<table>
<tr><th colspan="2">Question</th><th colspan="2">Marks</th></tr>
<tr><td>1</td><td>1</td><td>4 marks for AO3 (design) and 8 marks for AO3 (programming)

Mark Scheme

<table>
<tr><th>Level</th><th>Description</th><th>Mark Range</th></tr>
<tr><td>4</td><td>A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that meets most of the requirements. All of the appropriate design decisions have been taken. To award 12 marks, all of the requirements must be met.</td><td>10–12</td></tr>
<tr><td>3</td><td>There is evidence that a line of reasoning has been followed to produce a logically structured program. The program displays relevant prompts, inputs the two words and includes one iterative structure and two selection structures. An attempt has been made to check that all the characters in the first word are in the second word, although this may not work correctly under all circumstances. The solution demonstrates good design work as most of the correct design decisions have been made.</td><td>7–9</td></tr>
<tr><td>2</td><td>A program has been written and some appropriate, syntactically correct programming language statements have been written. There is evidence that a line of reasoning has been partially followed as although the program may not have the required functionality, it can be seen that the response contains some of the statements that would be needed in a working solution. There is evidence of some appropriate design work as the response recognises at least one appropriate technique that could be used by a working solution, regardless of whether this has been implemented correctly.</td><td>4–6</td></tr>
<tr><td>1</td><td>A program has been written and a few appropriate programming language statements have been written but there is no evidence that a line of reasoning has been followed to arrive at a working solution. The statements written may or may not be syntactically correct. It is unlikely that any of the key design elements of the task have been recognised.</td><td>1–3</td></tr>
</table>

Guidance

Evidence of AO3 design – 4 points:

Evidence of design to look for in responses:

1. Identifying that a selection structure is needed after all letter counts have been compared to output a message saying it can be made from the letters in the 2_{nd} word or that it can't
2. Identifying that a loop is needed that repeats a number of times based on the</td><td>12</td></tr>
</table>

		length of the first word // identifying that a loop is needed that repeats 26 times // identifying that a loop is needed that repeats a number of times determined by the number of unique characters in the first word 3. Identifying that the number of times a letter occurs in the first string needs to be less than or equal to the number of times it occurs in the second string 4. Boolean (or equivalent) variable used to indicate if the first word can be formed from the letters in the second word // array of suitable size to store the count of each letter Note that AO3 (design) points are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works. **Evidence for AO3 programming – 8 points:** Evidence of programming to look for in response: 5. (Suitable prompts asking user to enter the two words followed by) user inputs being assigned to appropriate variables (**R.** if inside or after iterative structure), two variables with appropriate data types created to store the two words entered by the user 6. Iterative structure to look at each letter in first word has correct syntax and start/end conditions // iterative structure to look at each letter in the alphabet has correct syntax and start/end conditions 7. Correctly counts the number of times that a letter occurs in one of the words 8. Selection structure that compares the count of a letter in the first word with the count of that letter in the second word **A.** incorrect counts **A.** incorrect comparison operator 9. Correctly counts the number of times each letter in one of the two words occurs 10. Program works correctly if the two words entered are the same 11. Program works correctly when first word contains more instances of a letter than there are in the second word (i.e. says that it cannot be formed from the second word) 12. Program works correctly for all word pairs consisting of just upper case letters **Alternative mark scheme** (based on removing an instance of a letter from the 2_{nd} word each time it appears in the 1_{st} word) 1. Identifying that a selection structure is needed after all the letters that appear in both words have been removed from the first word to output a message saying it can be made from the letters in the second word or that it can't 3. Identifying that a letter can be removed from the second word if it appears in the first word 7. Selection structure that checks if letter in first word appears in the second word 8. Removes a letter from the second word if it appears in the first word. 9. Sets indicator to false if a letter does not appear in the second word	
1	**2**	**Mark is for AO3 (evaluate)** ****** SCREEN CAPTURE ****** *Must match code from question 01.1, including prompts on screen capture matching*	**1**

		those in code. *Code for question 01.1 must be sensible.* Screen captures showing: • the string NINE being entered followed by the string ELEPHANTINE and then a message displayed saying that the first word can be formed from the second. • the string NINE being entered followed by the word ELEPHANT and then a message displayed saying that the first work cannot be formed from the second.	

Question			Marks
2	1	**Mark is for AO1 (understanding)** Local variables have more limited scope; Global variables exist throughout the entire program; Local variables only exist in a part/block/subroutine of the program; Local variables can only be accessed in a part/block/subroutine of the program; Global variables can be accessed from any part of the program; **Max 1 mark**	**1**
2	2	**Mark is for AO1 (knowledge)** Modularisation of a program; Allows reuse of subroutines; Less chance of side-effects; **A.** advantages resulting from modularisation eg easier to test each subroutine independently **Max 1 mark**	**1**

<table>
<tr><th>Question</th><th></th><th></th><th>Marks</th></tr>
<tr><td>3</td><td>1</td><td>

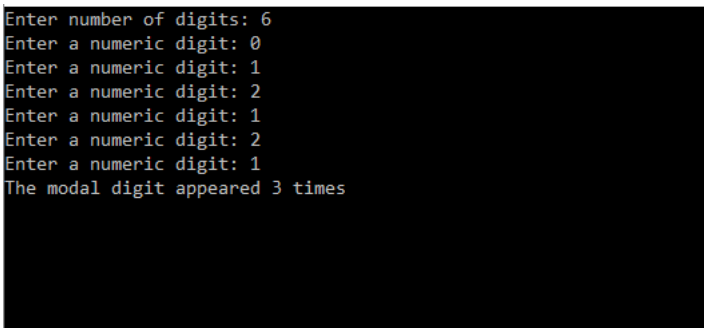
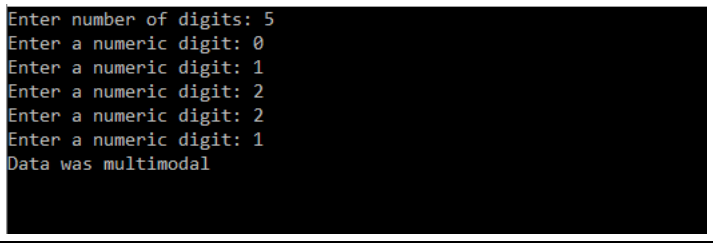
4 marks for AO3 (design) and 8 marks for AO3 (programming)

Mark Scheme

<table>
<tr><th>Level</th><th>Description</th><th>Mark Range</th></tr>
<tr><td>4</td><td>A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that meets most of the requirements. All of the appropriate design decisions have been taken. To award 12 marks, all of the requirements must be met.</td><td>10–12</td></tr>
<tr><td>3</td><td>There is evidence that a line of reasoning has been followed to produce a logically structured program. The program displays relevant prompts, inputs the required numbers, at least one iterative structure and one selection structure and suitable data structure(s) to store the numbers entered and the frequencies. An attempt has been made to determine the modal frequency, although this may not work correctly under all circumstances. The solution demonstrates good design work as most of the correct design decisions have been made.</td><td>7–9</td></tr>
<tr><td>2</td><td>A program has been written and some appropriate, syntactically correct programming language statements have been written. There is evidence that a line of reasoning has been partially followed as although the program may not have the required functionality, it can be seen that the response contains some of the statements that would be needed in a working solution. There is evidence of some appropriate design work as the response recognises at least one appropriate technique that could be used by a working solution, regardless of whether this has been implemented correctly.</td><td>4–6</td></tr>
<tr><td>1</td><td>A program has been written and a few appropriate programming language statements have been written but there is no evidence that a line of reasoning has been followed to arrive at a working solution. The statements written may or may not be syntactically correct. It is unlikely that any of the key design elements of the task have been recognised.</td><td>1–3</td></tr>
</table>

</td><td>12</td></tr>
</table>

		Guidance **Evidence of AO3 design – 4 points:** Evidence of design to look for in responses: 1. Identifying that data structure(s) are needed to store ten frequencies 2. Identifying that a loop is needed that repeats a number of times determined by the first number entered by the user 3. Identifying that a Boolean (or equivalent) variable is needed to store if the data was multimodal 4. Selection structure that either outputs a calculated number (**I.** incorrectly calculated) or a message saying "`Data was multimodal`" (**A.** any suitable message) Note that AO3 (design) points are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works. **Evidence for AO3 programming – 8 points:** Evidence of programming to look for in response: 5. Suitable prompts asking user to enter the number of digits followed by user inputs being assigned to appropriate variable **R.** if inside or after iterative structure 6. Correct number of numeric digits obtained from the user 7. Adds one to correct frequency count **R**. if only works for one digit 8. Selection structure, inside iterative structure, that correctly compares calculated frequency (**I.** incorrect frequency) of a digit with the highest frequency found so far 9. Boolean (or equivalent) variable that is used to indicate if data is multimodal is set to true under correct circumstances 10. Boolean (or equivalent) variable that is used to indicate if data is multimodal is set to false when new higher frequency is found 11. Program works correctly if the data has more than one modal value **A.** any sensible message 12. Program displays the correct frequency of the modal value under all circumstances and does not say data is multimodal when it is not **I.** frequency being displayed when data is multimodal **Max 11** if code contains any errors	
3	**2**	**Mark is for AO3 (evaluate)** ****** SCREEN CAPTURE ****** *Must match code from **3.1**, including prompts on screen capture matching those in code.* *Code for **3.1** must be sensible.* Screen captures showing:	**1**

		• the number 6 being entered followed by the numbers 0, 1, 2, 1, 2 and 1 (**I.** order of these six numbers) and then a message displayed saying 3 • the number 5 being entered followed by the numbers 0, 1, 2, 2 and 1 (**I.** order of these five numbers) and then a message displayed saying that the data is multimodal.   	

Question			Marks
4	1	**Mark is for AO1 (understanding)** A subroutine that calls itself;	1
4	2	**Mark is for AO1 (understanding)** The circumstance(s) when a recursive subroutine does not call itself;	1
4	3	**All marks AO1 (knowledge)** local variables; return address; parameters; register values; **A.** example of register that would be in stack frame **Max 2**	2

Question			Marks
5	1	**4 marks for AO3 (design) and 8 marks for AO3 (programming)** **Mark Scheme**	12

Level	Description	Mark Range
4	A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that meets most of the requirements. All of the appropriate design decisions have been taken. To award 12 marks, all of the requirements must be met.	10–12
3	There is evidence that a line of reasoning has been followed to produce a logically structured program. The program displays relevant prompts, inputs the required number, has at least one iterative structure and one selection structure and uses appropriate variables to store most of the needed data. An attempt has been made to determine if a number is a Harshad number, although this may not work correctly under all circumstances. The solution demonstrates good design work as most of the correct design decisions have been made.	7–9
2	A program has been written and some appropriate, syntactically correct programming language statements have been written. There is evidence that a line of reasoning has been partially followed as although the program may not have the required functionality, it can be seen that the response contains some of the statements that would be needed in a working solution. There is evidence of some appropriate design work as the response recognises at least one appropriate technique that could be used by a working solution, regardless of whether this has been implemented correctly.	4–6
1	A program has been written and a few appropriate programming language statements have been written but there is no evidence that a line of reasoning has been followed to arrive at a working solution. The statements written may or may not be syntactically correct. It is unlikely that any of the key design elements of the task have been recognised.	1–3

Guidance

Evidence of AO3 design – 4 points:

Evidence of design to look for in responses:

1. Identifying that integer division is needed when calculating the sum of the digits // identifying that a character in string needs to be converted to a number data type when calculating the sum of the digits
2. Identifying that a loop is needed that repeats a number of times determined by the number entered by the user // identifying that a loop is needed that repeats until the *n*th Harshad number is found
3. Identifying that nested iteration is needed
4. Selection structure that compares sum of digits (**I.** incorrectly calculated) with a number

Note that AO3 (design) points are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.

Evidence for AO3 programming – 8 points:

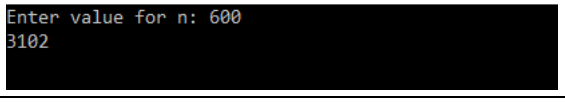
Evidence of programming to look for in response:

5. Suitable prompt asking user to enter a number followed by user input being assigned to appropriate variable
6. Iterative structure that repeats a number of times sufficient to find all the digits of a number
7. Calculates the sum of all the digits of a number
8. Calculates the remainder from dividing a number by its sum of digits **A.** incorrect calculation for sum of digits
9. Resets the variable used to store the sum of digits to 0 in an appropriate place
10. Program works correctly for the first nine Harshad numbers (1 to 9)
11. Program will display 10/12/18 if the user enters the number 10/11/12
12. Program displays the correct value for the *n*th Harshad number under all circumstances **I.** displaying Harshad numbers that appear before the *n*th Harshad number

Alternative mark scheme
This mark scheme is to be used if solution uses a recursive subroutine to calculate the sum of the digits.

3. Identifying that a recursive subroutine is needed to calculate the sum of the digits.
6. Recursive subroutine has an appropriate base case.
9. Sets the variable used to store the sum of digits to the result returned by the call to the recursive subroutine in an appropriate place.

Max 11 if any errors.

5	**2**	**Mark is for AO3 (evaluate)** ****** SCREEN CAPTURE ****** *Must match code from **5.1**, including prompts on screen capture matching those in code.* *Code for **5.1** must be sensible.* Screen capture showing the number 600 being entered and then a message displayed saying 3102 	**1**

Question			Marks
6	1	**Mark is for AO2 (analyse)** Statement 1 can't be correct because it means Statement 5 / Statement 6 is true which means Statement 1 is false; Statement 1 can't be correct because it would mean Statement 2 is correct which would mean all of the other statements have to be both correct and incorrect; Statement 1 can't be correct because it would mean Statement 4 is correct which means that Statements 2 and 3 have to be both correct and incorrect; Questions says only one of the statements is true so Statement 1 can't be true as that means more than one statement would be true; **Max 1**	**1**
6	2	**Mark is for AO2 (analyse)** (Statement) 5;	**1**
6	3	**All marks AO2 (analyse)** Statement 3 can't be correct because Statement 1 is false; Statement 3 can't be correct because the question says only one of the statements is correct; Statement 3 can't be correct because that would mean Statement 2 would be a contradiction as this would mean Statement 3 would have to be incorrect; If Statement 2 is true then Statement 4 has to be false. As Statements 1 and 3 are false for Statement 4 to be false Statement 2 has to be false as well (otherwise one of the above would be true). This is a contradiction so Statement 2 can't be true; Statements 1, 2 and 3 are false so Statement 4 is false; If Statement 6 is true then 5 has to be false implying at least one of Statements 1 to 4 have to be true but they are all false so Statement 6 has to be false; **Max 2**	**2**

Question			Marks
7	1	**Mark is for AO1 (knowledge)** A subroutine that calls itself;	1

<table>
<tr><th colspan="2">Question</th><th></th><th>Marks</th></tr>
<tr><td>8</td><td>1</td><td>

4 marks for AO3 (design) and 8 marks for AO3 (programming)

Mark Scheme

<table>
<tr><th>Level</th><th>Description</th><th>Mark Range</th></tr>
<tr><td>4</td><td>A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that meets most of the requirements. All of the appropriate design decisions have been taken. To award 12 marks, all of the requirements must be met.</td><td>10–12</td></tr>
<tr><td>3</td><td>There is evidence that a line of reasoning has been followed to produce a logically structured program. The program displays relevant prompts, inputs the required string, has at least one iterative structure and at least one selection structure and uses appropriate variables to store most of the needed data. An attempt has been made to swap the positions of vowels in the string, although this may not work correctly under all circumstances. The solution demonstrates good design work as most of the correct design decisions have been made.</td><td>7–9</td></tr>
<tr><td>2</td><td>A program has been written and some appropriate, syntactically correct programming language statements have been written. There is evidence that a line of reasoning has been partially followed as although the program may not have the required functionality, it can be seen that the response contains some of the statements that would be needed in a working solution. There is evidence of some appropriate design work as the response recognises at least one appropriate technique that could be used by a working solution, regardless of whether this has been implemented correctly.</td><td>4–6</td></tr>
<tr><td>1</td><td>A program has been written and a few appropriate programming language statements have been written but there is no evidence that a line of reasoning has been followed to arrive at a working solution. The statements written may or may not be syntactically correct. It is unlikely that any of the key design elements of the task have been recognised.</td><td>1–3</td></tr>
</table>

</td><td>12</td></tr>
</table>

Guidance

Evidence of AO3 design – 4 points:

Evidence of design to look for in responses:

1. Identifying that string concatenation is needed when swapping vowels in the string // identifying that swapping items in a list of characters is needed.
2. Identifying that a loop is needed that repeats a number of times determined by the word entered by the user // identifying that a loop is needed that repeats a number of times determined by the number of vowels in the word entered by the user.
3. Identifying that two integer variables are needed to store positions of characters in the string // identifying that an ordered list of vowels in the string needs to be created // identifying one integer variable is needed to show the distance from the start and end of the string (**R.** if no attempt to use this integer with the start and end positions of the string).
4. Selection structure that checks if a character is a vowel **A.** more than one selection structure used **R.** if no attempt at comparing with each of the five vowels.

Note that AO3 (design) points are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.

Evidence for AO3 programming – 8 points:

Evidence of programming to look for in response:

5. Suitable prompt asking user to enter a string followed by user input being assigned to appropriate variable.
6. Iterative structure that repeats a number of times that is sufficient to check all the characters in the string.
7. Correctly checks if a character is a vowel.
8. Correctly checks all characters in the string to see if they are vowels.
9. Swaps/moves the position of two characters in the string.
10. Program only moves/changes the position of vowels.
11. Program works correctly if a string contains one vowel and works correctly if a string contains no vowels. **R.** if program does not attempt to swap positions of vowels or identify that there are less than two vowels.
12. Program works correctly under all circumstances.

I. additional loop to get program to repeat multiple times.

DPT. mark points 7 and 8 if only checks for some vowels or includes at most one non-vowel character.

Max 11 if any errors

Question			Marks
8	2	**Mark is for AO3 (evaluate)** ****** SCREEN CAPTURE ****** *Must match code from **8.1**, including prompts on screen capture matching those in code.* *Code for **8.1** must be sensible.* Screen capture showing the string `persepolis` being entered and then the string `pirsopeles` being displayed and screen capture showing the string `darius` being entered and then the string `durias` being displayed and screen capture showing the string `xerxes` being entered and then the string `xerxes` being displayed; **I.** order of tests   	1

Question			Marks
9	1	**Mark is for AO1 (understanding)** Allows direct (**A.** faster) access to the value being looked-up // No need to search through the list to find a value (assuming a good choice of hash function);	1

Question			Marks
10	1	**Mark is for AO2 (analyse)** The colour is not yellow // the chosen shape was not the yellow circle // the colour is blue or pink;	1
10	2	**Mark is for AO2 (analyse)** The shape is not a square // the chosen shape was not the blue square // the shape is a triangle or circle;	1
10	3	**Mark is for AO2 (analyse)** Pink triangle;	1

Question			Marks
11	1	**4 marks for AO3 (design) and 8 marks for AO3 (programming)** **Mark Scheme**	12

Level	Description	Mark Range
4	A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that meets most of the requirements. All of the appropriate design decisions have been taken. To award 12 marks, all of the requirements must be met.	10–12
3	There is evidence that a line of reasoning has been followed to produce a logically structured program. The program displays relevant prompts, inputs the required string, has at least one iterative structure and at least one selection structure and uses appropriate variables to store most of the needed data. An attempt has been made to test for most of the criteria for a valid string, although these may not work correctly under all circumstances. The solution demonstrates good design work as most of the correct design decisions have been made.	7–9
2	A program has been written and some appropriate, syntactically correct programming language statements have been written. There is evidence that a line of reasoning has been partially followed as although the program may not have the required functionality, it can be seen that the response contains some of the statements that would be needed in a working solution. There is evidence of some appropriate design work as the response recognises at least one appropriate technique that could be used by a working solution, regardless of whether this has been implemented correctly.	4–6
1	A program has been written and a few appropriate programming language statements have been written but there is no evidence that a line of reasoning has been followed to arrive at a working solution. The statements written may or may not be syntactically correct. It is unlikely that any of the key design elements of the task have been recognised.	1–3

Guidance

Evidence of AO3 design – 4 points:

Evidence of design to look for in responses:

1. Identifying that an iteration structure is needed that repeats a number of times based on the length of the string entered by the user.
2. Identifying that nested iteration is needed.
3. Identifying that an integer variable is needed to store the sum of the ASCII codes and that Boolean variable(s) (**A.** any suitable equivalent) are needed to track if there are duplicate characters and non-uppercase characters (**R.** if no attempt to use the Boolean variable (or equivalent) to indicate the result of at least one validation check).
4. Selection structure that checks if two characters in the string are the same **R.** if not inside their iteration structure (or equivalent)

Note that AO3 (design) points are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.

Evidence for AO3 programming – 8 points:

Evidence of programming to look for in response:

5. User input being assigned to appropriate variable.
6. Correctly gets the ASCII code for a character.
7. Adds ASCII code for character to a total.
8. Correctly checks if every character is uppercase. **A.** checks every character is not lowercase
9. Correctly checks if a character is duplicated. **R.** if only checks if a character is a duplicate for some of the other characters in the string **R.** if will always say a character is a duplicate
10. Iteration structure that repeats until string is valid. **A.** if some validation checks are missing or incorrect **R.** if subsequent iterations would not work in same way e.g. because Boolean variables not reset inside iteration structure
11. Program rejects all strings that are less than five characters or more than seven characters in length.
12. Program works correctly under all circumstances.

Max 11 if any errors

Question			Marks

11	2	**Mark is for AO3 (evaluate)**
		
		****** SCREEN CAPTURE ******
		*Must match code from **11.1**, including prompts on screen capture(s) matching those in code.*
		*Code for **11.1** must be sensible.*
		
		Screen captures showing the string(s) entered and result(s) of each of the tests;
		I. order of tests
		A. tests done individually or done as one extended test
		
		`Enter a string: BOIL`
		`not valid`
		`Enter a string: BRAisE`
		`not valid`
		`Enter a string: ROAST`
		`not valid`
		`Enter a string: BLANCH`
		`valid`
		
		`Enter a string: PRESSURECOOK`
		`not valid`
		`Enter a string:`
		
		Note for examiners: example screen captures shown here match the order of the test data given in the question but there is no requirement for the tests to be done in any particular order.
			1

Question			Marks
12		**All marks AO1 (understanding)** **Example answers** Easier to test/debug as each subroutine can be tested separately; Easier to understand the code if sensible identifiers are used for subroutine names; Code can be easily reused as each subroutine is independent of rest of program; Makes it easier to work as a team of programmers as each subroutine can be worked on independently; Can be used as often as needed without having to write the code each time; Makes program easier to maintain/update (in future) as fewer changes will need to be made to make an update; Allows the use of recursive techniques because subroutines can call themselves; Easier to understand the code as each subroutine can be considered in isolation; Less likely to be errors in the code due to reuse of code; Reduces/eliminates side effects (eg unexpected change to value in global variable) through use of local variables; **Max 3** **Notes for examiners** Each advantage must be different. Mark should only be awarded if there is an explanation of how the advantage is achieved. Mark should only be awarded if the explanation is relevant for the stated advantaged.	3

Question			Marks
13	1	**4 marks for AO3 (design) and 8 marks for AO3 (programming)** **Mark Scheme**	12

Level	Description	Mark Range
4	A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that meets most of the requirements. All of the appropriate design decisions have been taken. To award 12 marks, all of the requirements must be met.	10–12
3	There is evidence that a line of reasoning has been followed to produce a logically structured program. The program displays relevant prompts, inputs the required data, has at least one iterative structure and at least one selection structure and uses appropriate variables to store most of the needed data. An attempt has been made to test for increasing and decreasing numbers, although this may not work correctly under all circumstances. The solution demonstrates good design work as most of the correct design decisions have been made.	7–9
2	A program has been written and some appropriate, syntactically correct programming language statements have been written. There is evidence that a line of reasoning has been partially followed as, although the program may not have the required functionality, it can be seen that the response contains some of the statements that would be needed in a working solution. There is evidence of some appropriate design work as the response recognises at least one appropriate technique that could be used by a working solution, regardless of whether this has been implemented correctly.	4–6
1	A program has been written and a few appropriate programming language statements have been written, but there is no evidence that a line of reasoning has been followed to arrive at a working solution. The statements written may or may not be syntactically correct. It is unlikely that any of the key design elements of the task have been recognised.	1–3

		Guidance **Evidence of AO3 design – 4 points:** Evidence of design to look for in responses: 1. Identifying that an iteration structure is needed that repeats a number of times based on the number of digits in the number entered by the user. 2. Identifying that selection structures (**A.** equivalent) for the three possible outcomes (bouncy, not bouncy, perfectly bouncy) is needed. 3. Identifying that variables with suitable data types are needed to store the number of digits followed by a larger digit and the number of digits followed by a smaller digit (**A.** any suitable equivalent). 4. Recognising the need to use input as an integer for the indefinite iteration and as a string to access individual digits // attempting to use remainder division with a power of ten. Note that AO3 (design) points are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works. **Evidence for AO3 programming – 8 points:** Evidence of programming to look for in response: 5. User input being assigned to appropriate variable. **A.** array of integers as long as at least 8 digits are allowed. 6. Indefinite iteration with correct condition containing attempt to get user input. 7. Iteration structure that repeats the correct number of times (one less than number of digits). 8. Compares two consecutive digits. 9. Selection structure with no incorrect contents for when next digit is larger than current digit. **R.** if any incorrect conditions. 10. Selection structure with no incorrect contents for when next digit is less than current digit. **R.** if any incorrect conditions. 11. Correctly detects that a number with all digits the same is an increasing number and correctly detects that a number with all digits the same is a decreasing number // correctly detects that a number with all digits the same is not a bouncy number 12. Selection structure(s) (**A.** equivalent) after iterative structure – for the three possible outcomes (bouncy, not bouncy, perfectly bouncy). **A.** would output messages that a perfectly bouncy number is perfectly bouncy and also bouncy. **R.** if bouncy number would result in output of perfectly bouncy. **R.** if no attempt made to detect bouncy number **R.** if no attempt made to detect perfectly bouncy number **Max 11** if any errors	

13	**2**	**Mark is for AO3 (evaluate)**
		
		****** SCREEN CAPTURE ******
		*Must match code from **13.1**, including messages on screen capture(s) matching those in code.*
		*Code for **13.1** must be sensible.*
		
		Screen captures showing the integer(s) entered and result(s) of each of the tests;
		I. order of tests
		
		`Enter a number: -3`
		`Enter a number: 14982`
		`perfectly bouncy`
		`>>>`
		`Enter a number: 1234`
		`not bouncy`
		`>>>`
		
		Note for examiners: example screen captures shown here match the order of the test data given in the question but there is no requirement for the tests to be done in any particular order.
			1

Question			Marks
14	1	**Mark is for AO1 (knowledge)** Joining (two) strings (to form one string // into one);	1