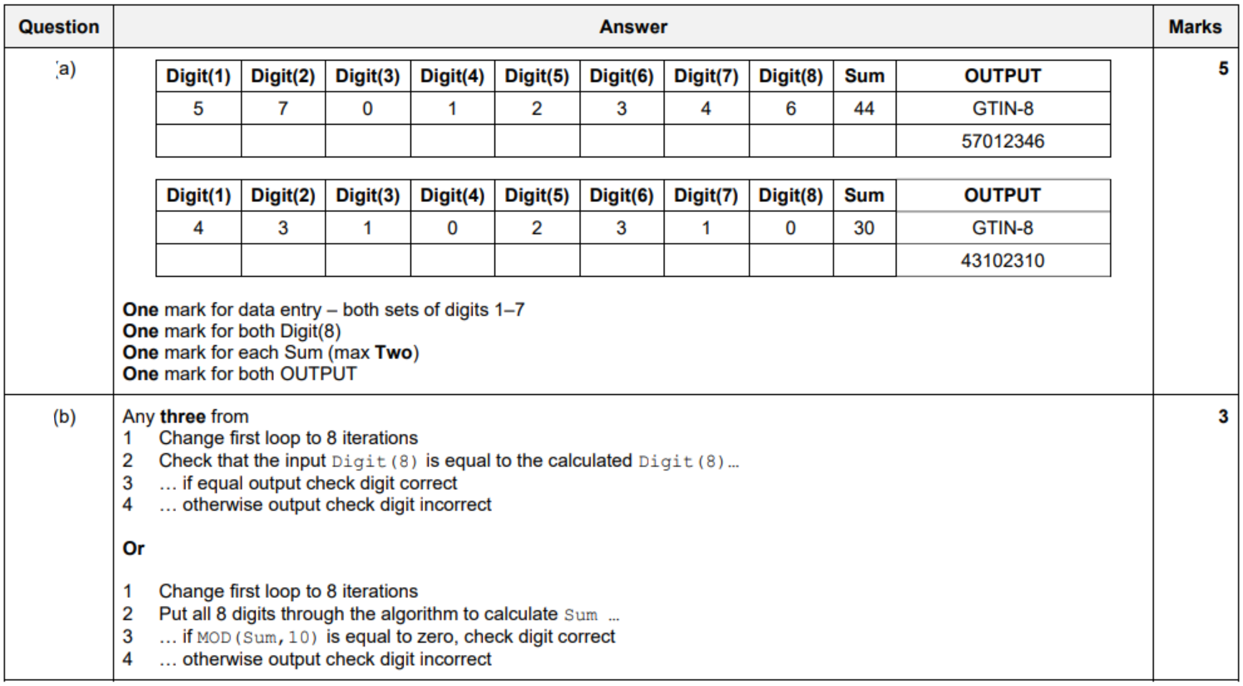
← (1 mark) →← (1 mark) →

[4]

(b) – convert a denary number to hexadecimal
– and output it in reverse order

[2]

Q128)



<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th>Digit(1)</th><th>Digit(2)</th><th>Digit(3)</th><th>Digit(4)</th><th>Digit(5)</th><th>Digit(6)</th><th>Digit(7)</th><th>Digit(8)</th><th>Sum</th><th>OUTPUT</th></tr>
<tr><td>5</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>6</td><td>44</td><td>GTIN-8</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>57012346</td></tr>
</table>
<table>
<tr><th>Digit(1)</th><th>Digit(2)</th><th>Digit(3)</th><th>Digit(4)</th><th>Digit(5)</th><th>Digit(6)</th><th>Digit(7)</th><th>Digit(8)</th><th>Sum</th><th>OUTPUT</th></tr>
<tr><td>4</td><td>3</td><td>1</td><td>0</td><td>2</td><td>3</td><td>1</td><td>0</td><td>30</td><td>GTIN-8</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>43102310</td></tr>
</table>
<b>One</b> mark for data entry – both sets of digits 1–7
<b>One</b> mark for both Digit(8)
<b>One</b> mark for each Sum (max <b>Two</b>)
<b>One</b> mark for both OUTPUT
</td><td>5</td></tr>
<tr><td>(b)</td><td>
Any <b>three</b> from
1 Change first loop to 8 iterations
2 Check that the input <code>Digit(8)</code> is equal to the calculated <code>Digit(8)</code>…
3 … if equal output check digit correct
4 … otherwise output check digit incorrect

<b>Or</b>

1 Change first loop to 8 iterations
2 Put all 8 digits through the algorithm to calculate <code>Sum</code> …
3 … if <code>MOD(Sum,10)</code> is equal to zero, check digit correct
4 … otherwise output check digit incorrect
</td><td>3</td></tr>
</table>

Q129)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th>Number1</th><th>Number2</th><th>Sign</th><th>Answer</th><th>OUTPUT</th></tr>
<tr><td>5</td><td>7</td><td>+</td><td>12</td><td>12</td></tr>
<tr><td>6</td><td>2</td><td>-</td><td>4</td><td>4</td></tr>
<tr><td>4</td><td>3</td><td>*</td><td>12</td><td>12</td></tr>
<tr><td>7</td><td>8</td><td>?</td><td>0</td><td></td></tr>
<tr><td>0</td><td>0</td><td>/</td><td>(0)</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>
← 1 mark → ← 1 mark → ← 1 mark →
</td><td>3</td></tr>
<tr><td>(b)</td><td>
CASE Sign OF … ENDCASE (1)
List +, -, *, / with correct assignments (1)
OTHERWISE Answer ← 0 (1)
Example
CASE Sign OF
    '+' : Answer ← Number1 + Number2
    '-' : Answer ← Number1 - Number2
    '*' : Answer ← Number1 * Number2
    '/' : Answer ← Number1 / Number2
    OTHERWISE Answer ← 0
ENDCASE
</td><td>3</td></tr>
</table>

Q130)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th>Fib</th><th>Prev2</th><th>Prev1</th><th>Number</th><th>OUTPUT</th></tr>
<tr><td>1</td><td>0</td><td>1</td><td>7</td><td></td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>6</td><td></td></tr>
<tr><td>2</td><td>1</td><td>2</td><td>5</td><td></td></tr>
<tr><td>3</td><td>2</td><td>3</td><td>4</td><td></td></tr>
<tr><td>5</td><td>3</td><td>5</td><td>3</td><td></td></tr>
<tr><td>8</td><td>5</td><td>8</td><td>2</td><td>8</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>< 1 Mark ></td><td colspan="2"><-------- 1 Mark --------></td><td><1 Mark></td><td><1 Mark></td></tr>
</table>
</td><td>4</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(b)</td><td>
<table>
<tr><th>Fib</th><th>Prev2</th><th>Prev1</th><th>Number</th><th>OUTPUT</th></tr>
<tr><td>1</td><td>0</td><td>1</td><td>2</td><td>1</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="3"><-------------- 1 Mark --------------></td><td colspan="2"><-------- 1 Mark --------></td></tr>
</table>
</td><td>2</td></tr>
</table>

Q131)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th rowspan="2">Index</th><th rowspan="2">Count</th><th rowspan="2">Value</th><th colspan="8">PassMarks</th><th rowspan="2">OUTPUT</th></tr>
<tr><th>[0]</th><th>[1]</th><th>[2]</th><th>[3]</th><th>[4]</th><th>[5]</th><th>[6]</th><th>[7]</th></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>0</td><td>58</td><td>58</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>1</td><td>40</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>2</td><td>67</td><td></td><td>67</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>3</td><td>85</td><td></td><td></td><td>85</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td>4</td><td>12</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td>5</td><td>13</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td>6</td><td>75</td><td></td><td></td><td></td><td>75</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>7</td><td>82</td><td></td><td></td><td></td><td></td><td>82</td><td></td><td></td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Number passed 5</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1 mark</td><td>1 mark</td><td>1 mark</td><td>1 mark</td><td colspan="7">1 mark</td><td>1 mark</td></tr>
</table>
</td><td>6</td></tr>
<tr><td>(b)</td><td><b>One</b> from:
• Stores numbers greater than 50 in an array
• Outputs number of times pass mark has been met
• Find the number of pass marks</td><td>1</td></tr>
</table>

Q132)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th>First</th><th>Last</th><th>UserIn</th><th>Middle</th><th>Found</th><th>OUTPUT</th></tr>
<tr><td>0</td><td>16</td><td></td><td></td><td>FALSE</td><td></td></tr>
<tr><td>0</td><td>16</td><td>10</td><td>8</td><td>FALSE</td><td></td></tr>
<tr><td>0</td><td>7</td><td>10</td><td>3</td><td>FALSE</td><td></td></tr>
<tr><td>4</td><td>7</td><td>10</td><td>5</td><td>TRUE</td><td>TRUE</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
<b>One</b> mark per correct column</td><td>6</td></tr>
</table>

Question	Answer	Marks
(b)	**Two** from: • Search for the value input … • … using an array… • … of sorted data	2

Q133)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td>
<table>
<tr><th>Flag</th><th>Number</th><th>Divisor</th><th>Value</th><th>OUTPUT</th></tr>
<tr><td>False</td><td>5</td><td>2</td><td>2</td><td></td></tr>
<tr><td></td><td></td><td>3</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>5 is prime</td></tr>
<tr><td>False</td><td>6</td><td>2</td><td>3</td><td></td></tr>
<tr><td>True</td><td></td><td>3</td><td>2</td><td></td></tr>
<tr><td>True</td><td></td><td>4</td><td></td><td></td></tr>
<tr><td>False</td><td>8</td><td>2</td><td>4</td><td></td></tr>
<tr><td>True</td><td></td><td>3</td><td>2</td><td></td></tr>
<tr><td>True</td><td></td><td>4</td><td>2</td><td></td></tr>
<tr><td></td><td></td><td>5</td><td></td><td></td></tr>
<tr><td>False</td><td>0</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>
<b>One</b> mark for each correct column
</td><td>5</td></tr>
</table>

Q134)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th>Flag</th><th>Count</th><th>Num [0]</th><th>Num [1]</th><th>Num [2]</th><th>Num [3]</th><th>Num [4]</th><th>Store</th></tr>
<tr><td></td><td></td><td>45</td><td>56</td><td>30</td><td>12</td><td>15</td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td>45</td></tr>
<tr><td></td><td></td><td>56</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td>45</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td><td>12</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>15</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td>12</td><td></td></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
<b>One</b> mark – <code>Flag</code> column
<b>One</b> mark – <code>Count</code> column
<b>One</b> mark – <code>Num[0]</code> and <code>Num[1]</code> columns
<b>One</b> mark – <code>Num[2]</code>, <code>Num[3]</code> and <code>Num[4]</code> columns
<b>One</b> mark – <code>Store</code> column
</td><td><b>5</b></td></tr>
<tr><td>(b)</td><td>Any <b>two</b> from:
• The algorithm sorts/orders numbers
• … into descending order / from largest to smallest</td><td><b>2</b></td></tr>
</table>

Q135)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td><b>One</b> mark for each correct column
<table>
<tr><th>Password</th><th>PasswordRepeat</th><th>OUTPUT</th></tr>
<tr><td></td><td></td><td>(Please enter password)</td></tr>
<tr><td>Secret</td><td></td><td>Reject</td></tr>
<tr><td></td><td></td><td>(Please enter password)</td></tr>
<tr><td>Secret</td><td></td><td>Reject</td></tr>
<tr><td></td><td></td><td>(Please enter password)</td></tr>
<tr><td>VerySecret</td><td>VerySecret</td><td>Accept</td></tr>
<tr><td></td><td></td><td>(Please enter password)</td></tr>
<tr><td>Pa55word</td><td>Pa55word</td><td>Accept</td></tr>
<tr><td></td><td></td><td>(Please enter password)</td></tr>
<tr><td>999</td><td></td><td>Reject</td></tr>
</table>
</td><td>3</td></tr>
</table>

Question	Answer	Marks
(b)	Any **four** from: • Position: before `INPUT PasswordRepeat` // at start • …use a (variable) counter (for number of tries) **or** flag • … initialise variable counter **or** flag • Position after `IF Length(Password) >= 8 THEN` or after `INPUT PasswordRepeat` • … insert `REPEAT/WHILE/`(conditional) loop • Position after `OUTPUT "Reject"` • … add one to counter (for number of tries) • … output a message `"Try again"` • … add `INPUT PasswordRepeat` • Position after `OUTPUT "Accept"` • … reset flag to show password matched • Position after `ENDIF` • … (insert `UNTIL/ENDWHILE)` to exit the loop after three tries **or** if the repeated password matches the original	4

Q136)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>a)</td><td><b>One</b> mark per mark point, max <b>five</b>
MP1 correct <code>Counter</code> and <code>Limit</code> columns
MP2 correct <code>Value</code> column
MP3 correct <code>First</code> column
MP4 correct <code>Last</code> column
MP5 correct OUTPUT
<table>
<tr><th>Counter</th><th>Value</th><th>First</th><th>Last</th><th>Limit</th><th>OUTPUT</th></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td>8</td><td></td></tr>
<tr><td>1</td><td>66</td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>606</td><td>6</td><td>6</td><td></td><td>606</td></tr>
<tr><td>3</td><td>6226</td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>8448</td><td></td><td></td><td></td><td></td></tr>
<tr><td>5</td><td>642</td><td>6</td><td>2</td><td></td><td></td></tr>
<tr><td>6</td><td>747</td><td>7</td><td>7</td><td></td><td>747</td></tr>
<tr><td>7</td><td>77</td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td>121</td><td>1</td><td>1</td><td></td><td>121</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td><td><b>5</b></td></tr>
</table>

Question	Answer	Marks
(b)	**One** mark per mark point, max **two** • checks for / outputs 3-digit numbers • … where the first and last digit are the same	**2**

Q137)

Question	Answer	Marks
(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	7

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

Question	Answer	Marks
(b)	**One** mark per mark point, max **two** • to find / output prime numbers • … store prime numbers in an array	2
(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	3

Q138)

Total	Reject	Weight	Output
0	0		
1.8		1.8	
	1	26.0	
8.8		7.0	
20.1		11.3	
30.1		10.0	
32.6		2.5	
	2	25.2	
37.6		5.0	
57.4		19.8	
	3	29.3	
		–1	57.4, 3
(2 marks) (–1 for each error) (then follow though)	(1 mark)	1 mark)	(1 mark) (allow follow through) (from Total and Reject)

[5]

Q139)

(a)

Trace table set 1

A	B	C	D	E	F	Total	Check	Output
5	2	4	3	1	5	38	5	Accept

←----------------------------------(1 mark)------------------------→←----------(1 mark)------→

Trace table set 2

A	B	C	D	E	F	Total	Check	Output
3	2	1	0	7	3	45	1	Reject

←----------------------------------(1 mark)------------------------→←----------(1 mark)------→

[4]

(b) – (modulo 11) check digit calculation [1]

(c) **1** mark for identifying the problem, **2** marks for the solution

Problem – doesn't deal correctly with remainder 10/a check digit of X

Solution – check Z for X as a final digit
– have a special case where check = 10
– accept where `Check = 10` and F = X [3]

Q140)

Total	Reject	Weight	Output
0	0		
1.8		1.8	
	1	26.0	
8.8		7.0	
20.1		11.3	
30.1		10.0	
32.6		2.5	
	2	25.2	
37.6		5.0	
57.4		19.8	
	3	29.3	
		–1	57.4, 3
(2 marks) (–1 for each error) (then follow though)	(1 mark)	1 mark)	(1 mark) (allow follow through) (from Total and Reject)

[5]

Q141)

(a)

Number 1 Trace table

X	Posn	New	T1	T2	Output
5	1	0			
	10	1	2	1	
2	100	1	1	0	
		101			
					101

← (1 mark) →← (1 mark) →← (1 mark) →

Number 2 Trace table

X	Posn	New	T1	T2	Output
12	1	0			
	10	0	6	0	
6	100	0	3	0	
3	1000	100	1	1	
		1100			
					1100

← (1 mark) →← (1 mark) →← (1 mark) →

[6]

(b) Converts a (denary) number to binary [1]

Q142)

Riders	Reject	Height	Output
0	0		
1		1.4	
2		1.3	
	1	1.1	
3		1.3	
	2	1.0	
4		1.5	
5		1.2	
6		1.3	
7		1.4	
8		1.3	
			Ready to go 2
(1 mark)	(1 mark)	(1 mark)	(1 mark)

[4]

Q143)

Area	Tins	Height	Width	Doors	Windows
0	0	3	5	1	0
13.5		3	7	0	0
34.5		3	5	0	3
46.5		3	7	1	1
65		–1	0	0	0
	7				
(2 marks)	←(1 mark)→	←		(1 mark)	→

1 mark 0, 13.5
1 mark for rest

[4]

Q144)

Riders	Reject	Height	Output
0	0		
1		1.4	
2		1.3	
	1	1.1	
3		1.3	
	2	1.0	
4		1.5	
5		1.2	
6		1.3	
7		1.4	
8		1.3	
			Ready to go 2
(1 mark)	(1 mark)	(1 mark)	(1 mark)

[4]

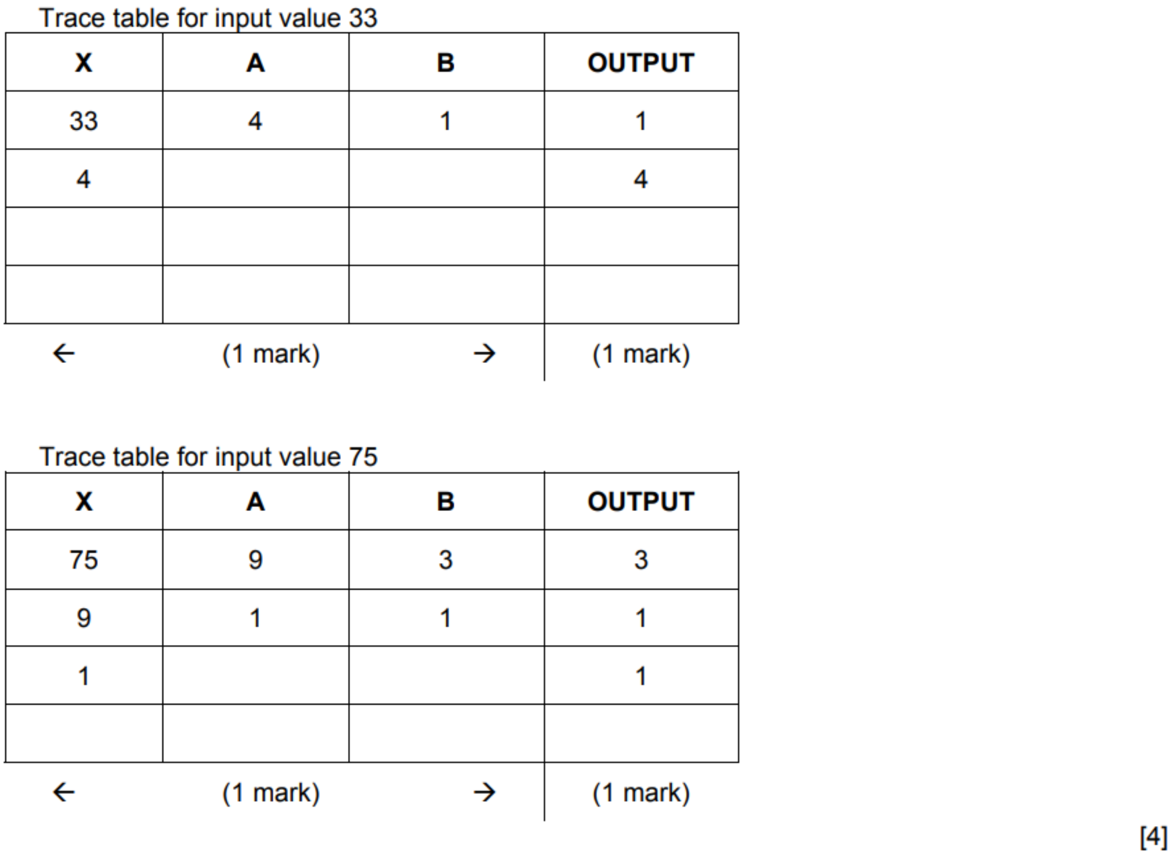
Q145)

Price	Change	Dollars	TenCents	OUTPUT
6.29				
	3.71			
		3		
	0.71			3 dollars
	0.21			One 50 cent coin
			2	2 ten cent coins
				One 5 cent coin

(1 mark per correct column)

[5]

Q146)



Trace table for input value 33

X	A	B	OUTPUT
33	4	1	1
4			4
←	(1 mark)	→	(1 mark)

Trace table for input value 75

X	A	B	OUTPUT
75	9	3	3
9	1	1	1
1			1
←	(1 mark)	→	(1 mark)

[4]

Q147)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td>
<table>
<tr><th>HighF</th><th>HighC</th><th>TempF</th><th>OUTPUT</th></tr>
<tr><td>–100</td><td>–100</td><td></td><td></td></tr>
<tr><td></td><td></td><td>68</td><td></td></tr>
<tr><td>68</td><td>18</td><td>46</td><td></td></tr>
<tr><td>68</td><td>18</td><td>50</td><td></td></tr>
<tr><td>68</td><td>18</td><td>86</td><td></td></tr>
<tr><td>86</td><td>27</td><td>65</td><td></td></tr>
<tr><td>86</td><td>27</td><td>50</td><td></td></tr>
<tr><td>86</td><td>27</td><td>40</td><td></td></tr>
<tr><td>86</td><td>27</td><td>30</td><td></td></tr>
<tr><td>86</td><td>27</td><td>–1</td><td>The highest temperature is, 86 Fahrenheit, 27 Celsius.</td></tr>
<tr><td>(1 Mark)</td><td>(1 Mark)</td><td>(1 Mark)</td><td>(2 Marks – see below)</td></tr>
</table>
The literal correct output is “The highest temperature is, 86 Fahrenheit, 27 Celsius.”
1 mark for values 86 and 27, 1 mark for correct output words, spacing and punctuation.
</td><td>5</td></tr>
</table>

Q148)

Question	Answer				Marks
	Weight	**Reject**	**Total Weight**	**OUTPUT**	**5**
		0	0		
	13		13		
	17		30		
	26	1			
	25		55		
	5		60		
	10		70		
	15		85		
	35	2			
	20		105		
			85	Weight of items 85 Number of items rejected 2	
	(1mark)	(1 mark)	(1 mark to 1st 85) (1 mark 105, 85)	(1 mark)	

Q149)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td>
<table>
<tr><th>HighF</th><th>HighC</th><th>TempF</th><th>OUTPUT</th></tr>
<tr><td>–100</td><td>–100</td><td></td><td></td></tr>
<tr><td></td><td></td><td>68</td><td></td></tr>
<tr><td>68</td><td>18</td><td>46</td><td></td></tr>
<tr><td>68</td><td>18</td><td>50</td><td></td></tr>
<tr><td>68</td><td>18</td><td>86</td><td></td></tr>
<tr><td>86</td><td>27</td><td>65</td><td></td></tr>
<tr><td>86</td><td>27</td><td>50</td><td></td></tr>
<tr><td>86</td><td>27</td><td>40</td><td></td></tr>
<tr><td>86</td><td>27</td><td>30</td><td></td></tr>
<tr><td>86</td><td>27</td><td>–1</td><td>The highest temperature is, 86 Fahrenheit, 27 Celsius.</td></tr>
<tr><td>(1 Mark)</td><td>(1 Mark)</td><td>(1 Mark)</td><td>(2 Marks – see below)</td></tr>
</table>
The literal correct output is “The highest temperature is, 86 Fahrenheit, 27 Celsius.”
1 mark for values 86 and 27, 1 mark for correct output words, spacing and punctuation.
</td><td>5</td></tr>
</table>

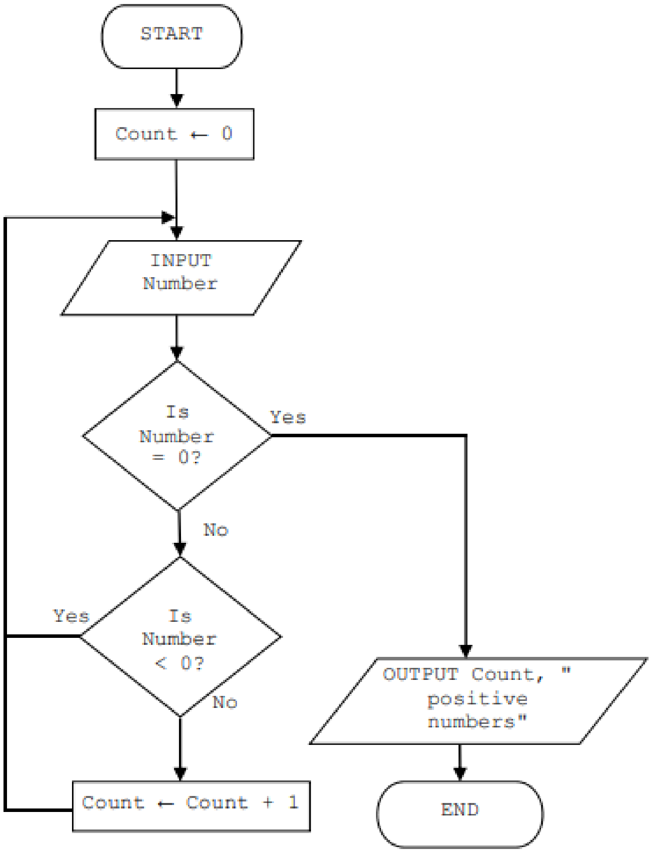
Q150)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th>Flag</th><th>Count</th><th>Name[1]</th><th>Name[2]</th><th>Name[3]</th><th>Name[4]</th><th>Temp</th></tr>
<tr><td></td><td></td><td>Jamal</td><td>Amir</td><td>Eve</td><td>Tara</td><td></td></tr>
<tr><td>0</td><td>1</td><td>Amir</td><td>Jamal</td><td>Eve</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>1</td><td>2</td><td>Amir</td><td>Jamal</td><td>Eve</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>1</td><td>3</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>1</td><td>4</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>0</td><td>1</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>0</td><td>2</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>0</td><td>3</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>0</td><td>4</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
(1 Mark) (1 Mark) (--------1 Mark--------) (--------1 Mark--------) (1 Mark)
</td><td>5</td></tr>
<tr><td>(b)</td><td>1 mark per bullet:
∞ Sorting the names
∞ Ascending order / A to Z / lowest to highest / Alphabetic order</td><td>2</td></tr>
</table>

Q151)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th>Accept</th><th>Reject</th><th>Count</th><th>Sack</th><th>OUTPUT</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td></td><td></td></tr>
<tr><td>1</td><td></td><td>1</td><td>50.4</td><td></td></tr>
<tr><td>2</td><td></td><td>2</td><td>50.3</td><td></td></tr>
<tr><td></td><td>1</td><td>3</td><td>49.1</td><td></td></tr>
<tr><td>3</td><td></td><td>4</td><td>50.3</td><td></td></tr>
<tr><td>4</td><td></td><td>5</td><td>50.0</td><td></td></tr>
<tr><td>5</td><td></td><td>6</td><td>49.5</td><td></td></tr>
<tr><td>6</td><td></td><td>7</td><td>50.2</td><td></td></tr>
<tr><td>7</td><td></td><td>8</td><td>50.3</td><td></td></tr>
<tr><td>8</td><td></td><td>9</td><td>50.5</td><td></td></tr>
<tr><td></td><td>2</td><td>10</td><td>50.6</td><td>8 2</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>← (1 mark) →</td><td>← (1 mark) →</td><td>← (1 mark) →</td><td>← (1 mark) →</td><td>← (1 mark) →</td></tr>
</table>
</td><td>5</td></tr>
<tr><td>(b)</td><td>– change to `Is Count = 50?`
– remove `IS Sack > 50.5?`</td><td>2</td></tr>
</table>

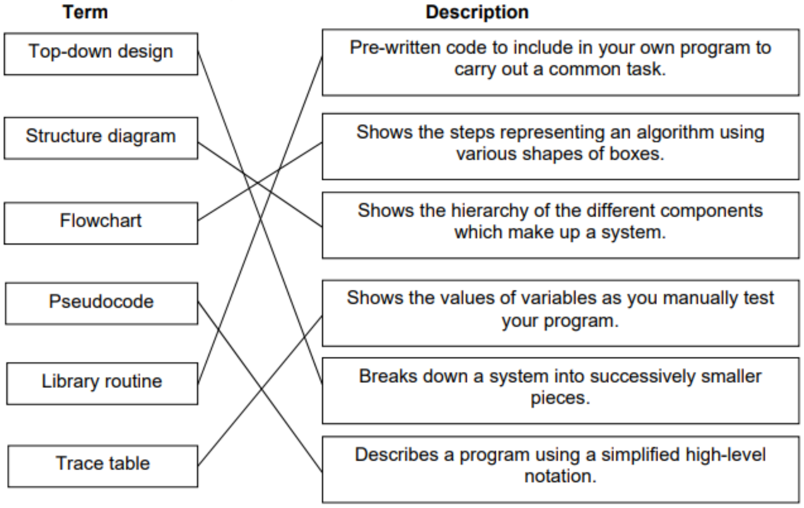
Q152)

(a)	**One** mark per correct pair of actions, process, Input/Output, Tests (apart from START and END) max 3 **One** mark complete Flowlines, **one** mark working flowlines, **one** mark correct use flowchart symbols  START Count ← 0 INPUT Number Is Number = 0? Yes No Yes Is Number < 0? No OUTPUT Count, " positive numbers" Count ← Count + 1 END	**6**
(b)	Any **two** from: • Use another counter/variable • Update this counter/variable when the number is less than zero/count all numbers **and** subtract the positive numbers • Output this counter/variable at the end // Output both counters at the end	**2**

Q153)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th>Max</th><th>Counter</th><th>Num</th><th>OUTPUT</th></tr>
<tr><td>–1000.00</td><td>0</td><td>6.30</td><td></td></tr>
<tr><td>6.30</td><td>1</td><td>18.62</td><td></td></tr>
<tr><td>18.62</td><td>2</td><td>50.01</td><td></td></tr>
<tr><td>50.01</td><td>3</td><td>3.13</td><td></td></tr>
<tr><td>50.01</td><td>4</td><td>2.05</td><td></td></tr>
<tr><td>50.01</td><td>5</td><td>50.10</td><td></td></tr>
<tr><td>50.10</td><td>6</td><td>40.35</td><td></td></tr>
<tr><td>50.10</td><td>7</td><td>30.69</td><td></td></tr>
<tr><td>50.10</td><td>8</td><td>0.85</td><td></td></tr>
<tr><td>50.10</td><td>9</td><td>17.30</td><td></td></tr>
<tr><td>50.10</td><td>10</td><td></td><td>50.10</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>
← 1 mark →← 1 mark →← 1 mark →
</td><td>3</td></tr>
<tr><td>(b)</td><td>
<b>One</b> mark for each correct change (max <b>two</b>)
Box 2 Change the initialization value of the current ‘Max’ variable to a very high number
Box 4 Change the inequality from > to <
Boxes 2, 4, 5, 8 Change the Max variable to something more suitable e.g. Min
</td><td>2</td></tr>
</table>

Q154)

	1 mark for each correct line, maximum 5 marks  **Term** Top-down design Structure diagram Flowchart Pseudocode Library routine Trace table **Description** Pre-written code to include in your own program to carry out a common task. Shows the steps representing an algorithm using various shapes of boxes. Shows the hierarchy of the different components which make up a system. Shows the values of variables as you manually test your program. Breaks down a system into successively smaller pieces. Describes a program using a simplified high-level notation.	**5**

Q155)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th>Flag</th><th>TestNum</th><th>Num</th><th>OUTPUT</th></tr>
<tr><td>True</td><td>7</td><td>6</td><td></td></tr>
<tr><td></td><td></td><td>5</td><td></td></tr>
<tr><td></td><td></td><td>4</td><td></td></tr>
<tr><td></td><td></td><td>3</td><td></td></tr>
<tr><td></td><td></td><td>2</td><td></td></tr>
<tr><td></td><td></td><td>1</td><td>7</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="3"><------------------- 1 Mark ---------------------></td><td><- 1 Mark - ></td></tr>
</table>
</td><td>2</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(b)</td><td>
<table>
<tr><th>Flag</th><th>TestNum</th><th>Num</th><th>OUTPUT</th></tr>
<tr><td>True</td><td>6</td><td>5</td><td></td></tr>
<tr><td></td><td></td><td>4</td><td></td></tr>
<tr><td></td><td></td><td>3</td><td></td></tr>
<tr><td>False</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4"><---------------- 1 Mark ------------->←---------1 Mark ------------------- ></td></tr>
</table>
</td><td>2</td></tr>
<tr><td>(c)</td><td>1 mark for correct purpose e.g.
Works out if the number entered is a prime number.</td><td>[1]</td></tr>
</table>

Q156)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th>Height</th><th>Depth</th><th>Chlorine</th><th>OK</th><th>OUTPUT</th></tr>
<tr><td>6</td><td>2.5</td><td>2</td><td>True</td><td>Pool OK to use</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>
<table>
<tr><th>Height</th><th>Depth</th><th>Chlorine</th><th>OK</th><th>OUTPUT</th></tr>
<tr><td>4</td><td>3</td><td>1.5</td><td>True</td><td>Water too deep</td></tr>
<tr><td></td><td></td><td></td><td>False</td><td></td></tr>
</table>
<table>
<tr><th>Height</th><th>Depth</th><th>Chlorine</th><th>OK</th><th>OUTPUT</th></tr>
<tr><td>6</td><td>3.5</td><td>4</td><td>True</td><td>Water too deep</td></tr>
<tr><td></td><td></td><td></td><td>False</td><td>Too much chlorine add more water</td></tr>
</table>
1 mark for first 4 columns in each trace table (max 3)
1 mark for the output in each trace table (max 3)
</td><td>6</td></tr>
<tr><td>(b)</td><td>Any <b>one</b> from:
Cannot add more water if the water is too deep
No validation e.g. allows a negative height/depth/amount of chlorine
Tells you to add chlorine when there is no water
Runs only once</td><td>1</td></tr>
</table>

Q157)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td>
<table>
<tr><th>Total</th><th>Count</th><th>Distinction</th><th>Mark</th><th>OUTPUT</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>50</td><td></td></tr>
<tr><td>50</td><td>1</td><td>0</td><td>70</td><td></td></tr>
<tr><td>120</td><td>2</td><td>0</td><td>65</td><td></td></tr>
<tr><td>185</td><td>3</td><td>0</td><td>30</td><td></td></tr>
<tr><td>215</td><td>4</td><td>0</td><td>95</td><td></td></tr>
<tr><td>310</td><td>5</td><td>1</td><td>50</td><td></td></tr>
<tr><td>360</td><td>6</td><td>1</td><td>55</td><td></td></tr>
<tr><td>415</td><td>7</td><td>1</td><td>85</td><td></td></tr>
<tr><td>500</td><td>8</td><td>2</td><td>65</td><td></td></tr>
<tr><td>565</td><td>9</td><td>2</td><td>35</td><td></td></tr>
<tr><td>600</td><td>10</td><td></td><td>–1</td><td>Number of Distinctions 2</td></tr>
<tr><td></td><td></td><td></td><td></td><td>Average Mark 60</td></tr>
</table>
1 mark for Total and Count columns both correct.
1 mark for each correct column apart from Total and Count.
If no marks awarded allow 1 mark for initialisation of Total, Count and Distinction, set to zero.
</td><td>4</td></tr>
</table>

Q158)

TreadReject	Count	Depth	OUTPUT
0	1		
	2	1.7	
	3	1.9	
1	4	1.4	
	5	1.8	
	6	2.0	
			Car is potentially roadworthy

TreadReject	Count	Depth	OUTPUT
0	1		
1	2	1.2	
	3	1.9	
2	4	1.4	
	5	1.8	
	6	2.4	
			Car is not roadworthy

1 mark for each correct pairs of columns.

4

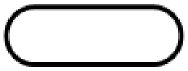
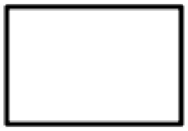
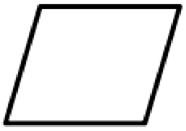
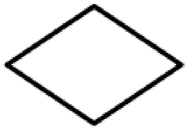
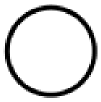
Q159)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td><b>One</b> mark for each correct row
<table>
<tr><th>Description</th><th>Structure diagram</th><th>Flowchart</th><th>Library routines</th></tr>
<tr><td>A modelling tool used to show the hierarchy of a system</td><td>✓</td><td></td><td></td></tr>
<tr><td>A collection of standard programs available for immediate use</td><td></td><td></td><td>✓</td></tr>
<tr><td>A graphical representation used to represent an algorithm</td><td></td><td>✓</td><td></td></tr>
<tr><td>A graphical representation to show how a system is broken into sub-systems</td><td>✓</td><td></td><td></td></tr>
</table></td><td><b>4</b></td></tr>
</table>

Q160)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td>

One mark for each correct symbol **and** name / description / example of use (maximum four marks)

<table>
<tr><th>Symbol</th><th>Description of use</th></tr>
<tr><td></td><td>Terminator – start / end the flowchart</td></tr>
<tr><td></td><td>Process – to show calculations, etc.</td></tr>
<tr><td></td><td>Input / Output</td></tr>
<tr><td></td><td>Decision – to show condition</td></tr>
<tr><td></td><td>Continuation – to extend the flowchart and allow it to join up</td></tr>
</table>

</td><td>4</td></tr>
</table>

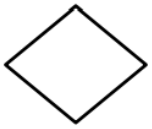
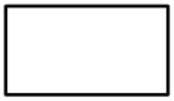
Q161)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th><code>Value</code></th><th><code>Calc1</code></th><th><code>Calc2</code></th><th><code>OUTPUT</code></th></tr>
<tr><td>50</td><td>25</td><td>16</td><td></td></tr>
<tr><td>33</td><td>16</td><td>11</td><td></td></tr>
<tr><td>18</td><td>9</td><td>6</td><td>18</td></tr>
<tr><td>15</td><td>7</td><td>5</td><td></td></tr>
<tr><td>30</td><td>15</td><td>10</td><td>30</td></tr>
<tr><td>-1</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>
<b>One</b> mark for each correct column (max <b>four</b>)</td><td>4</td></tr>
<tr><td>(b)</td><td>Any <b>two</b> correct statements e.g.
• The program outputs a value
• That is divisible by 6 // 2 and 3</td><td>2</td></tr>
</table>

Q162)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td><b>One</b> mark for correct input (<code>PointsWon</code> and <code>PointsLost</code>)
<b>One</b> mark for correct calculations (<code>Difference</code>)
<b>One</b> mark for correct output
<table>
<tr><th><code>PointsWon</code></th><th><code>PointsLost</code></th><th><code>Difference</code></th><th>OUTPUT</th></tr>
<tr><td>5000</td><td>4474</td><td>526</td><td>Keep on trying</td></tr>
<tr><td>6055</td><td>2000</td><td>4055</td><td>Well done move up</td></tr>
<tr><td>7900</td><td>9800</td><td>-1900</td><td>Sorry move down</td></tr>
<tr><td>3000</td><td>2150</td><td>850</td><td>Keep on trying</td></tr>
<tr><td>-1</td><td>6700</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>3</td></tr>
<tr><td>(b)</td><td>Any <b>three</b> from:
• Add extra decision box …
• … before checking for difference greater than or equal to 1000 // change Is difference >= 1000 to >= 1000 and <= 5000
• Check for difference greater than 5000
• Add extra Output ‘Fantastic leap up two levels’…
• … before flowline returns to input</td><td>3</td></tr>
</table>

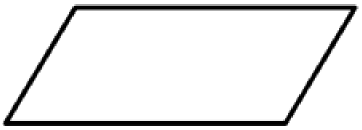
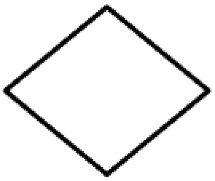
Q163)

Question	Answer	Marks
	Decision  **Process**  **One** mark for each correct symbol	2

Q164)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td><strong>One</strong> mark for correct input columns (V, W, X, Y, Z)
<strong>One</strong> mark for correct calculation column A
<strong>One</strong> mark for correct calculation column B
<strong>One</strong> mark for correct output column
<table>
<tr><th>V</th><th>W</th><th>X</th><th>Y</th><th>Z</th><th>A</th><th>B</th><th>OUTPUT</th></tr>
<tr><td>5</td><td>4</td><td>6</td><td>2</td><td>1</td><td>56</td><td>1</td><td>Valid</td></tr>
<tr><td>9</td><td>3</td><td>2</td><td>1</td><td>6</td><td>40</td><td>7</td><td>Invalid</td></tr>
<tr><td>7</td><td>6</td><td>1</td><td>5</td><td>1</td><td>61</td><td>6</td><td>Invalid</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td></td><td></td><td></td></tr>
</table></td><td><strong>4</strong></td></tr>
<tr><td>(b)</td><td>• (Use first four digits input to) calculate a check digit
• Check if the check digit input is valid</td><td><strong>2</strong></td></tr>
</table>

Q165)

Question	Answer	Marks
	Input/Output  **One** mark for each correct symbol **Decision** 	**2**

Q166)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td><b>One</b> mark for each correct column
<table>
<tr><th>Y</th><th>Z</th><th>A</th><th>OUTPUT</th></tr>
<tr><td>11</td><td>4</td><td>3</td><td>Invalid</td></tr>
<tr><td>6</td><td>2</td><td>0</td><td>Valid</td></tr>
<tr><td>3</td><td>9</td><td>0</td><td>Valid</td></tr>
<tr><td>3</td><td>2</td><td>1</td><td>Invalid</td></tr>
<tr><td>2</td><td>6</td><td>0</td><td>Valid</td></tr>
<tr><td>0</td><td>0</td><td></td><td></td></tr>
</table></td><td><b>4</b></td></tr>
<tr><td>(b)</td><td>Any <b>two</b> from:
Checking if the remainder, when the larger number is divided by the smaller number, is zero
To see if the larger number is a multiple of the smaller number
To see if the smaller number is a factor of the larger number</td><td><b>2</b></td></tr>
</table>

Q167)

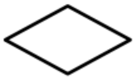
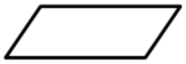
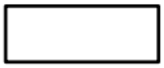
Question	Answer	Marks
(a)	**One** mark for each correct column **(Max 4)**	**4**

`Value`	`Diff1`	`Diff2`	OUTPUT
50	50	0	Accept: Extreme
75	25	25	Accept: Normal
99	1	49	Accept: Normal
28			Reject: Abnormal
82	18	32	Accept: Normal
150			Reject: Abnormal
–1			

Question	Answer	Marks
(b)	**One** mark per bullet point **(Max 2)** • To output the type of test data • … by performing a range check //… by checking if numbers are within the range 50 and 100 (inclusive) (or not).	**2**

Q168)

(a)		**3**

Pseudocode statement	Flowchart symbol
`IF Number = 20`	
`PRINT Number`	
`Number ← Number + 1`	

(b)	`IF Number = 20` selection `PRINT Number` output `Number ← Number + 1` counting	**3**

Q169)

Question	Answer	Marks
(a)	**One** mark for each correct column	**5**

`Op`	`Value1`	`Value2`	`Ans`	**OUTPUT**
1				
	87			
		14	101	
3				
	2			
		30	60	
5				
	10			
		6		Input Error
4				
	10			
		2	5	
0				

Question	Answer	Marks
(b)	To work as a calculator // to add, subtract, multiply or divide a pair of numbers	**1**
(c)	To output/store the result/the value of `Ans` // Adding prompts for data entry.	**1**

Q170)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td>One mark for each correct column
<table>
<tr><th>Counter</th><th>Distinction</th><th>Mark</th><th>Award</th><th>OUTPUT</th></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>1</td><td>88</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>74</td><td></td><td></td></tr>
<tr><td>3</td><td></td><td>60</td><td></td><td></td></tr>
<tr><td>4</td><td>2</td><td>90</td><td></td><td></td></tr>
<tr><td>5</td><td>3</td><td>84</td><td></td><td></td></tr>
<tr><td>6</td><td>4</td><td>87</td><td></td><td></td></tr>
<tr><td>7</td><td>5</td><td>95</td><td></td><td></td></tr>
<tr><td>8</td><td></td><td>72</td><td></td><td></td></tr>
<tr><td>9</td><td>6</td><td>84</td><td></td><td></td></tr>
<tr><td>10</td><td></td><td>66</td><td></td><td></td></tr>
<tr><td></td><td></td><td>-1</td><td>0.6</td><td>Highly Commended</td></tr>
</table></td><td>5</td></tr>
</table>

Q171)

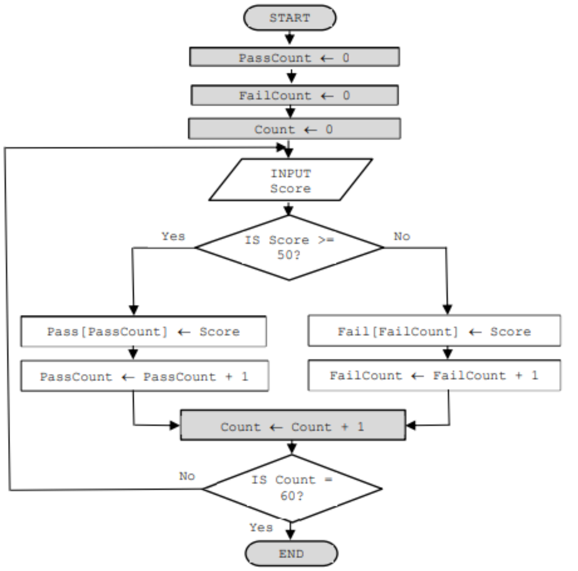
<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td><b>One</b> mark for each correct column
<table>
<tr><th>List</th><th>Value</th><th>List1</th><th>List2</th><th>OUTPUT</th></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>77</td><td></td><td>77</td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>16</td><td></td><td>93</td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>35</td><td>35</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>-7</td><td></td><td>86</td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td></tr>
</table></td><td>5</td></tr>
</table>

Q172)

Question	Answer	Marks
	One mark for each correct column	**5**

Counter	Pass	Mark	Help	OUTPUT
0	0			
1	1	88		
2		24		
3	2	60		
4		30		
5	3	44		
6		17		
7		25		
8		22		
9	4	54		
10		6		
		999	0.4	Extra Help

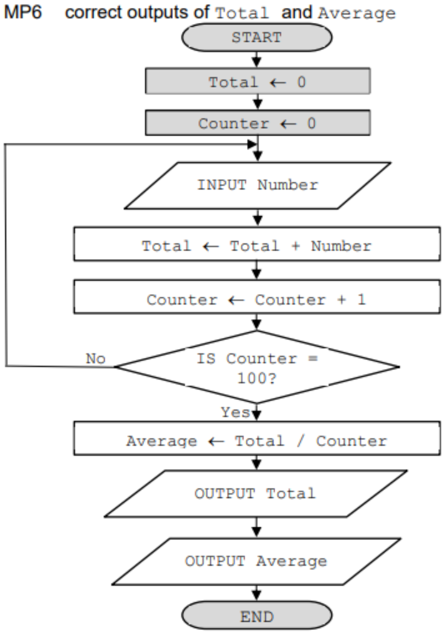
Q173)

Question	Answer	Marks
`(a)	**One** mark per mark point, max **six** MP1 input box MP2 correct check of `Score` MP3 assign `Score` to `Pass` correctly MP4 assign `Score` to `Fail` correctly MP5 increment **both** array counters MP6 correct check of number of scores  START PassCount ← 0 FailCount ← 0 Count ← 0 INPUT Score IS Score >= 50? Yes No Pass[PassCount] ← Score PassCount ← PassCount + 1 Fail[FailCount] ← Score FailCount ← FailCount + 1 Count ← Count + 1 IS Count = 60? No Yes END	6

Question	Answer	Marks

b)	**One** mark per mark point, max **four**
	
	MP1 appropriate conditional loop structure
	MP2 correct identification of invalid input
	MP3 appropriate error message
	MP4 repeated input of score until correct
	
	`WHILE Score < 0 OR Score > 100 (DO)`
	`  OUTPUT "Your entry must be between 0 and 100, inclusive, please try again "`
	`  INPUT Score`
	`ENDWHILE`
	
	Or:
	
	`REPEAT`
	`  IF Score < 0 OR Score > 100`
	`    THEN`
	`      OUTPUT "Your entry must be between 0 and 100, inclusive, please try again "`
	`      INPUT Score`
	`  ENDIF`
	`UNTIL Score >= 0 AND Score <= 100`
		4

Q174)

Question	Answer	Marks
	One mark per mark point, max **six** MP1 input box MP2 correct totalling using `Total` MP3 correct counting using `Counter` MP4 correct conditional statement for `Counter` MP5 correct calculation of `Average` MP6 correct outputs of `Total` and `Average`  START Total ← 0 Counter ← 0 INPUT Number Total ← Total + Number Counter ← Counter + 1 No IS Counter = 100? Yes Average ← Total / Counter OUTPUT Total OUTPUT Average END	6

Q175)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>`(a)</td><td><b>One</b> mark for each correct column, max <b>four</b>
<table>
<tr><th>Sold</th><th>Stock</th><th>Total</th><th>OUTPUT</th></tr>
<tr><td></td><td>50</td><td>0</td><td></td></tr>
<tr><td>24</td><td>26</td><td>24</td><td></td></tr>
<tr><td>12</td><td>14</td><td></td><td>Add new stock</td></tr>
<tr><td></td><td>64</td><td>36</td><td></td></tr>
<tr><td>6</td><td>58</td><td>42</td><td></td></tr>
<tr><td>30</td><td>28</td><td>72</td><td></td></tr>
<tr><td>12</td><td>16</td><td></td><td>Add new stock</td></tr>
<tr><td></td><td>66</td><td>84</td><td></td></tr>
<tr><td>18</td><td>48</td><td>102</td><td></td></tr>
<tr><td>-1</td><td></td><td></td><td>102</td></tr>
</table>
</td><td>4</td></tr>
<tr><td>(b)</td><td><b>One</b> mark for identification of error, max <b>one</b>
• the stock level will fall below zero / become negative
<b>One</b> mark per mark point, max <b>two</b>
• before subtracting the amount Sold from Stock
• test that the stock level / Stock is greater than the rolls to be sold / Sold
• provide a suitable error message / ask to re-input</td><td>3</td></tr>
</table>

Q176)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td>

One mark per mark point, max **five**

MP1 correct `Counter` and `Number` columns
MP2 correct `Hundreds` column
MP3 correct `Temp` and `Tens` columns
MP4 correct `Units` column
MP5 correct OUTPUT column

<table>
<tr><th>Counter</th><th>Number</th><th>Hundreds</th><th>Temp</th><th>Tens</th><th>Units</th><th>OUTPUT</th></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td>97</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>876</td><td>8</td><td>76</td><td>7</td><td>6</td><td>Hundreds: 8 Tens: 7 Units: 6</td></tr>
<tr><td>3</td><td>4320</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>606</td><td>6</td><td>6</td><td>0</td><td>6</td><td>Hundreds: 6 Tens: 0 Units: 6</td></tr>
<tr><td>5</td><td>9875</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td>42</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td>124</td><td>1</td><td>24</td><td>2</td><td>4</td><td>Hundreds: 1 Tens: 2 Units: 4</td></tr>
<tr><td>8</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

</td><td>5</td></tr>
</table>

Q177)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td><b>One</b> mark for each correct column, max <b>four</b>
<table>
<tr><th><code>Stock</code></th><th><code>Total</code></th><th><code>Sale</code></th><th>OUTPUT</th></tr>
<tr><td>10</td><td>0</td><td></td><td></td></tr>
<tr><td>9</td><td>1</td><td>Y</td><td></td></tr>
<tr><td>8</td><td>2</td><td>Y</td><td></td></tr>
<tr><td>7</td><td>3</td><td>Y</td><td></td></tr>
<tr><td>6</td><td>4</td><td>Y</td><td></td></tr>
<tr><td>5</td><td>5</td><td>Y</td><td></td></tr>
<tr><td>4</td><td>6</td><td>Y</td><td></td></tr>
<tr><td>14</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>N</td><td>Add new stock</td></tr>
<tr><td></td><td></td><td></td><td>6</td></tr>
</table>
</td><td>4</td></tr>
<tr><td>(b)</td><td><b>One</b> mark per mark point, max <b>three</b>
<ul>
<li>input a number / quantity</li>
<li>… check that the number is less than (or equal to) the number in stock</li>
<li>… after checking for N</li>
<li>update <code>Stock</code> by that number / <code>Stock ← Stock – Number</code></li>
<li>update <code>Total</code> by that number / <code>Total ← Total + Number</code></li>
</ul>
</td><td>3</td></tr>
</table>

Q178)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td><b>One</b> mark per correct column, max <b>four</b>
<table>
<tr><th>Pointer</th><th>Letter</th><th>Choice</th><th>OUTPUT</th></tr>
<tr><td>1</td><td>F</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td>Letter F is represented by Foxtrot</td></tr>
<tr><td></td><td></td><td></td><td>Another Letter? (Y or N)</td></tr>
<tr><td></td><td></td><td>Y</td><td></td></tr>
<tr><td>1</td><td>D</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td>Letter D is represented by Delta</td></tr>
<tr><td></td><td></td><td></td><td>Another Letter? (Y or N)</td></tr>
<tr><td></td><td></td><td>N</td><td></td></tr>
</table>
</td><td>4</td></tr>
<tr><td>7(b)</td><td>(Linear) search</td><td>1</td></tr>
</table>

Question	Answer	Marks
(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	2

Q179)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<b>One</b> mark for each column F, C and T
<b>Two</b> marks for columns X[1] to X[5] all entries correct or
<b>One</b> mark for columns X[1] to X[5] with one error
<table>
<tr><th>F</th><th>C</th><th>X[1]</th><th>X[2]</th><th>X[3]</th><th>X[4]</th><th>X[5]</th><th>T</th></tr>
<tr><td></td><td></td><td>10</td><td>1</td><td>5</td><td>7</td><td>11</td><td></td></tr>
<tr><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td>10</td></tr>
<tr><td>1</td><td>2</td><td>1</td><td>10</td><td></td><td></td><td></td><td>10</td></tr>
<tr><td>1</td><td>3</td><td></td><td>5</td><td>10</td><td></td><td></td><td>10</td></tr>
<tr><td>1</td><td>4</td><td></td><td></td><td>7</td><td>10</td><td></td><td></td></tr>
<tr><td></td><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>5</td></tr>
<tr><td>(b)</td><td>
<b>One</b> mark for each point
<ul>
<li>(bubble) sort data in array</li>
<li>in ascending order</li>
</ul>
</td><td>2</td></tr>
</table>

Q180)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>

One mark per mark point, max **six**

- correct `Total` column
- correct `Value` column
- correct `Five1` column
- correct `Five2` column
- correct `Ten1` and `Ten2` columns
- correct OUTPUT column

<table>
<tr><th>Total</th><th>Value</th><th>Five1</th><th>Five2</th><th>Ten1</th><th>Ten2</th><th>OUTPUT</th></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>5</td><td>1</td><td>1</td><td>0</td><td>0.5</td><td>Rejected</td></tr>
<tr><td></td><td>50</td><td>10</td><td>10</td><td>5</td><td>5</td><td></td></tr>
<tr><td>50</td><td>52</td><td>10</td><td>10.4</td><td></td><td></td><td>Rejected</td></tr>
<tr><td></td><td>555</td><td>111</td><td>111</td><td>55</td><td>55.5</td><td>Rejected</td></tr>
<tr><td></td><td>57</td><td>11</td><td>11.4</td><td></td><td></td><td>Rejected</td></tr>
<tr><td></td><td>500</td><td>100</td><td>100</td><td>50</td><td>50</td><td></td></tr>
<tr><td>550</td><td>–1</td><td></td><td></td><td></td><td></td><td>550</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

</td><td>6</td></tr>
<tr><td>(b)</td><td>

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

- to find if an input is divisible by (both 5 and) 10
- … add them together **and** output the total

</td><td>2</td></tr>
</table>

Q181)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td><b>One</b> mark per correct column, max <b>five</b>
<table>
<tr><th>Value</th><th>Average</th><th>Total</th><th>Count</th><th>OUTPUT</th></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td></td></tr>
<tr><td>25</td><td></td><td>25</td><td>1</td><td></td></tr>
<tr><td>35</td><td></td><td>60</td><td>2</td><td></td></tr>
<tr><td>3</td><td></td><td>63</td><td>3</td><td></td></tr>
<tr><td>0</td><td>21</td><td></td><td></td><td>Total is 63</td></tr>
<tr><td></td><td></td><td></td><td></td><td>Average is 21</td></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td></td></tr>
<tr><td>57</td><td></td><td>57</td><td>1</td><td></td></tr>
<tr><td>20</td><td></td><td>77</td><td>2</td><td></td></tr>
<tr><td>25</td><td></td><td>102</td><td>3</td><td></td></tr>
<tr><td>18</td><td></td><td>120</td><td>4</td><td></td></tr>
<tr><td>0</td><td>30</td><td></td><td></td><td>Total is 120</td></tr>
<tr><td></td><td></td><td></td><td></td><td>Average is 30</td></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td></td></tr>
<tr><td>−1</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>5</td></tr>
<tr><td>(b)</td><td><b>One</b> mark per mark point, max <b>two</b>
MP1 to add together / find the average of a batch of numbers
MP2 the <b>total and average</b> are <b>output</b> (when the batch is complete/when 0 is entered)
MP3 when 0 is entered a new batch is started.</td><td>2</td></tr>
</table>

Q182)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th>Accept</th><th>Reject</th><th>PartOK</th><th>Error</th><th>OUTPUT</th></tr>
<tr><td>0</td><td>0</td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>3</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td></td><td>1</td><td>N</td><td></td><td></td></tr>
<tr><td>4</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>5</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>6</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>7</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td></td><td>2</td><td>N</td><td></td><td></td></tr>
<tr><td>8</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>9</td><td></td><td>Y</td><td></td><td></td></tr>
<tr><td>10</td><td></td><td>Y</td><td>20</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>Too many rejected 20% error</td></tr>
</table>
<b>One</b> mark for each column
</td><td><b>5</b></td></tr>
</table>

Question	Answer	Marks
(b)	**One** mark for each point **max three** • after the Input box // before the first decision box • insert a process box • to convert the input to upper case **OR** • change the first decision / add another decision box • to accept `'y'` as well • by adding `OR PartOK = 'y'`	**3**

Q183)

Question	Answer	Marks
(a)	**One** mark per mark point **MP1** Correct **Limit** column **MP2** Correct **Count** column **MP3** Array columns for input stage **MP4** Array columns for sort stage **MP5** Correct **Flag** column **MP6** Correct **Swap** column **MP7** Correct **Result** and **OUTPUT** columns	7

		Numbers										
Limit	**Count**	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**	**[6]**	**[7]**	**Flag**	**Swap**	**Result**	**OUTPUT**
7	1	47										
	2		50									
	3			52								
	4				60							
	5					80						
	6						63					
	7							70				
	8								TRUE			
	1								FALSE			
	2											
	3											
	4											
	5									80		
						63	80		TRUE			
	6									80		
							70	80	TRUE			
	7											
	1								FALSE			
	2											
	3											
	4											
	5											
	6											
	7											
											4	60

Question	Answer	Marks
(b)	**One** mark per mark point • A set of numbers is input / stored in an array • The numbers are sorted / Bubble sort into (ascending) order • The middle / median value is found / output	3

Q184)

Question	Answer	Marks
(a)	**One** mark for each of columns `A`, `B` and `T` **Two** marks for columns `List[1]` to `List[5]` all entries correct or **One** mark for columns `List[1]` to `List[5]` with one error	**5**

A	B	List[1]	List[2]	List[3]	List[4]	List[5]	T
		15	17	20	5	9	
FALSE	1	17	15				15
TRUE	2		20	15			15
TRUE	3						
TRUE	4				9	5	5
	5						
FALSE	1	20	17				17
TRUE	2						
	3						
	4						
	5						
FALSE	1						
	2						
	3						
	4						
	5						

Question	Answer	Marks
(b)	**One** mark for each point MP1 (bubble) sort data in array MP2 in descending order	**2**

Q185)

Question	Answer	Marks
(a)	MP1 Correct `L` column MP2 Correct `S` column MP3 Correct `T` column MP4 Correct `A` column MP5 Correct `Limit`, `Count` and `Value` columns MP6 Correct OUTPUT columns	6

L	S	T	A	Limit	Count	Value	OUTPUT
0	10000	0	0	10	1	30	
30		30			2	18	
	18	48			3	8	
	8	56			4	25	
		81			5	12	
		93			6	17	
		110			7	2	
	2	112			8	50	
50		162			9	15	
		177			10	5	
		182	18.2		11		L = 50 S = 2 T = 182 A = 18.2

Question	Answer	Marks
(b)	**One** mark per mark point, **max two** • Any **two** from finds / outputs the largest, smallest, total and average of a set of numbers • All **four** of finds / outputs the largest, smallest, total and average of a set of numbers	2

Question	Answer	Marks
c)	• (The identifiers `L`, `S`, `T` and `A`) are **single** letters • … so do not give any indication of what values they hold. • For programs to be maintainable, identifiers should have **meaningful** names.	3
(d)	**One** mark for every two appropriate identifiers, **max two**	2

Original identifier	Improved identifier
`L`	`Largest` / `Maximum`
`S`	`Smallest` / `Minimum`
`T`	`Total` / `Sum`
`A`	`Average` / `Mean`

Q186)

Question	Answer	Marks
(a)	**One** mark for each error identified and correction: • Line 03 Temp should be Temperature • Line 04 = should be <> • Line 14 OR should be AND • Line 19 WHILE should be UNTIL	4

```
01 REPEAT
02     OUTPUT "Please enter temperature "
03     INPUT Temperature
04     IF Temperature <> 999
05       THEN
06         IF Temperature > 38.0
07           THEN
08             OUTPUT "Temperature too high"
09         ENDIF
10         IF Temperature < 35.0
11           THEN
12             OUTPUT "Temperature too low"
13         ENDIF
14         IF Temperature >= 35.0 AND Temperature <= 38.0
15           THEN
16             OUTPUT "Temperature normal"
17         ENDIF
18     ENDIF
19 UNTIL Temperature = 999
```

Question	Answer	Marks
(b)	**One** mark for each point • greater than or equal to 35 • and less than or equal to 38	2
(c)	**One** mark for each correct column **max two**	2

Temperature	**OUTPUT**
	(Please enter temperature)
34.22	Temperature too low
	(Please enter temperature)
36.1	Temperature normal
	(Please enter temperature)
37.4	Temperature normal
	(Please enter temperature)
38.0	Temperature normal
	(Please enter temperature)
999	

Q187)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td><b>One</b> mark per mark point (<b>max five</b>)
• Correct <code>Value</code> column
• Correct <code>Count</code> column
• Correct <code>Answer</code> column (down to first OUTPUT (120)) – Shaded grey
• Correct <code>Answer</code> column (remaining rows)
• Correct OUTPUT column
<table>
<tr><th><code>Value</code></th><th><code>Count</code></th><th><code>Answer</code></th><th>OUTPUT</th></tr>
<tr><td>5</td><td></td><td>5</td><td></td></tr>
<tr><td></td><td>4</td><td>20</td><td></td></tr>
<tr><td></td><td>3</td><td>60</td><td></td></tr>
<tr><td></td><td>2</td><td>120</td><td></td></tr>
<tr><td></td><td>1</td><td>120</td><td>120</td></tr>
<tr><td>6</td><td></td><td>6</td><td></td></tr>
<tr><td></td><td>5</td><td>30</td><td></td></tr>
<tr><td></td><td>4</td><td>120</td><td></td></tr>
<tr><td></td><td>3</td><td>360</td><td></td></tr>
<tr><td></td><td>2</td><td>720</td><td></td></tr>
<tr><td></td><td>1</td><td>720</td><td>720</td></tr>
<tr><td>−1</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>5</td></tr>
<tr><td>(b)</td><td><b>One</b> mark for correct answer
Example:
• It multiplies the input number by one less than itself repeatedly, until the value is 1
• It calculates the number of permutations of all the numbers up to the input value</td><td>1</td></tr>
<tr><td>(c)</td><td><b>One</b> mark per mark point (<b>max two</b>)
• The program would accept the value and enter the <code>FOR</code> loop
• <code>Count</code> would keep reducing by 1 and would never reach 1, as it would already be less than 1
• There would be an endless loop</td><td>2</td></tr>
</table>

Q188)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>My!Hidden
<table>
<tr><th>Password</th><th>Accept</th><th>Index</th><th>Found</th><th>OUTPUT</th></tr>
<tr><td>My!Hidden</td><td>TRUE</td><td></td><td></td><td>(Please enter password)</td></tr>
<tr><td></td><td></td><td>1</td><td>FALSE</td><td></td></tr>
<tr><td></td><td></td><td>2</td><td>FALSE</td><td></td></tr>
<tr><td></td><td></td><td>3</td><td>TRUE</td><td></td></tr>
<tr><td></td><td></td><td>4</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>Accepted</td></tr>
</table>
For each trace table, <b>one mark</b> Password and output, <b>one mark</b> Accept, Index and Found columns</td><td></td></tr>
<tr><td>(b)</td><td><b>One</b> mark for each correct point
<ul>
<li>maximum length 20 characters <b>and</b> minimum length 8 characters</li>
<li>cannot be all upper-case letters or all lower-case letters // must contain upper-case and lower-case letters</li>
<li>must contain an !</li>
</ul></td><td>3</td></tr>
</table>