

Question		Marks																
1	1	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>	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 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.	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	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 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.	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 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																		

		<p>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</p> <ol style="list-style-type: none"> 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 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 <p>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.</p> <p>Evidence for AO3 programming – 8 points:</p> <p>Evidence of programming to look for in response:</p> <ol style="list-style-type: none"> (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 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 Correctly counts the number of times that a letter occurs in one of the words 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 Correctly counts the number of times each letter in one of the two words occurs Program works correctly if the two words entered are the same 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) Program works correctly for all word pairs consisting of just upper case letters <p>Alternative mark scheme (based on removing an instance of a letter from the 2nd word each time it appears in the 1st word)</p> <ol style="list-style-type: none"> 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 Identifying that a letter can be removed from the second word if it appears in the first word Selection structure that checks if letter in first word appears in the second word Removes a letter from the second word if it appears in the first word. Sets indicator to false if a letter does not appear in the second word 	
1	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from question 01.1, including prompts on screen capture matching</i></p>	1

		<p><i>those in code.</i></p> <p><i>Code for question 05.1 must be sensible.</i></p> <p>Screen captures showing:</p> <ul style="list-style-type: none">• 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	<p>All marks for AO3 (programming)</p> <p>1. Creates a random number;</p> <p>2. Selection structure with random number used in condition;</p> <p>3. Selection structure with one message in then part and one message in else part – one of the messages must be the original message “Sorry, you don’t know how to ***.” and one must be the new message “Sorry, I don’t know what *** means.”;</p> <p>R. other messages R. if spacing incorrect I. case I. punctuation A. answers that use two selection structures as long as they are equivalent to using an if...then...else structure</p> <p>4. Each message has probability of being displayed 50% of the time; A. any suitable message A. answers with value between 0 and just less than 1 is generated where 0.5 is rounded incorrectly</p> <p>Max 3 if code contains errors Max 2 if both error messages could be displayed sometimes</p>
2	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from question 2.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for question 2.1 must be sensible.</i></p> <p>Screen captures showing the command <code>eat</code> being entered (I. any text after the <code>eat</code> command) followed by one of the two messages – this should be done at least twice and there must be evidence that both messages can be displayed;</p>

Question		Marks
3	1	<p>All marks for AO3 (programming)</p> <ol style="list-style-type: none"> 1. Iterative structure to loop through each item in <code>Items</code>; 2. Selection structure inside iterative structure with valid syntax and correct condition for selection structure that compares player's ID (1001) / <code>Inventory</code> with location of an item; 3. One added to appropriately-named variable used to count number of objects in inventory; R. if not inside selection structure inside iterative structure 4. Selection structure, after attempt at iterative structure, that compares count of items in inventory (A. incorrect count) with the number 5 (A. alternative logic e.g. <code>> 4</code>); R. if incorrect logic 5. Message inside attempt at selection structure from mark point 4 saying that player can't carry any more; A. selection structure in wrong place in code 6. If the number of items in the inventory is fewer than five then code added does not prevent item from being added to inventory; Note for examiners: this mark can only be awarded if mark points 1 and 4 have been awarded 7. If the number of items in the inventory is five (or more) then the item is not added to the inventory, the item stays in its current location and the result of getting the item is not executed; A. other values to five for number of items in inventory based on incorrect answer for mark point 4 <p>Max 6 marks if code contains errors</p>
3	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from question 3.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for question 3.1 must be sensible.</i></p> <p>Screen capture(s) showing that the red die and torch are picked up by the player but not the book;</p>

Question		Marks
4	1	<p>All marks for AO3 (programming)</p> <p>12</p> <ol style="list-style-type: none"> 1. Creating a new subroutine called <code>DropItem</code>; R. other names for subroutine I. case 2. Adding new option to the selection structure in <code>PlayGame</code> for the drop command; 3. Call to <code>DropItem</code> inside the option added for mark point 2; R. if name does not match name of created subroutine R. if parameter list for subroutine call does not match parameter list for new subroutine 4. Parameter list for the new subroutine and contains <code>Items</code>, the item to drop and the current location of the player; I. additional parameters that are not needed A. alternatives to these parameters as long as evidence of attempt to get them to be usable is in code eg passing <code>Characters</code> instead of just the location of the player as long as some code to extract the location is included in the new subroutine <p>The following all relate to the <code>DropItem</code> subroutine:</p> <ol style="list-style-type: none"> 5. Gets the index of the item to drop; 6. Selection structure that checks if the item to drop does exist and results in appropriate error message being displayed if it doesn't; 7. Selection structure that checks if the item to drop is in the player's inventory and results in appropriate error message being displayed if it isn't; 8. Selection structure that checks if item to drop is fragile; 9. If item is in player's inventory and is fragile an appropriate message is displayed; A. incorrect conditions for mark points 7 and/or 8 10. If item is in player's inventory and is fragile then item is removed from <code>Items</code> // if item is in player's inventory and is fragile then the location of the item is changed to a location that does not exist; A. incorrect conditions for mark points 7 and/or 8 11. Location of item to drop is changed to the current location if it is in the player's inventory and is not fragile and an appropriate message is displayed; A. incorrect conditions for mark points 7 and/or 8 A. no attempts for mark points 7 and/or 8 12. Logic for mark points 6 –11 is correct, program won't display any incorrect messages and does not try to access position -1 in <code>Items</code> if item does not exist; <p>Max 11 if code contains errors</p>
4	2	<p>Mark is for AO3 (evaluate)</p> <p>1</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from question 4.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for question 4.1 must be sensible.</i></p> <p>Screen capture(s) showing that the player's inventory contains just the flask and that the contents of the room are the apple, torch and red die;</p>

Question		Marks
5	1	<p>All marks for AO3 (programming)</p> <ol style="list-style-type: none"> 1. Code modified to roll the player's die three times; 2. Appropriate data structure(s) / variables to store the results of the player's dice rolls; 3. Code identifies the highest/smallest of the three numbers rolled by the player; 4. Code multiplies one of the results of the player's dice rolls by 100, another by 10 and adds the results of these two multiplications to the result of the other die roll; 5. Correct calculation of the player's score; 6. Code modified to roll the other character's die three times; 7. Correct calculation of the other character's score; 8. All expected messages, including messages showing the result of each die roll, displayed under the expected circumstances <p>Max 7 if code contains errors or if other parts of the subroutine no longer work correctly</p>
5	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from question 5.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for question 5.1 must be sensible.</i></p> <p>Screen capture(s) showing two tests with correct scores calculated for both player and other character and correct result displayed; A. missing results of individual die rolls not displayed</p>

Question			Marks															
6	1	4 marks for AO3 (design) and 8 marks for AO3 (programming)	12															
		<u>Mark Scheme</u>																
		<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>		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 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.	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
		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 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.	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																

		<p><u>Guidance</u></p> <p>Evidence of AO3 design – 4 points:</p> <p>Evidence of design to look for in responses:</p> <ol style="list-style-type: none"> 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) <p>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.</p> <p>Evidence for AO3 programming – 8 points:</p> <p>Evidence of programming to look for in response:</p> <ol style="list-style-type: none"> 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 <p>Max 11 if code contains any errors</p>	
6	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 06.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for 06.1 must be sensible.</i></p> <p>Screen captures showing:</p>	1

- the number 6 being entered followed by the numbers 0, 1, 2, 1, 2 and 1 (i.e. 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.e. order of these five numbers) and then a message displayed saying that the data is multimodal.

```
Enter number of digits: 6
Enter a numeric digit: 0
Enter a numeric digit: 1
Enter a numeric digit: 2
Enter a numeric digit: 1
Enter a numeric digit: 2
Enter a numeric digit: 1
The modal digit appeared 3 times
```

```
Enter number of digits: 5
Enter a numeric digit: 0
Enter a numeric digit: 1
Enter a numeric digit: 2
Enter a numeric digit: 2
Enter a numeric digit: 1
Data was multimodal
```

Question		Marks
7	1	<p>All marks for AO3 (programming)</p> <p>1. Indefinite iterative structure contains code that gets the name from the user; 2. One correct condition; 3. Both correct conditions and correct logic for the iterative structure; 4. Displays error message if no name is entered // displays error message if a name that has already been used is entered; 5. Displays error message under all correct circumstances and only under correct circumstances;</p> <p>Max 4 if code contains errors</p>
7	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 7.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for 7.1 must be sensible.</i></p> <p>Screen captures showing error message(s) being shown for the two invalid names and then showing the message asking for the starting balance when a valid name is entered;</p> <pre>Enter L for a large settlement, anything else for a normal size settlement: Enter D for default companies, anything else to add your own start companies: D ***** ***** MENU ***** ***** 1. Display details of households 2. Display details of companies 3. Modify company 4. Add new company 6. Advance to next day Q. Quit Enter your choice: 4 Enter a name for the company: You must enter a name. Enter a name for the company: AQA Burgers That name is already being used. Enter a name for the company: In Jest Enter the starting balance for the company:</pre>

Question		Marks
8	1	<p>All marks for AO3 (programming)</p> <ol style="list-style-type: none"> 1. Creating a new class called <code>AffluentHousehold</code>; R. other names for class I. case and minor typos 2. New class inherits from <code>Household</code>; 3. Constructor created that overrides base class constructor with call made to base class constructor; R. if incorrect parameters 4. Sets the value of <code>ChanceEatOutPerDay</code> to 1; R. if before call to base class constructor R. If not after attempt at call to base class constructor <p>The following all relate to the <code>AddHousehold</code> method:</p> <ol style="list-style-type: none"> 5. Selection structure with correct condition; 6. Creates an <code>AffluentHousehold</code> object; R. if it also creates a household 7. Creates an <code>AffluentHousehold</code> under the correct circumstances and a <code>Household</code> under the correct circumstances; R. if new household not added to <code>Households</code> <p>Max 6 if code contains errors</p>
8	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 8.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for 8.1 must be sensible.</i></p> <p>Screen capture(s) showing that households with an X value less than 100 have an eat out percentage of 1;</p> <pre> 225 Coordinates: (123, 32) Eat out percentage: 0.8259652 226 Coordinates: (317, 914) Eat out percentage: 0.845291 227 Coordinates: (77, 743) Eat out percentage: 1 228 Coordinates: (681, 434) Eat out percentage: 0.3261211 229 Coordinates: (886, 440) Eat out percentage: 0.2608214 230 Coordinates: (786, 939) Eat out percentage: 0.230395 231 Coordinates: (296, 716) Eat out percentage: 0.2893967 232 Coordinates: (6, 735) Eat out percentage: 1 233 Coordinates: (809, 465) Eat out percentage: 0.5536526 234 Coordinates: (560, 411) Eat out percentage: 0.1806425 235 Coordinates: (88, 158) Eat out percentage: 1 236 Coordinates: (999, 865) Eat out percentage: 0.3803484 237 Coordinates: (181, 677) Eat out percentage: 0.6760774 238 Coordinates: (661, 452) Eat out percentage: 0.77483 239 Coordinates: (906, 654) Eat out percentage: 0.6682643 240 Coordinates: (791, 116) Eat out percentage: 0.4946947 241 Coordinates: (988, 561) Eat out percentage: 0.8663161 242 Coordinates: (312, 580) Eat out percentage: 0.8935117 243 Coordinates: (795, 3) Eat out percentage: 0.3254315 244 Coordinates: (458, 950) Eat out percentage: 0.2387292 245 Coordinates: (768, 933) Eat out percentage: 0.3635655 246 Coordinates: (735, 322) Eat out percentage: 0.3908745 247 Coordinates: (880, 768) Eat out percentage: 0.7230505 248 Coordinates: (939, 237) Eat out percentage: 0.9397836 249 Coordinates: (728, 613) Eat out percentage: 0.3923739 </pre>

Question			Marks
9	1	<p>All marks for AO3 (programming)</p> <p>Marks for changes to the <code>Simulation</code> class:</p> <ol style="list-style-type: none"> Two extra options displayed on the modify company menu using appropriate messages; Selection structures for the new menu options with appropriate condition(s); Gets the user to enter the interest rate when getting a loan and the amount to pay back when paying back under the appropriate circumstances; A. done in appropriate places in the <code>Company</code> class; Calls to appropriate methods in <code>Company</code> class in the selection structures; <p>Marks for changes to the <code>Company</code> class:</p> <ol style="list-style-type: none"> Attributes of appropriate data types created for <code>LoanBalance</code> and <code>InterestRate</code>; Correct calculation of daily interest payment and new balance in <code>ProcessDayEnd</code>; R. if the balance is changed before previous balance concatenated with <code>Details</code> Selection structure to check if <code>LoanBalance</code> is 0 when user chooses to get a loan; A. check for less than or equal to 0 <code>Balance</code>, <code>LoanBalance</code> and <code>InterestRate</code> set to correct values in the selection structure; <code>LoanBalance</code> and <code>Balance</code> changed by the correct amount when user chooses to pay back part of the loan; All attributes in <code>Company</code> are only accessed and modified by methods in <code>Company</code>; R. if no attempt to access or modify the attributes used when getting or paying back a loan. <p>Max 9 marks if code contains errors</p>	10

Question			Marks
9	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 9.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for 9.1 must be sensible.</i></p> <p>Screen capture(s) showing that the balance for AQA Burgers is approximately 92 000; Note for examiners: due to random numbers in simulation exact balance can vary.</p> <div><pre>***** *** Details of all companies: *** ***** Name: AQA Burgers Type of business: fast food Current balance: 92320.17 Average cost per meal: 5 Average price per meal: 10</pre></div>	1

Question			Marks
10	1	<p>All marks for AO3 (programming)</p> <ol style="list-style-type: none"> 1. Created new method called <code>GetOrderedListOfOutlets</code>; R. other names for method I. case and minor typos 2. Method returns a list/array; 3. Outlet 0 is added to the route first; 4. Iterative structure that repeats until all outlets have been added to the route; 5. Has variable that is used to store shortest distance found between two nodes so far and a variable to store which outlet results in the shortest distance; 6. Iterative structure that looks at each outlet for which distance from previous outlet in route needs to be calculated; A. looks at all outlet except previous outlet 7. No outlet can appear more than once in route created; R. if adds or two or fewer outlets to the list only R. if no attempt to check if outlet has already been added or equivalent 8. Route created contains all the company's outlets; 9. Shortest distance between two nodes variable set to suitable starting value and reset after each outlet (except last one) is added to route; 10. <code>GetOrderedListOfOutlets</code> implements the algorithm described in Figure 6 in the question; 11. Modified <code>CalculateDeliveryCost</code> so that it calls <code>GetOrderedListOfOutlets</code> instead of <code>GetListOfOutlets</code>; A. alternative identifier used as long as match that used for mark point 1 <p>Max 10 if code contains errors or if other parts of the subroutine no longer work correctly</p>	11

Question		Marks
10	2	1
<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p>Must match code from 10.1, including prompts on screen capture matching those in code.</p> <p>Code for 10.1 must be sensible.</p> <p>Screen capture(s) showing that the delivery cost for AQA Burgers is 22.10446;</p> <div><pre>***** ***** MENU ***** ***** 1. Display details of households 2. Display details of companies 3. Modify company 4. Add new company 6. Advance to next day Q. Quit Enter your choice: 2 ***** *** Details of all companies: *** ***** Name: AQA Burgers Type of business: fast food Current balance: 86000 Average cost per meal: 5 Average price per meal: 10 Daily costs: 100 Delivery costs: 22.10466 Reputation: 95.4542 Number of outlets: 7 Outlets 1. Coordinates: (200, 203) Capacity: 120 Maximum Capacity: 221 Daily Costs: 200 Visits today: 0 2. Coordinates: (300, 987) Capacity: 120 Maximum Capacity: 195 Daily Costs: 200 Visits today: 0 3. Coordinates: (500, 500) Capacity: 120 Maximum Capacity: 202 Daily Costs: 200 Visits today: 0 4. Coordinates: (305, 303) Capacity: 120 Maximum Capacity: 202 Daily Costs: 200 Visits today: 0 5. Coordinates: (874, 456) Capacity: 120 Maximum Capacity: 201 Daily Costs: 200 Visits today: 0 6. Coordinates: (23, 408) Capacity: 120 Maximum Capacity: 200 Daily Costs: 200 Visits today: 0 7. Coordinates: (412, 318) Capacity: 120 Maximum Capacity: 195 Daily Costs: 200 Visits today: 0 Name: Ben Thor Cuisine</pre></div>		

Question			Marks															
11	1	<p>4 marks for AO3 (design) and 8 marks for AO3 (programming)</p> <p><u>Mark Scheme</u></p> <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 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.</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>	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	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

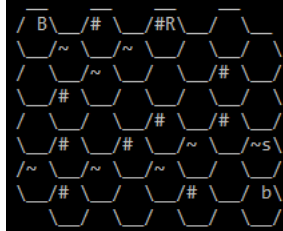
This mark scheme is to be used if solution uses a recursive subroutine to calculate the sum of the digits.

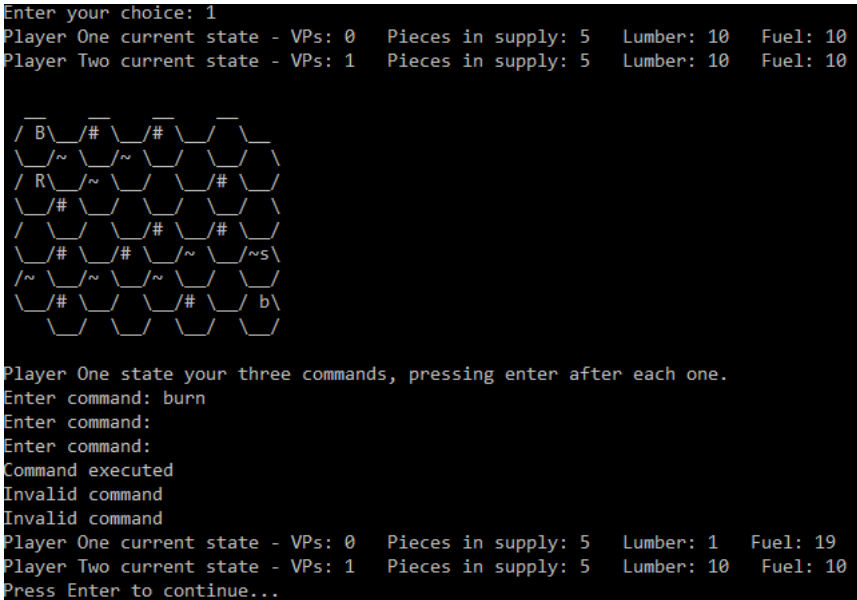
3. Identifying that a recursive subroutine is needed to calculate the sum of the digits.
6. Recursive subroutine has an appropriate base case.
9. Sets the variable used to store the sum of digits to the result returned by the call to the recursive subroutine in an appropriate place.

Max 11 if any errors.

11	2	<div><div>Mark is for AO3 (evaluate)</div><div>**** SCREEN CAPTURE ****</div><div>Must match code from 11.1, including prompts on screen capture matching those in code.</div><div>Code for 11.1 must be sensible.</div><div>Screen capture showing the number 600 being entered and then a message displayed saying 3102</div><div><div>Enter value for n: 600</div><div>3102</div></div></div>	1
----	---	--	---

Question		Marks
12	1	<p>All marks for AO3 (programming)</p> <p>1. Correctly checks if a piece belongs to a player; 2. Correctly checks if a piece is a LESS piece; 3. Correct logic for selection structure for a player’s LESS piece and one added to that player’s victory points if a piece is a LESS piece belonging to that player; 4. Mark points 1 to 3 done for other player; 5. Only adds victory points for LESS pieces if they have not been destroyed;</p> <p>Max 4 if code contains errors</p>
12	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE **** <i>Must match code from 12.1.</i> <i>Code for 12.1 must be sensible.</i></p> <p>Screen captures showing the correct VP totals for both players (2 for player one and 7 for player two);</p> <div><pre>Enter command: Invalid command Invalid command Invalid command Player One current state - VPs: 1 Pieces in supply: 2 Lumber: 5 Fuel: 5 Player Two current state - VPs: 5 Pieces in supply: 2 Lumber: 5 Fuel: 5 Press Enter to continue... /#B\ /#b\ /~1\ \ \ /~ \ /~ \ /~ \ / \ /~ \ /~ \ \ \ / \ /~ \ /~ \ / \ /#1\ /# \ \ \ /#L\ /~ \ /~S\ \ / \ /~ \ /~ \ Player Two state your three commands, pressing enter after each one. Enter command: Enter command: Enter command: Invalid command Invalid command Invalid command Player One current state - VPs: 2 Pieces in supply: 2 Lumber: 5 Fuel: 5 Player Two current state - VPs: 7 Pieces in supply: 2 Lumber: 5 Fuel: 5 Press Enter to continue...</pre></div>

Question		Marks	
13	1	<p>All marks for AO3 (programming)</p> <p>1. Creating a new class called <code>RangerPiece</code>; R. other names for class I. case and minor typos</p> <p>2. New class inherits from <code>Piece</code> and has a constructor that overrides base class constructor with call made to base class constructor; R. if incorrect parameters</p> <p>3. Constructor sets <code>PieceType</code> to "R"; R. if before call to base class constructor R. "r"</p> <p>4. Subroutine called <code>CheckMoveIsValid</code> created that overrides base class method and correct code for normal move R. if incorrect parameters</p> <p>5. Selection structure with correct conditions that allow move from forest terrain to forest terrain;</p> <p>6. Correct fuel cost returned for all moves (forest to forest, distance of one, illegal move, distance of one with peat bog as start terrain, distance of one with peat bog as end terrain);</p> <p>The following relates to the <code>AddPiece</code> subroutine:</p> <p>7. Selection structure with correct condition in appropriate place in code which results in call to constructor for new class;</p> <p>Max 6 if code contains errors</p>	7
13	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE **** <i>Must match code from 13.1.</i> <i>Code for 13.1 must be sensible.</i></p> <p>Screen capture(s) showing that two commands were executed and third command wasn't followed by grid with R piece in 3rd cell on top row;</p> <div><pre>Player One state your three commands, pressing enter after each one. Enter command: move 8 12 Enter command: move 12 2 Enter command: move 2 3 Command executed Command executed That move can't be done Player One current state - VPs: 0 Pieces in supply: 5 Lumber: 10 Fuel: 8 Player Two current state - VPs: 1 Pieces in supply: 5 Lumber: 10 Fuel: 10 Press Enter to continue...</pre></div> <p>Player Two state your three commands, pressing enter after each one. Enter command:</p>	1

Question		Marks
14	1	<p>All marks for AO3 (programming)</p> <p>Marks for changes to the <code>ExecuteCommand</code> method:</p> <ol style="list-style-type: none"> 1. selection structure with correct condition for <code>burn</code> command; 2. selection structures with correct condition to check that there is lumber in the player's supply; 3. returns correct string (A. minor typos, I. case) if player has no lumber; 4. generates a random integer; 5. random integer generated is in correct range; 6. reduces lumber by correct amount; 7. increases fuel by correct amount; <p>Marks for changes to other parts of program:</p> <ol style="list-style-type: none"> 8. Returns <code>True</code> from <code>CheckCommandIsValid</code> if <code>burn</code> command was used; <p>Max 7 marks if code contains errors</p>
14	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE **** <i>Must match code from 14.1.</i> <i>Code for 14.1 must be sensible.</i></p> <p>Screen capture(s) showing that Player One's lumber has decreased by the same amount as their fuel has increased; Notes for examiners: due to random numbers in game exact values can vary; screen capture could show R or S below the B in the top-left corner of the grid; lumber and fuel both had an initial value of 10.</p> 

Question		Marks
15	1	<p>All marks for AO3 (programming)</p> <ol style="list-style-type: none"> Created new method called <code>GetFogOfWar</code>; R. other names for method I. case and minor typos Method returns a Boolean value and takes the index of a tile as a parameter; A. alternatives to passing index of tile eg tile itself I. other parameters Check to see if tile passed as parameter to method contains a piece belonging to the active player; Gets all the neighbours of the tile passed as a parameter to the method; Gets all the neighbours of the tiles identified in mark point 4 // gets all the neighbours of the tiles identified in mark point 4 not already got; Checks at least one neighbouring tile contains a piece belonging to the active player; Iterative structure that looks at each tile identified as being within two of the tile passed to the method; A. not all tiles identified correctly Every time a tile is checked the <code>PieceID</code> in the tile is obtained; Returns a value of <code>False</code> if it correctly identifies, for the tiles checked, that the tile contains a piece belonging to the active player; Method <code>GetFogOfWar</code> returns the correct value under all circumstances; Modified <code>GetPieceTypeInTile</code> so that it calls <code>GetFogOfWar</code>; A. alternative identifier used as long as match that used for mark point 1 <code>GetPieceTypeInTile</code> returns a space character if the value returned by <code>GetFogOfWar</code> is <code>True</code>; <code>GetPieceTypeInTile</code> returns the piece in the tile if there is a piece in the tile and a space character if either there is not a piece in the tile or when the value returned by <code>GetFogOfWar</code> is <code>True</code>; R. if no attempt for either mark points 11 or 12 <p>Alternative answer for mark points 4, 5 and 7</p> <ol style="list-style-type: none"> Iterative structure that is used to check every tile; Gets the distance of each tile from the tile passed as a parameter to the method; Gets all tiles within a distance of two from the tile passed as a parameter to the method; <p>Note: award mark points 4, 5 and 7 (both methods) for solutions where loop could terminate early if value of false is returned due to identification of a tile containing the player's piece that is within distance of two from tile passed as a parameter to the method, before all tiles that need to be checked have been identified.</p> <p>Max 12 if code contains errors or if other parts of the subroutine <code>GetPieceTypeInTile</code> no longer work correctly</p>
15	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 15.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for 15.1 must be sensible.</i></p>

Screen capture(s) showing that for the game from game1.txt the grid is displayed correctly at the start of both player's turns;

```
1. Default game
2. Load game
Q. Quit

Enter your choice: 2
Enter the name of the file to load: game1.txt
Player One current state - VPs: 0   Pieces in supply: 2   Lumber: 5   Fuel: 5
Player Two current state - VPs: 0   Pieces in supply: 2   Lumber: 5   Fuel: 5

  ~B~#b~
  ~L~
  ~
  ~
  ~1~#~
  ~L~~S~

Player One state your three commands, pressing enter after each one.
Enter command:
Enter command:
Enter command:
Invalid command
Invalid command
Invalid command
Player One current state - VPs: 1   Pieces in supply: 2   Lumber: 5   Fuel: 5
Player Two current state - VPs: 5   Pieces in supply: 2   Lumber: 5   Fuel: 5
Press Enter to continue...

  ~B~#b~1~
  ~L~
  ~
  ~
  ~1~#~
  ~L~~S~

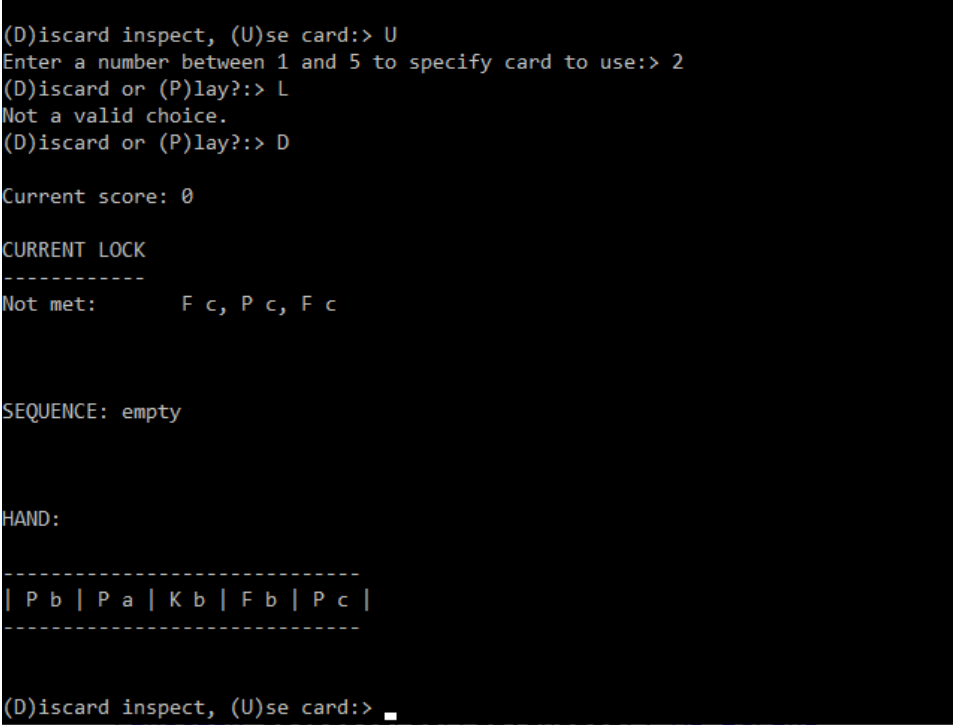
Player Two state your three commands, pressing enter after each one.
Enter command:
```


Question			Marks															
16	1	<p>4 marks for AO3 (design) and 8 marks for AO3 (programming)</p> <p><u>Mark Scheme</u></p> <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>	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 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.	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	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 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.	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																

	<p><u>Guidance</u></p> <p>Evidence of AO3 design – 4 points:</p> <p>Evidence of design to look for in responses:</p> <ol style="list-style-type: none"> 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. <p>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.</p> <p>Evidence for AO3 programming – 8 points:</p> <p>Evidence of programming to look for in response:</p> <ol style="list-style-type: none"> 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. <p>I. additional loop to get program to repeat multiple times.</p> <p>DPT. mark points 7 and 8 if only checks for some vowels or includes at most one non-vowel character.</p> <p>Max 11 if any errors</p>	
--	--	--

Question			Marks
16	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 16.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for 16.1 must be sensible.</i></p> <p>Screen capture showing the string <code>persepolis</code> being entered and then the string <code>pirsopeles</code> being displayed and screen capture showing the string <code>darius</code> being entered and then the string <code>durias</code> being displayed and screen capture showing the string <code>xerxes</code> being entered and then the string <code>xerxes</code> being displayed;</p> <p>I. order of tests</p> <div><div>Enter a string: persepolis</div><div>pirsopeles</div></div> <div><div>Enter a string: darius</div><div>durias</div></div> <div><div>Enter a string: xerxes</div><div>xerxes</div></div>	1

Question			Marks
17		Mark is for AO2 (analyse) 2;	1

Question		Marks
18	1	<p>All marks for AO3 (programming)</p> <ol style="list-style-type: none"> 1. Iterative structure contains code that gets the choice from the player; 2. One correct condition; 3. Both correct conditions and correct logic; 4. Displays error message under all correct circumstances and only under correct circumstances; <p>Max 3 if code contains errors</p> <p>4</p>
18	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 18.1.</i></p> <p><i>Code for 18.1 must be sensible.</i></p> <p>Screen capture showing same message as code for 18.1 displayed when L is entered followed by D being entered and accepted;</p>  <p>Notes for examiners: ignore contents of the hand and the current score. (B)lasting cap might be shown in list of choices or might not be. Output after entering D will be different if a difficulty card was drawn.</p> <p>1</p>

Question		Marks
19	1	<p>All marks for AO3 (programming)</p> <ol style="list-style-type: none"> 1. Creating a new subroutine called <code>GetNumberOfToolCards</code>; R. other identifiers I. case I. minor spelling mistakes 2. New subroutine has mechanism to return an integer and returns an integer value; I. incorrect value A. other numeric data types 3. Iterative structure that repeats a number of times based on the size of <code>Cards</code>; 4. Gets type of card inside iterative structure; 5. Selection structure inside iterative structure that compares (their attempt at) type of card with at least one of P, F or K; 6. Selection structure with correct conditions and value to return incremented by one inside selection structure; <p>The following mark points relate to the <code>PlayGame</code> subroutine:</p> <ol style="list-style-type: none"> 7. Valid calls to <code>GetNumberOfToolCards</code> or <code>GetNumberOfCards</code> and value returned by this call is displayed; A. alternative identifier for subroutine if matches identifier used for mark point 1 8. Appropriate messages displayed along with values returned by calls to <code>GetNumberOfToolCards</code> and <code>GetNumberOfCards</code>; R. if before display of current score R. if after display of player's hand <p>Alternative answer for mark points 5 and 6</p> <ol style="list-style-type: none"> 5. Selection structure inside iterative structure that compares (their attempt at) type of card with <code>Dif</code>; 6. Selection structure with correct condition and value incremented by one inside selection structure, this value is subtracted from the total number of cards before being returned to the calling routine; <p>Alternative answer for mark points 4 and 5</p> <ol style="list-style-type: none"> 4. Gets score for card inside iterative structure; 5. Selection structure inside iterative structure that compares (their attempt at) score for card with zero; <p>Max 7 if code contains errors</p>

Question		Marks
19	2	1
<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 19.1.</i></p> <p><i>Code for 19.1 must be sensible.</i></p> <p>Screen capture(s) showing the values of 33 and 28 followed by the values of 32 and 27; (A. alternative values for the second set of numbers if there is evidence that a difficulty card was drawn from the deck)</p> <pre>Enter L to load a game from a file, anything else to play a new game:> Current score: 0 Cards left in deck: 33 Tool cards left in deck: 28 CURRENT LOCK ----- Not met: K a Not met: K b Not met: K c : : SEQUENCE: empty : HAND: : ----- P c F c P c P c P b ----- (D)iscard inspect, (U)se card:> U Enter a number between 1 and 5 to specify card to use:> 2 (D)iscard or (P)lay?> D Current score: 0 Cards left in deck: 32 Tool cards left in deck: 27 CURRENT LOCK ----- Not met: K a Not met: K b Not met: K c : : SEQUENCE: empty : HAND: : ----- P c P c P c P b P a ----- (D)iscard inspect, (U)se card:> _</pre>		

Question		Marks
20	1	<p>All marks for AO3 (programming)</p> <p>1. Create a variable, with an appropriate name and data type, to use to keep track if there is a blasting cap available and give variable a value of <code>True</code> (or equivalent) // create a variable, with an appropriate name and data type, to use to keep track if blasting cap has been used and give variable a value of <code>False</code> (or equivalent); R. if inside iterative structure that repeats until game is over</p> <p>2. Selection structure, in appropriate place that checks for the player's choice being <code>B</code> (or suitable alternative) and appropriate modified message in <code>GetChoice</code> subroutine;</p> <p>3. Selection structure that checks if the player has a blasting cap (or does not have a blasting cap);</p> <p>4. If there is a blasting cap (A. incorrect condition) gets the player's choice of challenge; R. if value is not of integer data type, unless it is converted to be an integer before it is used</p> <p>5. If they chose to use a blasting cap (A. incorrect condition) sets the value of variable used to indicate if there is a blasting cap to <code>False</code> (or equivalent);</p> <p>6. Selection structure that checks player's choice of challenge is less than or equal to the number of challenges;</p> <p>7. Selection structure that checks player's choice of challenge has not already been met; R. if checks the wrong challenge</p> <p>8. If conditions for both mark points 6 and 7 are met displays message saying blasting cap has been used; I. incorrect logic for selection structure(s)</p> <p>9. Changes the met status of the challenge specified by the player inside selection structure(s) for mark points 6 and 7; I. incorrect logic for selection structure(s) R. if changes the wrong challenge</p> <p>Max 8 marks if code contains errors</p>
20	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 20.1.</i></p> <p><i>Code for 20.1 must be sensible.</i></p> <p>Screen capture(s) showing that the third condition is met after use of blasting cap and that use of a second blasting cap is not permitted;</p> <p>Notes for examiners: ignore messages about number of cards in deck and number of tool cards in deck.</p>

		<pre>(D)iscard inspect, (B)lasting cap, (U)se card:> B Challenge to blast? 3 Blasting cap used. CURRENT LOCK ----- Not met: P a, F a, P a Challenge met: K b Challenge met: P c, F b, P a Challenge met: K a Current score: 8 Cards left in deck: 30 Tool cards left in deck: 25 CURRENT LOCK ----- Not met: P a, F a, P a Challenge met: K b Challenge met: P c, F b, P a Challenge met: K a SEQUENCE: ----- K b F c K a ----- HAND: ----- P a K a P b F a P a ----- (D)iscard inspect, (B)lasting cap, (U)se card:> B Current score: 8 Cards left in deck: 30 Tool cards left in deck: 25 CURRENT LOCK ----- Not met: P a, F a, P a Challenge met: K b Challenge met: P c, F b, P a Challenge met: K a SEQUENCE: ----- K b F c K a ----- HAND: ----- P a K a P b F a P a ----- (D)iscard inspect, (B)lasting cap, (U)se card:> _</pre>	
--	--	---	--

Question		Marks
21	1	<p>All marks for AO3 (programming)</p> <p>Mark points 1 to 9 relate to the <code>TrapCard</code> class.</p> <ol style="list-style-type: none"> 1. Creating a new class called <code>TrapCard</code> that inherits from <code>DifficultyCard</code>; R. other names for class I. case and minor typos 2. Constructor calls parent class constructor and then sets <code>Type</code> to <code>Trp</code> // Constructor sets <code>Type</code> to <code>Trp</code> and sets <code>CardNumber</code> to value of parameter; 3. Subroutine called <code>Process</code> created that overrides parent class method and contains a call to parent class method (or equivalent); 4. Iteration structure that repeats based on the number of challenges on the current lock; 5. Selection structure, inside iteration structure, that compares the value of a challenge's status with either <code>True</code> or <code>False</code>; 6. Adds challenge / index of challenge to a list if challenge has been met // after ascertaining that at least one challenge has been met repeatedly selects a random challenge; 7. Selects a random challenge that has been met; R. if could select a challenge that has not been met under some circumstances 8. Changes the status of the selected challenge to not met (<code>False</code>); R. if multiple challenges changed 9. If no challenges have been met then a call is made to the parent class method (or equivalent); R. if could also set a met challenge's status to <code>False</code> 10. Modified <code>GetCardFromDeck</code> so that trap cards are processed in the same way as difficulty cards; R. other messages I. case and minor typos 11. Modified <code>GetCardFromDeck</code> so it displays the message <code>Trap!</code> if a trap card is drawn; R. other messages R. if message displayed when non-trap card is drawn I. case and minor typos 12. Modified <code>SetupCardCollectionFromGameFile</code> so that it creates trap cards instead of difficulty cards; <p>Max 11 if code contains errors or if other parts of the subroutines <code>GetCardFromDeck</code> or <code>SetupCardCollectionFromGameFile</code> no longer work correctly</p>
21	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 21.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for 21.1 must be sensible.</i></p> <p>Screen capture(s) showing that for the game from <code>game1.txt</code> one of the two challenges that was met is now shown as not met;</p>

		<pre>CURRENT LOCK ----- Not met: P a, F a, P a Challenge met: K b Not met: P c, F b, P a Challenge met: K a SEQUENCE: ----- K b F c K a P a ----- HAND: ----- K a P b F a P a P a ----- (D)iscard inspect, (B)lasting cap, (U)se card:> U Enter a number between 1 and 5 to specify card to use:> 1 (D)iscard or (P)lay?:> P Trap! Difficulty encountered! HAND: ----- P b F a P a P a ----- To deal with this you need to either lose a key (enter 1-5 to specify position o f key) or (D)iscard five cards from the deck:> D Current score: 12 Cards left in deck: 27 Tool cards left in deck: 23 CURRENT LOCK ----- Not met: P a, F a, P a Not met: K b Not met: P c, F b, P a Challenge met: K a SEQUENCE: ----- K b F c K a P a K a ----- HAND: ----- P b F a P a P a P a ----- (D)iscard inspect, (B)lasting cap, (U)se card:> _</pre>	
--	--	--	--

Question			Marks															
22	1	<p>4 marks for AO3 (design) and 8 marks for AO3 (programming)</p> <p><u>Mark Scheme</u></p> <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 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.</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>	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	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																

	<p><u>Guidance</u></p> <p>Evidence of AO3 design – 4 points:</p> <p>Evidence of design to look for in responses:</p> <ol style="list-style-type: none"> 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) <p>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.</p> <p>Evidence for AO3 programming – 8 points:</p> <p>Evidence of programming to look for in response:</p> <ol style="list-style-type: none"> 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. <p>Max 11 if any errors</p>	
--	---	--

Question		Marks
22	2	1
<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 22.1, including prompts on screen capture(s) matching those in code.</i></p> <p><i>Code for 22.1 must be sensible.</i></p> <p>Screen captures showing the string(s) entered and result(s) of each of the tests;</p> <p>I. order of tests</p> <p>A. tests done individually or done as one extended test</p> <p>Enter a string: BOIL not valid</p> <p>Enter a string: BRAisE not valid</p> <p>Enter a string: ROAST not valid</p> <p>Enter a string: BLANCH valid</p> <p>Enter a string: PRESSURECOOK not valid</p> <p>Enter a string: </p> <p>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.</p>		

Question			Marks
23	1	<p>All marks for AO3 (programming)</p> <ol style="list-style-type: none"> 1. Iterative structure contains code that gets the choice from the player; 2. One correct condition; 3. Both correct conditions and correct logic; 4. Displays error message under all correct circumstances and only under correct circumstances; R. message same as original prompt <p>Max 3 if code contains errors</p>	4
23	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 23.1.</i></p> <p><i>Code for 23.1 must be sensible.</i></p> <p>Screen capture showing message displayed when 6 is entered followed by 4 being entered and accepted;</p>	1

```

      1  2  3  4  5  6
-----
1 |  |  | K1 |  |  |
2 |  | ! | ! | ! |  |
3 |  |  |  |  |  |
4 |  |  |  |  |  |
5 |  | " | " | " | " |
6 |  |  | k2 |  |  |
-----

Move option offer: jazair

Player One
Score: 100
Move option queue: 1. ryott   2. chowkidar   3. cuirassier   4. faujdar   5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer: 9
Choose the move option from your queue to replace (1 to 5): 6
Error - try again.
Choose the move option from your queue to replace (1 to 5): 4

      1  2  3  4  5  6
-----
1 |  |  | K1 |  |  |
2 |  | ! | ! | ! |  |
3 |  |  |  |  |  |
4 |  |  |  |  |  |
5 |  | " | " | " | " |
6 |  |  | k2 |  |  |
-----

Move option offer: jazair

Player One
Score: 98
Move option queue: 1. ryott   2. chowkidar   3. cuirassier   4. jazair   5. jazair

Turn: Player One

Choose move option to use from queue (1 to 3) or 9 to take the offer:
```

Note for examiners: Bhukampa might be shown in list of choices or might not be.

Question			Marks
24	1	<p>All marks for AO3 (programming)</p> <p>Mark points 1 to 6 refer to the new method <code>ProcessBhukampa</code>; mark points 7 to 9 refer to <code>PlayGame</code>.</p> <ol style="list-style-type: none"> 1. Create a new method called <code>ProcessBhukampa</code>; R. other names for method; I. case and minor typos 2. Generates two random numbers; 3. Correct range for random numbers generated (0 to 35); A. 1 to 6 if generating random row/column position instead of position in <code>Board</code> if these are then used to create two valid square references and will be able to generate the full range of valid square references A. 1 to 6 if generating random row/column position instead of position in <code>Board</code> if these are then used to create an index between 0 and 35 4. Repeats until the two random numbers are different; 5. Swaps positions of two squares in <code>Board</code> list/array; R. only swapping the pieces that are in the two squares 6. Repeats attempt at (any of) their code for mark points 2 to 5 five times; 7. Call to new method from <code>PlayGame</code>; R. if not in iterative structure that gets move option from user 8. Selection structure in <code>PlayGame</code> with correct condition (= 8, or equivalent) 9. When <code>bhukampa</code> is chosen, player's score is decreased by 15 and call made to <code>DisplayState</code>; R. if (sometimes) executes when <code>bhukampa</code> not chosen <p>Max 8 marks if code contains errors</p>	9
24	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE **** <i>Must match code from 24.1.</i> <i>Code for 24.1 must be sensible.</i></p> <p>Screen capture(s) showing option 8 being selected, player one score of 85 and new board state; A. score of 70 or 55</p> <p>Notes for examiners: new board state is (partly) random so will not exactly match the one shown in this mark scheme.</p>	1

		<pre> Move option offer: jazair Player One Score: 100 Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair Turn: Player One Choose move option to use from queue (1 to 3) or 9 to take the offer or 8 for a bhukampa: 8 1 2 3 4 5 6 ----- 1 k1 2 " 3 k2 4 5 " 6 ----- Move option offer: jazair Player One Score: 85 Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair Turn: Player One Choose move option to use from queue (1 to 3) or 9 to take the offer or 8 for a bhukampa: </pre>	
Question			Marks
25	1	<p>All marks for AO3 (programming)</p> <p>Mark points 1 to 6 relate to the Gacaka class.</p> <ol style="list-style-type: none"> 1. Creating a new class called Gacaka that inherits from Square; R. other names for class; I. case and minor typos 2. Method called SetPiece/GetPointsForOccupancy created that overrides parent class method; 3. Message "Trap!" displayed in SetPiece method and piece is added to Gacaka; R. other messages I. case and minor typos A. added in appropriate place in PlayGame 4. Value of PointsIfCaptured for piece in Gacaka is increased by 2; A. making PointsIfCaptured public // new piece added to Gacaka which is same as original piece except PointsIfCaptured is two higher 5. Value of 0 returned by GetPointsForOccupancy if there is no piece in the gacaka; R. if always returns a value of 0 6. Value of -3 returned by GetPointsForOccupancy if there is a piece in the gacaka; A. if only returned for that player's turn or on both player's turn R. if always returns a value of -3 <p>Mark points 7 to 8 relate to the CreateBoard method.</p> <ol style="list-style-type: none"> 7. An object of type Gacaka is created; 8. The Gacaka object is added to the correct position in the Board list; R. if there are not exactly 36 objects in the Board list; <p>Max 7 if code contains errors (including not checking who the piece belongs to)</p>	8

Question		Marks
25	2	1
<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 25.1.</i></p> <p><i>Code for 25.1 must be sensible.</i></p> <p>Screen capture(s) showing the correct final board state, the two player's scores and the message <code>Trap!</code> being displayed after the 2nd player's move; A. alternative messages if they match 25.1</p> <pre>Move option queue: 1. ryott 2. chowkidar 3. cuirassier 4. faujdar 5. jazair Turn: Player One Choose move option to use from queue (1 to 3) or 9 to take the offer or 8 for a bhukampa: 3 Enter the square containing the piece to move (row number followed by column number): 24 Enter the square to move to (row number followed by column number): 44 Trap! New score: 95 1 2 3 4 5 6 ----- 1 K1 2 ! ! ! 3 4 ! 5 " " " " 6 k2 ----- Move option offer: jazair Player Two Score: 100 Move option queue: 1. ryott 2. chowkidar 3. jazair 4. faujdar 5. cuirassier Turn: Player Two - Choose move option to use from queue (1 to 3) or 9 to take the offer or 8 for a bhukampa: 1 Enter the square containing the piece to move (row number followed by column number): 54 Enter the square to move to (row number followed by column number): 44 Trap! New score: 104 1 2 3 4 5 6 ----- 1 K1 2 ! ! ! 3 4 " 5 " " " " 6 k2 ----- Move option offer: jazair Player One Score: 95 Move option queue: 1. ryott 2. chowkidar 3. faujdar 4. jazair 5. cuirassier Turn: Player One Choose move option to use from queue (1 to 3) or 9 to take the offer or 8 for a bhukampa: </pre> <p>Note for examiners: Bhukampa might be shown in list of choices or might not be.</p>		

Question		Marks
26	1	<p>All marks for AO3 (programming)</p> <p>Mark points 1 to 11 relate to the <code>GetNoOfPossibleMoves</code> method.</p> <ol style="list-style-type: none"> 1. Creating a new method called <code>GetNoOfPossibleMoves</code> that takes <code>Board /</code> a list of squares as a parameter R. other method identifiers I. case and minor typos; 2. Iteration structure that repeats number of times based on size of board list; 3. Nested iteration structures that repeat correct number of times to check every combination of start and finish squares // nested iteration structures that repeat correct number of times to look at every combination of start square and legal move option. 4. Iteration structures that when combined will repeat enough times to check every combination of move option with the squares from their code for mark points 2 and 3; 5. Calculate the square reference for the start square; 6. Calculate the square reference for the finish square; 7. Calls the <code>CheckPlayerMove / CheckIfThereIsAMoveToSquare</code> method; A. suitable alternatives to calling method e.g. rewriting code from method 8. Checks if there is one of the player's pieces in the start square; 9. Checks if there is one of the opponent's pieces in the finish square and checks if the finish square does not contain a piece; 10. Adds one to the count of possible moves when (some) legal moves are found; <p>Note for examiners: maximum of 1 mark for mark points 8 and 9 if the program would attempt to use a method/property for a piece in an empty square. <code>CheckSquareIsValid</code> completes all checks needed for mark points 8 and 9 (if called twice) but is not easily accessible from the <code>Player</code> class.</p> <p>Note for examiners: mark points 7 to 9 do not have to be inside iterative structures to be awarded</p> <ol style="list-style-type: none"> 11. Call to <code>GetNoOfPossibleMoves</code> in appropriate place in <code>DisplayState</code> method; A. added to <code>DisplayBoard</code> instead of <code>DisplayState</code> 12. Value returned by <code>GetNoOfPossibleMoves</code> is displayed <p>Max 11 if code contains any errors</p>

26	2	<div><div>Mark is for AO3 (evaluate)</div><div>**** SCREEN CAPTURE ****</div><div>Must match code from 26.1, including prompts on screen capture matching those in code.</div><div>Code for 26.1 must be sensible.</div><div>Screen capture(s) showing that there are 52 legal moves for player two;</div><div></div><div>Note for examiners: Bhukampa might be shown in list of choices or might not be.</div></div>	1
----	---	--	---

Question			Marks															
27	1	<p>4 marks for AO3 (design) and 8 marks for AO3 (programming)</p> <p><u>Mark Scheme</u></p> <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 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.</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>	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	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

27	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 27.1, including messages on screen capture(s) matching those in code.</i></p> <p><i>Code for 27.1 must be sensible.</i></p> <p>Screen captures showing the integer(s) entered and result(s) of each of the tests; I. order of tests</p> <pre>Enter a number: -3 Enter a number: 14982 perfectly bouncy >>> Enter a number: 1234 not bouncy >>> </pre> <p>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.</p>	1
----	---	--	---

Question		Marks
28	1	<div><div>All marks for AO3 (programming)</div><div><div><div>1. Selection structure with correct condition for one boundary;</div><div>2. Second correct condition and correct connecting logic; A. second selection structure</div><div>3. Valid set to true if the conditions for allowed values are met; A. any equivalent method that would ensure that iteration takes place in these circumstances</div><div>4. Selection structure(s) added in correct location in code;</div></div><div><div>Alternative answer</div><div><div><div>1. Iteration structure modified with correct condition for one boundary;</div><div>2. Iteration structure modified with second correct condition;</div><div>3. Correct connecting logic for three conditions;</div><div>4. Column initialised to value that ensures code in loop executed at least once;</div></div></div><div><div>Max 3 if code contains errors</div><div>Max 3 if used value 8 instead of GridSize</div></div></div></div></div>
28	2	<div><div>Mark is for AO3 (evaluate)</div><div><div>**** SCREEN CAPTURE ****</div><div>Must match code from 28.1.</div><div>Code for 28.1 must be sensible.</div></div><div><div>Screen capture showing 1 is entered followed by 10 then 4 when asked to enter column number again, with 4 being accepted:</div><div><div>Press Enter to start a standard puzzle or enter name of file to load:</div><div><div>1 2 3 4 5 6 7 8</div><div>-----</div><div>8 - - - - - - </div><div>-----</div><div>7 @ - - - - - </div><div>-----</div><div>6 - - - @ - - </div><div>-----</div><div>5 - - - - - @ </div><div>-----</div><div>4 - @ @ - - - </div><div>-----</div><div>3 - - - - - - </div><div>-----</div><div>2 - - @ - - - </div><div>-----</div><div>1 - - - - - - </div><div>-----</div><div>Current score: 0</div><div>Enter row number: 1</div><div>Enter column number: 10</div><div>Enter column number: 4</div><div>Enter symbol:</div></div></div></div></div>

Question			Marks
29	1	<p>All marks for AO3 (programming)</p> <ol style="list-style-type: none"> 1. Create variables for average/total and highest, set to appropriate start values // creates a list to store all the scores for completed puzzles; R. if not in appropriate location A. equivalent variables 2. Compare final score for puzzle to highest score so far; 3. Change highest score found so far if condition for selection structure is met; A. incorrect condition 4. Calculate new average; A. calculate new total as long as there is an attempt to calculate average later in the code. 5. Display average and highest scores after iteration structure that checks if the user wants to do another puzzle; A. inside iterative structure if also inside a selection structure that checks that the user does not want to do another puzzle <p>Max 4 marks if code contains errors</p>	5
29	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 29.1.</i></p> <p><i>Code for 29.1 must be sensible.</i></p>	1

```
Press Enter to start a standard puzzle or enter name of file to load: puzzle1
```

```
  1 2 3 4 5
  -----
5 |Q|Q|@|-|-|
  -----
4 |Q|Q|-|-|-|
  -----
3 |-|X|Q|X|-|
  -----
2 |-|-|X|-|-|
  -----
1 |-|X|-|-|-|
  -----
```

```
Current score: 10
```

```
Enter row number: 5
```

```
Enter column number: 5
```

```
Enter symbol: X
```

```
  1 2 3 4 5
  -----
5 |Q|Q|@|-|X|
  -----
4 |Q|Q|-|-|-|
  -----
3 |-|X|Q|X|-|
  -----
2 |-|-|X|-|-|
  -----
1 |-|X|-|-|-|
  -----
```

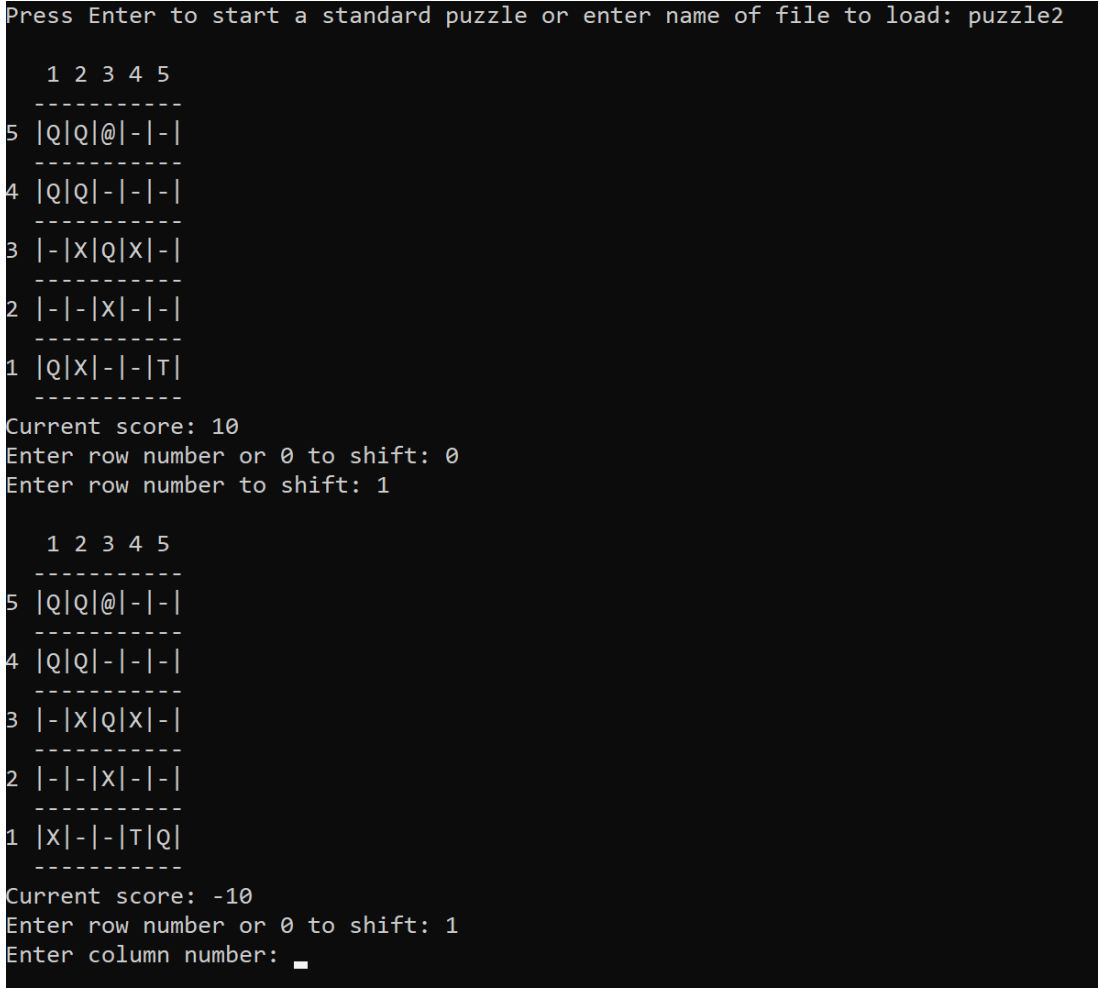
```
Puzzle finished. Your score was: 10
```

```
Do another puzzle? Y
```

```
Press Enter to start a standard puzzle or enter name of file to load: puzzle1
```

	<pre> 1 2 3 4 5 ----- 5 Q Q @ - - ----- 4 Q Q - - - ----- 3 - X Q X - ----- 2 - - X - - ----- 1 - X - - - ----- Current score: 10 Enter row number: 1 Enter column number: 4 Enter symbol: X 1 2 3 4 5 ----- 5 Q Q @ - - ----- 4 Q Q - - - ----- 3 - X Q X - ----- 2 - - X - - ----- 1 - X - X - ----- Puzzle finished. Your score was: 20 Do another puzzle? N High score: 20 Average score: 15 </pre>	
	Screen capture(s) showing puzzle1 was used followed by correct high score and average score;	

Question		Marks
30	1	<p>All marks for AO3 (programming)</p> <p>Mark points 1 to 6 refer to the new method <code>ShiftCellsInRowLeft</code>; mark points 7 to 11 refer to <code>AttemptPuzzle</code>.</p> <ol style="list-style-type: none"> 1. Create a new method called <code>ShiftCellsInRowLeft</code> with an integer parameter; R. other names for method; I. case and minor typos 2. Calculate the index of one cell in the specified row eg for the first cell in a row $(GridSize - Row) * GridSize$; A. correct use of <code>GetCell</code> 3. Store a cell in a temporary variable; 4. Iteration structure that repeats based on number of cells in a row (must result in attempting to inspect all but one cell in a row or attempting to inspect all cells in a row); 5. Move one cell in <code>Grid</code> one place to the left; 6. Moves the leftmost cell in the row to the end of the row; 7. Modified message about entering 0 to shift and selection structure that checks if 0 entered; R. if not in iterative structure that gets row from user 8. Gets the row to shift from the user; 9. Call to new method from <code>AttemptPuzzle</code> with correct parameter value; 10. Decrease score by 20; R. if not in selection structure for shifting cells option 11. Display new grid and new current score; R. if before score decreased <p>Max 2 marks for mark points 2, 4, 6 if actual numeric value used instead of <code>GridSize</code></p> <p>Max 1 mark for mark points 5, 6 if any cell would be lost during the shifting process</p> <p>Max 10 marks if code contains errors</p>

30	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 30.1.</i></p> <p><i>Code for 30.1 must be sensible.</i></p>  <p>Screen capture(s) showing the correct final grid state and score for the user;</p>	1
30	3	<p>Mark is for AO2 (apply)</p> <p>$(-1, 0);$</p>	1
30	4	<p>Mark is for AO2 (apply)</p> <p>$(GridSize - 1, 0);$</p>	1

Question		Marks
31	1	<p>All marks for AO3 (programming)</p> <p>Mark points 1 to 7 relate to the <code>CountdownCell</code> class; mark points 8 to 13 relate to the <code>AttemptPuzzle</code> method.</p> <ol style="list-style-type: none"> 1. Creating a new class called <code>CountdownCell</code>; R. other method identifiers I. case and minor typos 2. <code>CountdownCell</code> inherits from <code>BlockedCell</code> and has a constructor; 3. Constructor gives appropriate initial values for timer attribute and <code>Symbol</code>; A. Constructor gives appropriate initial values for timer attribute only, if <code>GetSymbol</code> overridden A. using <code>Symbol</code> as the timer if there is evidence of the numeric value in <code>Symbol</code> being decremented 4. Overrides <code>UpdateCell</code> method; 5. Decrease value of timer by 1 and update <code>Symbol</code> to match; 6. Selection structure that checks if value of timer is now 0 and if so changes <code>Symbol</code> to @; A. overriding <code>GetSymbol</code> to return @ when timer is 0 7. <code>Symbol</code> doesn't change <u>again</u> after becoming @ (eg if timer decreases to -1 it will stay the same); R. if inside selection structure that checks if timer value is equal to 0, unless other code prevents symbol changing on further iterations R. if no attempt to update the symbol with the timer values/use the symbol as a timer 8. Indefinite iteration structure that contains attempt to find an empty cell; 9. Generate random number in correct range; R. if range is based on a specific size for the grid 10. Checks if cell at random number position is empty; 11. Replaces cell in selected position with a <code>CountdownCell</code>; R. if more than one cell changed 12. Iteration structure that repeats a number of times equal to the number of cells in <code>Grid</code>; R. if only works with one size of <code>Grid</code> 13. Call to <code>UpdateCell</code> selected cell; R. if not in iteration structure for mark point 12 <p>Max 12 if code contains any errors</p>

31	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 31.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for 31.1 must be sensible.</i></p> <pre> Press Enter to start a standard puzzle or enter name of file to load: puzzle3 1 2 3 4 5 ----- 5 Q Q @ - - ----- 4 Q Q - - - ----- 3 - X Q X - ----- 2 - - X - - ----- 1 Q X - - T ----- Current score: 10 Enter row number or 0 to shift: 1 Enter column number: 4 Enter symbol: X 1 2 3 4 5 ----- 5 Q Q @ - - ----- 4 Q Q - - - ----- 3 3 X Q X - ----- 2 - - X - - ----- 1 Q X - X T ----- Current score: 20 Enter row number or 0 to shift: 5 Enter column number: 5 Enter symbol: T 1 2 3 4 5 ----- 5 Q Q @ - T ----- 4 Q Q - - - ----- 3 2 X Q X - ----- 2 - - X - - ----- 1 Q X - X T ----- Current score: 20 Enter row number or 0 to shift: 4 Enter column number: 5 Enter symbol: T </pre>	1
----	---	---	---

	<div><pre> 1 2 3 4 5 ----- 5 Q Q @ - T ----- 4 Q Q - - T ----- 3 1 X Q X - ----- 2 - - X - - ----- 1 Q X - X T ----- Current score: 20 Enter row number or 0 to shift: 3 Enter column number: 5 Enter symbol: T 1 2 3 4 5 ----- 5 Q Q @ - T ----- 4 Q Q - - T ----- 3 @ X Q X T ----- 2 - - X - - ----- 1 Q X - X T ----- Current score: 20 Enter row number or 0 to shift: _ </pre></div>	
	<div><p>Screen capture(s) showing that a number 3 appeared in a cell, changed to 2, then 1, then @;</p><p>Note for examiners: the location of the cell that shows a 3 (then 2, then 1, then @) and the three cells containing Ts (apart from row 1, column 5) are likely to be different to those shown here.</p></div>	