



A-level COMPUTER SCIENCE 7517/1

Paper 1

Mark scheme

June 2020

Version: 1.0 Final



206A7517/1/MS

Mark schemes are prepared by the Lead Assessment Writer and considered, together with the relevant questions, by a panel of subject teachers. This mark scheme includes any amendments made at the standardisation events which all associates participate in and is the scheme which was used by them in this examination. The standardisation process ensures that the mark scheme covers the students' responses to questions and that every associate understands and applies it in the same correct way. As preparation for standardisation each associate analyses a number of students' scripts. Alternative answers not already covered by the mark scheme are discussed and legislated for. If, after the standardisation process, associates encounter unusual answers which have not been raised they are required to refer these to the Lead Examiner.

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of students' reactions to a particular paper. Assumptions about future mark schemes on the basis of one year's document should be avoided; whilst the guiding principles of assessment remain constant, details will change, depending on the content of a particular examination paper.

Further copies of this mark scheme are available from aqa.org.uk

Copyright information

AQA retains the copyright on all its publications. However, registered schools/colleges for AQA are permitted to copy material from this booklet for their own internal use, with the following important exception: AQA cannot give permission to schools/colleges to photocopy any material that is acknowledged to a third party even for internal use within the centre.

Copyright © 2020 AQA and its licensors. All rights reserved.

Level of response marking instructions

Level of response mark schemes are broken down into levels, each of which has a descriptor. The descriptor for the level shows the average performance for the level. There are marks in each level.

Before you apply the mark scheme to a student's answer read through the answer and annotate it (as instructed) to show the qualities that are being looked for. You can then apply the mark scheme.

Step 1 Determine a level

Start at the lowest level of the mark scheme and use it as a ladder to see whether the answer meets the descriptor for that level. The descriptor for the level indicates the different qualities that might be seen in the student's answer for that level. If it meets the lowest level then go to the next one and decide if it meets this level, and so on, until you have a match between the level descriptor and the answer. With practice and familiarity you will find that for better answers you will be able to quickly skip through the lower levels of the mark scheme.

When assigning a level you should look at the overall quality of the answer and not look to pick holes in small and specific parts of the answer where the student has not performed quite as well as the rest. If the answer covers different aspects of different levels of the mark scheme you should use a best fit approach for defining the level and then use the variability of the response to help decide the mark within the level, ie if the response is predominantly level 3 with a small amount of level 4 material it would be placed in level 3 but be awarded a mark near the top of the level because of the level 4 content.

Step 2 Determine a mark

Once you have assigned a level you need to decide on the mark. The descriptors on how to allocate marks can help with this. The exemplar materials used during standardisation will help. There will be an answer in the standardising materials which will correspond with each level of the mark scheme. This answer will have been awarded a mark by the Lead Examiner. You can compare the student's answer with the example to determine if it is the same standard, better or worse than the example. You can then use this to allocate a mark for the answer based on the Lead Examiner's mark on the example.

You may well need to read back through the answer as you apply the mark scheme to clarify points and assure yourself that the level and the mark are appropriate.

Indicative content in the mark scheme is provided as a guide for examiners. It is not intended to be exhaustive and you must credit other valid points. Students do not have to cover all of the points mentioned in the Indicative content to reach the highest level of the mark scheme.

An answer which contains nothing of relevance to the question must be awarded no marks.

A-level Computer Science

Paper 1 (7517/1) – applicable to all programming languages A, B, C, D and E

June 2020

The following annotation is used in the mark scheme:

- ;** - means a single mark
- //** - means an alternative response
- /** - means an alternative word or sub-phrase
- A.** - means an acceptable creditworthy answer
- R.** - means reject answer as not creditworthy
- NE.** - means not enough
- I.** - means ignore
- DPT.** - means "Don't penalise twice". In some questions a specific error made by a candidate, if repeated, could result in the loss of more than one mark. The **DPT** label indicates that this mistake should only result in a candidate losing one mark, on the first occasion that the error is made. Provided that the answer remains understandable, subsequent marks should be awarded as if the error was not being repeated.

Examiners are required to assign each of the candidate's responses to the most appropriate level according to **its overall quality**, and then allocate a single mark within the level. When deciding upon a mark in a level examiners should bear in mind the relative weightings of the assessment objectives

eg

In question **05.1**, the marks available for the AO3 elements are as follows:

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

Where a candidate's answer only reflects one element of the AO, the maximum mark they can receive will be restricted accordingly.

Question			Marks
01	1	Mark is for AO2 (apply) 4 // $\sqrt{4^2+0^2};$	1
01	2	All marks AO2 (apply) 2 marks: 16 If final answer is incorrect then award a maximum of 1 mark for working: 1 mark: for either multiplying 4 by 4 or multiplying 3 by 0 1 mark: for adding together the sum of two (incorrect) products	2
01	3	All marks AO2 (analyse) The angle will still be the same; A. the direction will not change The magnitude will be doubled // the magnitude will now be 10;	2
01	4	All marks AO2 (analyse) The angle will be $180 - c$ // the angle will be $360 - 180 - c$ // the angle will be 143.13; A. the direction of a will be the opposite of its current direction The magnitude will still be the same;	2

Question			Marks										
02	1	All marks AO1 (understanding)	2										
		<table><tr><th>Statement</th><th>True or False?</th></tr><tr><td>All regular languages can be represented using a finite state machine without outputs.</td><td>True</td></tr><tr><td>The set of strings defined by a regular language is always finite in size.</td><td>False</td></tr><tr><td>There are some languages which can be represented in Backus-Naur Form (BNF) that are not regular languages.</td><td>True</td></tr></table>		Statement	True or False?	All regular languages can be represented using a finite state machine without outputs.	True	The set of strings defined by a regular language is always finite in size.	False	There are some languages which can be represented in Backus-Naur Form (BNF) that are not regular languages.	True		
		Statement		True or False?									
		All regular languages can be represented using a finite state machine without outputs.		True									
		The set of strings defined by a regular language is always finite in size.		False									
		There are some languages which can be represented in Backus-Naur Form (BNF) that are not regular languages.		True									
Mark as follows:													
1 mark: two rows correct													
1 mark: all three rows correct													
02	2	Mark is for AO2 (analyse)	1										
		<pre><sentence> ::= <np><v> <v><np> // <sentence> ::= <v><np> <np><v></pre> R. any answers that consist of more than one rule											
02	3	Mark is for AO2 (apply)	1										
		<table><tr><th>String</th><th>Valid sentence (Y/N)?</th></tr><tr><td>cuddle the cat</td><td>Y</td></tr><tr><td>drank a human</td><td>Y</td></tr><tr><td>the cat slept</td><td>Y</td></tr><tr><td>cat or dog</td><td>N</td></tr></table>		String	Valid sentence (Y/N)?	cuddle the cat	Y	drank a human	Y	the cat slept	Y	cat or dog	N
		String		Valid sentence (Y/N)?									
		cuddle the cat		Y									
		drank a human		Y									
		the cat slept		Y									
cat or dog	N												

Question			Marks
02	4	Mark is for AO2 (apply) Modify the existing rule for np: $\langle np \rangle ::= \langle d \rangle \langle n \rangle \mid \langle n \rangle$ // Modify an existing rule for sentence: $\langle sentence \rangle ::= \langle np \rangle \langle v \rangle \mid \langle n \rangle \langle v \rangle$ // Modify an existing rule for sentence: $\langle sentence \rangle ::= \langle v \rangle \langle np \rangle \mid \langle n \rangle \langle v \rangle$ // Modify an existing rule for sentence: $\langle sentence \rangle ::= \langle np \rangle \langle v \rangle \mid \langle v \rangle \langle np \rangle \mid \langle n \rangle \langle v \rangle$ // Create a new rule: $\langle sentence \rangle ::= \langle n \rangle \langle v \rangle$ // Create a new rule: $\langle np \rangle ::= \langle n \rangle$	1
02	5	All marks for AO2 (apply) Mark as follows: 2 marks: $8 \times 4 \times 3 \times 8 \times 4$ // $2 \times 4 \times 4 \times 3 \times 2 \times 4 \times 4$ // 3072 If final answer is incorrect then award a maximum of 1 mark for working: 1 mark: for calculating that there are 8 noun phrases // for calculating that there are 4×2 noun phrases 1 mark: for multiplying an incorrectly calculated number of noun phrases by the number of noun phrases, by 3, by 4 and by 4 again	2
02	6	Mark is for AO2 (apply) Infinitely more;	1

Question			Marks
03	1	Mark is for AO2 (apply) 4! // 4x3x2x1 // 4x3x2 // 24;	1
03	2	Mark is for AO2 (apply) n! // factorial of n; A. 1 * 2 * ... * n-1 * n (or similar)	1
03	3	Mark is for AO2 (analyse) The string could contain more than one occurrence of a character; Each athlete is unique, each character is not (guaranteed to be) unique; There are n characters in the string but not n distinct characters; Some of the anagrams could be duplicates; Max 1	1
03	4	All marks for AO1 (knowledge) A problem that can be solved; but not in a reasonable amount of time as the problem size increases // but has an exponential (or worse) time complexity // but there is no polynomial (or less) time solution;	2
03	5	Mark is for AO1 (understanding) One;	1
03	6	One mark is for AO1 (knowledge) and one mark is for AO1 (understanding) AO1 knowledge $O(n)$; Max 1 for AO1 understanding As the size of the list increases the time taken increases at the same rate; There is a loop that repeats n times;	2

Question			Marks
03	7	One mark is for AO1 (knowledge) and one mark is for AO1 (understanding) AO1 knowledge $O(\log n)$; Max 1 for AO1 understanding Each comparison halves the size of the list that has to be searched through; The time taken increases as the size of the list increases but by smaller and smaller amounts; If the size of the list doubles then the number of comparisons needed only increases by 1;	2

Question			Marks
04	1	All marks for AO1 (understanding) Max 2 for advantages of dynamic data structures No wasted memory; Can grow as more data is added to the data structure // no limit on number of items that can be added (except due to hardware limitations); Resources only allocated as they are needed (with static data structures they are allocated at creation even if not needed until later); Max 2 for disadvantages of dynamic data structures Additional memory needed for pointers; Can result in memory leak (if memory that is no longer needed is not returned to the heap); Can take longer to access an item directly (for data structures that allow this); A. can take longer to add a new item to the data structure (as memory needs to be allocated)	4
04	2	All marks for AO1 (understanding) Check that the queue is not already full; (if it isn't) then add 1 to the value of the rear pointer; then add the new item to the position indicated by the rear pointer; Alternative answer Check that the queue is not already full; (if it isn't) then add the new item to the position indicated by the rear pointer; then add 1 to the value of the rear pointer; Max 2 if any errors Max 1 if circular queue has been described	3

04	3	All marks for AO1 (understanding) Starting with the item at the rear of the queue move each item back one place in the array; Until you (reach the start of the queue or) find an item with the same <u>or</u> higher priority than the item to add; NE. same priority NE. higher priority Add the new item in the position before that item; A. answers which have the front of the queue as the last item in the array, start at the front and move each item forward one until the correct insertion point is found. A. answers that start from the front of the queue until position to insert item is found and then start at the back and move each item back one until position to insert item is found.	3
----	---	---	---

Question			Marks															
05	1	4 marks for AO3 (design) and 8 marks for AO3 (programming) <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	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 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																

		<u>Guidance</u> Evidence of AO3 design – 4 points: Evidence of design to look for in responses: 1. Identifying that data structure(s) are needed to store ten frequencies 2. Identifying that a loop is needed that repeats a number of times determined by the first number entered by the user 3. Identifying that a Boolean (or equivalent) variable is needed to store if the data was multimodal 4. Selection structure that either outputs a calculated number (I. incorrectly calculated) or a message saying "Data was multimodal" (A. any suitable message) Note that AO3 (design) points are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works. Evidence for AO3 programming – 8 points: Evidence of programming to look for in response: 5. Suitable prompts asking user to enter the number of digits followed by user inputs being assigned to appropriate variable R. if inside or after iterative structure 6. Correct number of numeric digits obtained from the user 7. Adds one to correct frequency count R. if only works for one digit 8. Selection structure, inside iterative structure, that correctly compares calculated frequency (I. incorrect frequency) of a digit with the highest frequency found so far 9. Boolean (or equivalent) variable that is used to indicate if data is multimodal is set to true under correct circumstances 10. Boolean (or equivalent) variable that is used to indicate if data is multimodal is set to false when new higher frequency is found 11. Program works correctly if the data has more than one modal value A. any sensible message 12. Program displays the correct frequency of the modal value under all circumstances and does not say data is multimodal when it is not I. frequency being displayed when data is multimodal Max 11 if code contains any errors	
05	2	Mark is for AO3 (evaluate) **** SCREEN CAPTURE **** <i>Must match code from 05.1, including prompts on screen capture matching those in code.</i> <i>Code for 05.1 must be sensible.</i> Screen captures showing:	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.	
	<pre>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</pre>	
	<pre>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</pre>	

Question			Marks
06	1	Mark is for AO2 (analyse) That the company name being searched for does not exist (in the simulation); A. user misspelt the company name	1
06	2	Marks are for AO2 (analyse) There is no need for the variable <code>Index</code> // no need to assign the value of <code>-1</code> to <code>Index</code>; If the loop terminates then can just return (the constant value) <code>-1</code> // if the company name is not found then can just return (the constant value) <code>-1</code>;	2

Question			Marks
07	1	Mark is for AO2 (analyse) GetDetails; R. if spelt incorrectly R. if any additional code I. case and spacing	1
07	2	Mark is for AO2 (analyse) Details; OldCapacity; R. if spelt incorrectly R. if any additional code I. case and spacing Max 1 mark	1
07	3	Mark is for AO1 (understanding) Private attributes can only be accessed by the class/object they belong to whereas protected attributes can also be accessed by any classes that inherit from the class they belong to; A. file instead of class/object (Java only) NE. private attribute can only be accessed by the class/object they belong to whereas protected attributes can be accessed by others classes/objects	1
07	4	Mark is for AO2 (analyse) The calculation of the daily costs will be inconsistent as it will be different in the AlterCapacity method; A. it won't have been updated in other locations which use that constant	1

Question			Marks
08	1	Mark is for AO2 (analyse) LargeSettlement; R. if spelt incorrectly R. if any additional code I. case and spacing	1
08	2	Mark is for AO2 (analyse) Household; R. if spelt incorrectly R. if any additional code I. case and spacing	1
08	3	Mark is for AO2 (analyse) Company; R. if spelt incorrectly R. if any additional code I. case and spacing	1
08	4	Mark is for AO2 (analyse) Aggregation;	1

Question			Marks
09		All marks are for AO2 (analyse) It stores the cumulative reputation for the companies in an array/list; It then generates a random number which is less than the total reputation; A. generates a random number based on the total reputation of the companies Finds the first cumulative reputation that the number is less than; The position of this cumulative reputation in the list indicates the company that the household will use;	4

Question		Marks
10	1	All marks for AO3 (programming) 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; Max 4 if code contains errors
10	2	Mark is for AO3 (evaluate) **** SCREEN CAPTURE **** <i>Must match code from 10.1, including prompts on screen capture matching those in code.</i> <i>Code for 10.1 must be sensible.</i> 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; <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
11	1	All marks for AO3 (programming) 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 The following all relate to the <code>AddHousehold</code> method: 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> Max 6 if code contains errors	7
11	2	Mark is for AO3 (evaluate) **** SCREEN CAPTURE **** <i>Must match code from 11.1, including prompts on screen capture matching those in code.</i> <i>Code for 11.1 must be sensible.</i> Screen capture(s) showing that households with an X value less than 100 have an eat out percentage of 1; <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>	1

Question			Marks
12	1	All marks for AO3 (programming) Marks for changes to the <code>Simulation</code> class: - 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; Marks for changes to the <code>Company</code> class: - 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. Max 9 marks if code contains errors	10

Question			Marks
12	2	Mark is for AO3 (evaluate) **** SCREEN CAPTURE **** <i>Must match code from 12.1, including prompts on screen capture matching those in code.</i> <i>Code for 12.1 must be sensible.</i> 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. <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>	1

Question			Marks
13	1	All marks for AO3 (programming) - Created new method called <code>GetOrderedListOfOutlets</code>; R. other names for method I. case and minor typos - Method returns a list/array; - Outlet 0 is added to the route first; - Iterative structure that repeats until all outlets have been added to the route; - 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; - 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 - 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 - Route created contains all the company's outlets; - Shortest distance between two nodes variable set to suitable starting value and reset after each outlet (except last one) is added to route; <code>GetOrderedListOfOutlets</code> implements the algorithm described in Figure 6 in the question; - 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 Max 10 if code contains errors or if other parts of the subroutine no longer work correctly	11

Question		Marks
13	2	1
Mark is for AO3 (evaluate) **** SCREEN CAPTURE **** Must match code from 13.1, including prompts on screen capture matching those in code. Code for 13.1 must be sensible. Screen capture(s) showing that the delivery cost for AQA Burgers is 22.10446; <pre>***** ***** MENU ***** ***** 1. Display details of households 2. Display details of companies 3. Modify company 4. Add new company 6. Advance to next day 0. 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>		

VB.Net

Question		Marks
05	1	12
<pre> Dim Current As Integer Dim Frequencies(9) As Integer Dim ModeFrequency As Integer = 0 Dim Multimodal As Boolean = False Dim NoOfDigits As Integer Console.Write("Enter number of digits: ") NoOfDigits = Console.ReadLine For count = 1 To NoOfDigits Console.Write("Enter a numeric digit: ") Current = Console.ReadLine Frequencies(Current) += 1 Next For Count = 0 To 9 If Frequencies(Count) > ModeFrequency Then ModeFrequency = Frequencies(Count) Multimodal = False ElseIf Frequencies(Count) = ModeFrequency Then Multimodal = True End If Next If Multimodal Then Console.WriteLine("Data was multimodal") Else Console.WriteLine("The modal digit appeared " & ModeFrequency & " times") End If </pre>		
10	1	5
<pre> Public Sub AddCompany() Dim Balance, X, Y As Integer Dim CompanyName, TypeOfCompany As String Do Console.Write("Enter a name for the company: ") CompanyName = Console.ReadLine If CompanyName = "" Then Console.WriteLine("You must enter a name.") End If If GetIndexOfCompany(CompanyName) <> -1 Then Console.WriteLine("That name is already being used.") End If Loop While CompanyName = "" Or GetIndexOfCompany(CompanyName) <> - 1 Console.Write("Enter the starting balance for the company: ") </pre> Alternative answer <pre> Public Sub AddCompany() Dim Balance, X, Y As Integer Dim CompanyName, TypeOfCompany As String Console.Write("Enter a name for the company: ") CompanyName = Console.ReadLine While CompanyName = "" Or GetIndexOfCompany(CompanyName) <> -1 If CompanyName = "" Then Console.WriteLine("You have to enter a company name") Else </pre>		

		<pre> Console.WriteLine("Company name already exists, it has to be unique") End If Console.Write("Enter a name for the company: ") CompanyName = Console.ReadLine End While Console.Write("Enter the starting balance for the company: ") </pre>	
11	1	<pre> Public Sub AddHousehold() Dim X, Y As Integer GetRandomLocation(X, Y) If X < 100 Then Dim Temp As New AffluentHousehold(X, Y) Households.Add(Temp) Else Dim Temp As New Household(X, Y) Households.Add(Temp) End If End Sub Class AffluentHousehold Inherits Household Public Sub New(ByVal X As Integer, ByVal Y As Integer) MyBase.New(X, Y) ChanceEatOutPerDay = 1 End Sub End Class </pre>	7
12	1	From the simulation class <pre> Public Sub ModifyCompany(ByVal Index As Integer) Dim Choice As String Dim OutletIndex, X, Y As Integer Dim CloseCompany As Boolean Console.WriteLine(Environment.NewLine & "*****") Console.WriteLine("***** MODIFY COMPANY *****") Console.WriteLine("*****") Console.WriteLine("1. Open new outlet") Console.WriteLine("2. Close outlet") Console.WriteLine("3. Expand outlet") Console.WriteLine("4. Get Loan") Console.WriteLine("5. Pay back loan") Console.Write(Environment.NewLine & "Enter your choice: ") Choice = Console.ReadLine Console.WriteLine() If Choice = "2" Or Choice = "3" Then ... ElseIf Choice = "1" Then ... ElseIf Choice = "4" Then Console.Write("Enter the interest rate for the loan: ") Dim Rate As Single = Console.ReadLine Companies(Index).GetLoan(Rate) ElseIf Choice = "5" Then Console.Write("Enter the amount to pay back: ") Dim PayBackAmount As Single = Console.ReadLine Companies(Index).PayBackLoan(PayBackAmount) </pre>	10

```

    End If
    Console.WriteLine()
End Sub

```

From the Company class

```

Class Company
    Protected Name, Category As String
    Protected Balance, ReputationScore, AvgCostPerMeal,
    AvgPricePerMeal, DailyCosts, FamilyOutletCost, FastFoodOutletCost,
    NamedChefOutletCost, FuelCostPerUnit, BaseCostOfDelivery As Single
    Protected Outlets As New ArrayList
    Protected FamilyFoodOutletCapacity, FastFoodOutletCapacity,
    NamedChefOutletCapacity As Integer
    Protected InterestRate As Single
    Protected LoanBalance As Single
    ...

```

```

Public Function ProcessDayEnd() As String
    ...
    Next
    Details &= "Previous balance for company: " & Balance.ToString &
    Environment.NewLine
    Balance += ProfitLossFromOutlets - DailyCosts - DeliveryCosts -
    (LoanBalance * InterestRate)
    Details &= "New balance for company: " & Balance.ToString
    Return Details
End Function

```

```

Public Sub GetLoan(ByVal Rate As Single)
    If LoanBalance = 0 Then
        Balance += 10000
        LoanBalance = 10000
        InterestRate = Rate
    End If
End Sub

```

```

Public Sub PayBackLoan(ByVal Amount As Single)
    LoanBalance -= Amount
    Balance -= Amount
End Sub

```

Alternative answer for taking a loan

In ModifyCompany method in Simulation class

```

...
ElseIf Choice = "4" Then
    If Companies(Index).GetLoanBalance() <= 0 Then
        Console.Write("Enter Interest Rate: ")
        InterestRate = Console.ReadLine()
        Companies(Index).TakeOutLoan(InterestRate)
    End If

```

		Methods in Company Class Public Function GetLoanBalance() As Single Return LoanBalance End Function Public Sub TakeOutLoan(ByVal InterestRate As Single) Me.InterestRate = InterestRate LoanBalance = 10000 Balance += 10000 End Sub	
13	1	Private Function GetOrderedListOfOutlets() As ArrayList Dim OrderedList As New ArrayList Dim NearestOutlet As Integer OrderedList.Add(0) While OrderedList.Count < Outlets.Count Dim ShortestDistanceSoFar As Single = 1000000 For Count = 1 To Outlets.Count - 1 If Not OrderedList.Contains(Count) Then Dim Temp As Integer = GetDistanceBetweenTwoOutlets(OrderedList(OrderedList.Count - 1), Count) If Temp < ShortestDistanceSoFar Then NearestOutlet = Count ShortestDistanceSoFar = Temp End If End If Next OrderedList.Add(NearestOutlet) End While Return OrderedList End Function Public Function CalculateDeliveryCost() As Single Dim ListOfOutlets As ArrayList = GetOrderedListOfOutlets() Dim TotalDistance As Single = 0 Dim TotalCost As Single = 0 For Count = 0 To ListOfOutlets.Count - 2 TotalDistance += GetDistanceBetweenTwoOutlets(ListOfOutlets(Count), ListOfOutlets(Count + 1)) Next TotalCost += TotalDistance * FuelCostPerUnit Return TotalCost End Function	11

Python 2

Question			Marks
05	1	<pre> ModeFrequency = 0 Multimodal = False Frequencies = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0] NoOfDigits = int(input("Enter number of digits: ")) for count in range (1, NoOfDigits + 1): Current = int(input("Enter a numeric digit: ")) Frequencies[Current] += 1 for Count in range (0, 10): if Frequencies[Count] > ModeFrequency: ModeFrequency = Frequencies[Count] Multimodal = False elif Frequencies[Count] == ModeFrequency: Multimodal = True if Multimodal: print("Data was multimodal") else: print("The modal digit appeared " + str(ModeFrequency) + " times") </pre>	12
10	1	<pre> def AddCompany(self): CompanyName = input("Enter a name for the company: ") while CompanyName == "" or self.GetIndexOfCompany(CompanyName) != -1: if CompanyName == "": print("You have to enter a company name") else: print("Company name already exists, it has to be unique") CompanyName = input("Enter a name for the company: ") Balance = int(input("Enter the starting balance for the company: ")) TypeOfCompany = "" while not (TypeOfCompany == "1" or TypeOfCompany == "2" or TypeOfCompany == "3"): TypeOfCompany = input("Enter 1 for a fast food company, 2 for a family company or 3 for a named chef company: ") if TypeOfCompany == "1": TypeOfCompany = "fast food" elif TypeOfCompany == "2": TypeOfCompany = "family" else: TypeOfCompany = "named chef" X, Y = self._SimulationSettlement.GetRandomLocation() NewCompany = Company(CompanyName, TypeOfCompany, Balance, X, Y, self._FuelCostPerUnit, self._BaseCostforDelivery) self._Companies.append(NewCompany) </pre>	5
11	1	<pre> def AddHousehold(self): X, Y = self.GetRandomLocation() if X < 100: Temp = AffluentHousehold(X, Y) self._Households.append(Temp) else: Temp = Household(X, Y) self._Households.append(Temp) class AffluentHousehold(Household): </pre>	7

		<pre> def __init__(self, X, Y): super(AffluentHousehold, self).__init__(X, Y) self._ChanceEatOutPerDay = 1 </pre>	
12	1	From the simulation class <pre> def ModifyCompany(self, Index): print("\n*****") print("***** MODIFY COMPANY *****") print("*****") print("1. Open new outlet") print("2. Close outlet") print("3. Expand outlet") print("4. Get Loan") print("5. Pay back loan") Choice = input("\nEnter your choice: ") print() if Choice == "2" or Choice == "3": ... elif Choice == "1": ... elif Choice == "4": Rate = float(input("Enter the interest rate for the loan: ")) self._Companies[Index].GetLoan(Rate) elif Choice == "5": PayBackAmount = float(input("Enter the amount to pay back: ")) self._Companies[Index].PayBackLoan(PayBackAmount) print() </pre> From the Company class <pre> class Company: def __init__(self, Name, Category, Balance, X, Y, FuelCostPerUnit, BaseCostOfDelivery): self._Outlets = [] .. self._DailyCosts = 100 self._InterestRate = 0.0 self._LoanBalance = 0.0 if self._Category == "fast food": ... def ProcessDayEnd(self): ... Details += "Previous balance for company: " + str(self._Balance) + "\n" self._Balance += ProfitLossFromOutlets - self._DailyCosts - DeliveryCosts - (self._LoanBalance * self._InterestRate) Details += "New balance for company: " + str(self._Balance) return Details def GetLoan(self, Rate): if self._LoanBalance == 0: self._Balance += 10000 self._LoanBalance = 10000 </pre>	10

		<pre> self._InterestRate = Rate def PayBackLoan(self, Amount): self._LoanBalance -= Amount self._Balance -= Amount </pre> Alternative answer for taking a loan In ModifyCompany method in Simulation class <pre> ... elif Choice == "4": if Companies[Index].GetLoanBalance() <= 0: Rate = float(input("Enter the interest rate for the loan: ")) self._Companies[Index].TakeOutLoan(Rate) </pre> Methods in Company Class <pre> def GetLoanBalance(self): return self._LoanBalance def TakeOutLoan(self, InterestRate): self._InterestRate = InterestRate self._LoanBalance = 10000 self._Balance += 10000 </pre>	
13	1	<pre> def __GetOrderedListOfOutlets(self): OrderedList = [0] while len(OrderedList) < len(self._Outlets): ShortestDistanceSoFar = 1000000.0 for Count in range(1, len(self._Outlets)): if not Count in OrderedList: Temp = self.__GetDistanceBetweenTwoOutlets(OrderedList[len(OrderedList) - 1], Count) if Temp < ShortestDistanceSoFar: NearestOutlet = Count ShortestDistanceSoFar = Temp OrderedList.append(NearestOutlet) return OrderedList def CalculateDeliveryCost(self): ListOfOutlets = self.__GetOrderedListOfOutlets() TotalDistance = 0.0 for Current in range (0, len(ListOfOutlets) - 1): TotalDistance += self.__GetDistanceBetweenTwoOutlets(ListOfOutlets[Current], ListOfOutlets[Current + 1]) TotalCost = TotalDistance * self._FuelCostPerUnit return TotalCost </pre>	11

Python 3

Question			Marks
05	1	<pre> ModeFrequency = 0 Multimodal = False Frequencies= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0] NoOfDigits = int(raw_input("Enter number of digits: ")) for count in range (1, NoOfDigits + 1): Current = int(raw_input("Enter a numeric digit: ")) Frequencies[Current] += 1 for Count in range (0, 10): if Frequencies[Count] > ModeFrequency: ModeFrequency = Frequencies[Count] Multimodal = False elif Frequencies[Count] == ModeFrequency: Multimodal = True if Multimodal: print "Data was multimodal" else: print "The modal digit appeared " + str(ModeFrequency) + " times" </pre>	12
10	1	<pre> def AddCompany(self): CompanyName = raw_input("Enter a name for the company: ") while CompanyName == "" or self.GetIndexOfCompany(CompanyName) != -1: if CompanyName == "": print "You have to enter a company name" else: print "Company name already exists, it has to be unique" CompanyName = raw_input("Enter a name for the company: ") Balance = int(raw_input("Enter the starting balance for the company: ")) TypeOfCompany = "" while not(TypeOfCompany == "1" or TypeOfCompany == "2" or TypeOfCompany == "3"): TypeOfCompany = raw_input("Enter 1 for a fast food company, 2 for a family company or 3 for a named chef company: ") if TypeOfCompany == "1": TypeOfCompany = "fast food" elif TypeOfCompany == "2": TypeOfCompany = "family" else: TypeOfCompany = "named chef" X, Y = self._SimulationSettlement.GetRandomLocation() NewCompany = Company(CompanyName, TypeOfCompany, Balance, X, Y, self._FuelCostPerUnit, self._BaseCostforDelivery) self._Companies.append(NewCompany) </pre>	5
11	1	<pre> def AddHousehold(self): X, Y = self.GetRandomLocation() if X < 100: Temp = AffluentHousehold(X, Y) self._Households.append(Temp) else: Temp = Household(X, Y) self._Households.append(Temp) class AffluentHousehold(Household): </pre>	7

		<pre>def __init__(self, X, Y): super(AffluentHousehold, self).__init__(X, Y) self._ChanceEatOutPerDay = 1</pre>	
12	1	From the simulation class <pre>def ModifyCompany(self, Index): print "\n*****" print "***** MODIFY COMPANY *****" print "*****" print "1. Open new outlet" print "2. Close outlet" print "3. Expand outlet" print "4. Get Loan" print "5. Pay back loan" Choice = raw_input("\nEnter your choice: ") print if Choice == "2" or Choice == "3": ... elif Choice == "1": ... elif Choice == "4": Rate = float(raw_input("Enter the interest rate for the loan: ")) self._Companies[Index].GetLoan(Rate) elif Choice == "5": PayBackAmount = float(raw_input("Enter the amount to pay back: ")) self._Companies[Index].PayBackLoan(PayBackAmount) print</pre> From the Company class <pre>class Company: def __init__(self, Name, Category, Balance, X, Y, FuelCostPerUnit, BaseCostOfDelivery): self._Outlets = [] .. self._DailyCosts = 100 self._InterestRate = 0.0 self._LoanBalance = 0.0 if self._Category == "fast food": ... def ProcessDayEnd(self): ... Details += "Previous balance for company: " + str(self._Balance) + "\n" self._Balance += ProfitLossFromOutlets - self._DailyCosts - DeliveryCosts - (self._LoanBalance * self._InterestRate) Details += "New balance for company: " + str(self._Balance) return Details def GetLoan(self, Rate): if self._LoanBalance == 0: self._Balance += 10000 self._LoanBalance = 10000 self._InterestRate = Rate</pre>	10

		<pre>def PayBackLoan(self, Amount): self._LoanBalance -= Amount self._Balance -= Amount</pre> Alternative answer for taking a loan In ModifyCompany method in Simulation class <pre>... elif Choice == "4": if Companies[Index].GetLoanBalance() <= 0: Rate = float(raw_input("Enter the interest rate for the loan: ")) self._Companies[Index].TakeOutLoan(Rate)</pre> Methods in Company Class <pre>def GetLoanBalance(self): return self._LoanBalance def TakeOutLoan(self, InterestRate): self._InterestRate = InterestRate self._LoanBalance = 10000 self._Balance += 10000</pre>	
13	1	<pre>def __GetOrderedListOfOutlets(self): OrderedList = [0] while len(OrderedList) < len(self._Outlets): ShortestDistanceSoFar = 1000000.0 for Count in range(1, len(self._Outlets)): if not Count in OrderedList: Temp = self.__GetDistanceBetweenTwoOutlets(OrderedList[len(OrderedList) - 1], Count) if Temp < ShortestDistanceSoFar: NearestOutlet = Count ShortestDistanceSoFar = Temp OrderedList.append(NearestOutlet) return OrderedList def CalculateDeliveryCost(self): ListOfOutlets = self.__GetOrderedListOfOutlets() TotalDistance = 0.0 for Current in range (0, len(ListOfOutlets) - 1): TotalDistance += self.__GetDistanceBetweenTwoOutlets(ListOfOutlets[Current], ListOfOutlets[Current + 1]) TotalCost = TotalDistance * self._FuelCostPerUnit return TotalCost</pre>	11

C#

Question		Marks
05	1	12
	<pre> int current; int[] frequencies = new int[10]; int modeFrequency = 0; bool multimodal = false; int noOfDigits; Console.WriteLine("Enter number of digits: "); noOfDigits = Convert.ToInt32(Console.ReadLine()); for (int i = 0; i < noOfDigits; i++) { Console.WriteLine("Enter a numeric digit: "); current = Convert.ToInt32(Console.ReadLine()); frequencies[current] += 1; } for (int i = 0; i < 10; i++) { if (frequencies[i] > modeFrequency) { modeFrequency = frequencies[i]; multimodal = false; } else if (frequencies[i] == modeFrequency) { multimodal = true; } } if (multimodal) { Console.WriteLine("Data was multimodal"); } else { Console.WriteLine("The modal digit appeared " + modeFrequency + " times"); } Console.ReadLine(); </pre>	
10	1	5
	<pre> private void AddCompany() { int balance, x = 0, y = 0; string companyName, typeOfCompany = "9"; do { Console.WriteLine("Enter a name for the company: "); companyName = Console.ReadLine(); if (companyName == "") { Console.WriteLine("You must enter a name."); } else if (GetIndexofCompany(companyName) != -1) { Console.WriteLine("That name is already being used."); } } while (companyName == "" GetIndexofCompany(companyName) != -1); Console.WriteLine("Enter the starting balance for the company: "); </pre>	

		Alternative answer <pre>private void AddCompany() { int balance, x = 0, y = 0; string companyName, typeOfCompany = "9"; Console.WriteLine("Enter a name for the company: "); companyName = Console.ReadLine(); while (companyName == "" GetIndexOfCompany(companyName) != - 1) { if (companyName == "") { Console.WriteLine("You must enter a name."); } else if (GetIndexOfCompany(companyName) != -1) { Console.WriteLine("That name is already being used."); } Console.WriteLine("Enter a name for the company: "); companyName = Console.ReadLine(); } Console.WriteLine("Enter the starting balance for the company: "); }</pre>	
11	1	<pre>public void AddHousehold() { int x = 0, y = 0; GetRandomLocation(ref x, ref y); if (x < 100) { AffluentHousehold temp = new AffluentHousehold(x, y); households.Add(temp); } else { Household temp = new Household(x, y); households.Add(temp); } } class AffluentHousehold : Household { public AffluentHousehold(int x, int y) :base(x, y) { chanceEatOutPerDay = 1; } }</pre>	7
12	1	From the simulation class <pre>public void ModifyCompany(int index) { string choice; int outletIndex, x, y; bool closeCompany; Console.WriteLine("\n*****"); Console.WriteLine("***** MODIFY COMPANY *****"); Console.WriteLine("*****"); }</pre>	10

```

Console.WriteLine("1. Open new outlet");
Console.WriteLine("2. Close outlet");
Console.WriteLine("3. Expand outlet");
Console.WriteLine("4. Get Loan");
Console.WriteLine("5. Pay back loan");
Console.Write("\nEnter your choice: ");
choice = Console.ReadLine();
if (choice == "2" || choice == "3")
...
else if (choice == "1")
...
else if (choice == "4")
{
    Console.Write("Enter the interest rate for the loan: ");
    double rate = Convert.ToDouble(Console.ReadLine());
    companies[index].GetLoan(rate);
}
else if (choice == "5")
{
    Console.Write("Enter the amount to pay back: ");
    double payBackAmount = Convert.ToDouble(Console.ReadLine());
    companies[index].PayBackLoan(payBackAmount);
}
Console.WriteLine();
}

```

From the company class

```

class Company
{
    private static Random rnd = new Random();
    protected string name, category;
    protected double balance, reputationScore, avgCostPerMeal,
    avgPricePerMeal, dailyCosts, familyOutletCost, fastFoodOutletCost,
    namedChefOutletCost, fuelCostPerUnit, baseCostOfDelivery;
    protected List<Outlet> outlets = new List<Outlet>();
    protected int familyFoodOutletCapacity, fastFoodOutletCapacity,
    namedChefOutletCapacity;
    protected double loanBalance;
    protected double interestRate;
    ...

    public string ProcessDayEnd()
    ...
    }
    details += "Previous balance for company: " + balance.ToString()
    + "\n";
    balance += profitLossFromOutlets - dailyCosts - deliveryCosts -
    (loanBalance * interestRate);
    details += "New balance for company: " + balance.ToString();
    return details;
}

public void GetLoan(double rate)
{
    if (loanBalance == 0)
    {
        balance += 10000;
    }
}

```

		<pre> interestRate = rate; } } public void PayBackLoan(double amount) { loanBalance -= amount; balance -= amount; } </pre> Alternative answer for taking a loan In ModifyCompany method in Simulation class <pre> ... else if (choice == "4") { if (companies[index].GetLoanBalance() <= 0) { Console.WriteLine("Enter the interest rate for the loan: "); double rate = Convert.ToDouble(Console.ReadLine()); companies[index].TakeOutLoan(rate); } } </pre> Methods in Company Class <pre> public double GetLoanBalance() { return loanBalance; } public void TakeOutLoan(double interestRate) { this.interestRate = interestRate; loanBalance = 10000; balance += 10000; } </pre>	
13	1	<pre> private List<int> GetOrderedListOfOutlets() { List<int> orderedList = new List<int>(); int nearestOutlet = 0; ; orderedList.Add(0); while (orderedList.Count < outlets.Count) { double shortestDistanceSoFar = 1000000; for (int count = 1; count < outlets.Count ; count++) { if (!orderedList.Contains(count)) { double temp = GetDistanceBetweenTwoOutlets(orderedList[orderedList.Count - 1], count); </pre>	11

		<pre> if (temp < shortestDistanceSoFar) { nearestOutlet = count; shortestDistanceSoFar = temp; } } } orderedList.Add(nearestOutlet); } return orderedList; } public double CalculateDeliveryCost() { List<int> listOfOutlets = new List<int>(GetOrderedListOfOutlets()); double totalDistance = 0; double totalCost = 0; for (int current = 0; current < listOfOutlets.Count - 1; current++) { totalDistance += GetDistanceBetweenTwoOutlets(listOfOutlets[current], listOfOutlets[current + 1]); } totalCost = totalDistance * fuelCostPerUnit; return totalCost; } </pre>	
--	--	--	--

PASCAL/Delphi

Question			Marks
05	1	<pre> var Current, ModeFrequency, NoOfDigits, Count : Integer; Frequencies : array of Integer; Multimodal : Boolean; begin SetLength(Frequencies, 10); ModeFrequency := 0; MultiModal := False; for Count := 0 to 9 do Frequencies[Count] := 0; write('Enter number of digits: '); readln(NoOfDigits); for Count := 1 to NoOfDigits do begin write('Enter a numeric digit: '); readln(Current); Frequencies[Current] += 1; end; for Count := 0 to 9 do begin if Frequencies[Count] > ModeFrequency then begin ModeFrequency := Frequencies[Count]; MultiModal := False; end else if Frequencies[Count] = ModeFrequency then MultiModal := True; end; end; if MultiModal = True then writeln('Data was multimodal') else writeln('The modal digit appeared ' + inttostr(ModeFrequency) + ' times'); readln; end. </pre>	12
10	1	<pre> procedure Simulation.AddCompany(); var Balance, X, Y : Integer; CompanyName, TypeOfCompany : String; NewCompany : Company; begin repeat write('Enter a name for the company: '); readln(CompanyName); if CompanyName = '' then writeln('You must enter a name'); if GetIndexOfCompany(CompanyName) <> - 1 then writeln('That name is already being used'); until (CompanyName <> '') and (GetIndexOfCompany(CompanyName) = - 1); write('Enter the starting balance for the company: '); </pre> Alternative Answer <pre> write('Enter a name for the company: '); </pre>	5

		<pre> readln(CompanyName); while (CompanyName = '') or (GetIndexOfCompany(CompanyName) <> -1) do begin if CompanyName = '' then writeln('You have to enter a company name') else writeln('Company name already exists - it has to be unique'); write('Enter a name for the company: '); readln(CompanyName); end; </pre>	
11	1	<pre> type AffluentHousehold = class(Household) constructor New(X : Integer; Y : Integer); end; constructor AffluentHousehold.New(X : Integer; Y : Integer); begin XCoord := X; YCoord := Y; ChanceEatOutPerDay := 1; ID := NextID; inc(NextID); end; procedure Settlement.AddHousehold(); var X, Y : Integer; Temp : Household; TempAff : AffluentHousehold; begin SetLength(Households, length(Households) + 1); GetRandomLocation(X, Y); if X < 100 then begin TempAff := AffluentHousehold.New(X, Y); Households[length(Households) - 1] := TempAff; end else begin Temp := Household.New(X, Y); Households[length(Households) - 1] := Temp; end; end; </pre>	7
12	1	From the simulation class <pre> procedure Simulation.ModifyCompany(Index : Integer); var Choice : String; OutletIndex, CompIndex, X, Y : Integer; CloseCompany : Boolean; Rate, PayBackAmount : Real; begin writeln; writeln('*****'); writeln('***** MODIFY COMPANY *****'); writeln('*****'); writeln('1. Open new outlet'); writeln('2. Close outlet'); writeln('3. Expand outlet'); </pre>	10

```

writeln('4. Get loan');
writeln('5. Pay back loan');
writeln;
write('Enter your choice: ');
readln(Choice);
writeln;
if (Choice = '2') or (Choice = '3') then
...
else if Choice = '1' then
...
else if Choice = '4' then
begin
write('Enter the interest rate for the loan: ');
readln(Rate);
Self.Companies[Index].GetLoan(Rate);
end
else if Choice = '5' then
begin
write('Enter the amount to pay back: ');
readln(PayBackAmount);
Companies[Index].PayBackLoan(PayBackAmount);
end;
writeln;
end;

```

From the Company class

```

type
Company = class
protected
Name, Category : String;
Balance, ReputationScore, AvgCostPerMeal, AvgPricePerMeal,
DailyCosts, FamilyOutletCost, FastFoodOutletCost,
NamedChefOutletCost, FuelCostPerUnit, BaseCostOfDelivery,
InterestRate, LoanBalance : Real;
Outlets: TOutletArray;
...
function CalculateDeliveryCost() : Real;
procedure GetLoan(Rate : Real);
procedure PayBackLoan(Amount : Real);

constructor Company.New(NameInput : String; CategoryInput : String;
BalanceInput : Real; X : Integer; Y : Integer; FuelCostPerUnitInput
: Real; BaseCostOfDeliveryInput : Real);
begin
Self.FamilyOutletCost := 1000;
Self.FastFoodOutletCost := 2000;
Self.NamedChefOutletCost:= 15000;
Self.FamilyFoodOutletCapacity := 150;
Self.FastFoodOutletCapacity := 200;
Self.NamedChefOutletCapacity := 50;
Self.LoanBalance := 0;
Self.Name := NameInput;

procedure Company.GetLoan(Rate : Real);
begin
if Self.LoanBalance = 0 then
begin

```

		<pre> Self.Balance += 10000; Self.LoanBalance := 10000; Self.InterestRate := Rate; end; end; procedure Company.PayBackLoan(Amount : Real); begin Self.LoanBalance -= Amount; Self.Balance -= Amount; end; function Company.ProcessDayEnd() : String; ... Details += 'Previous balance for company: ' + floattostr(self.Balance) + #13#10; Self.Balance += ProfitLossFromOutlets - Self.DailyCosts - DeliveryCosts - (Self.LoanBalance * Self.InterestRate); Details += 'New balance for company: ' + floattostr(Self.Balance); ProcessDayEnd := Details; end; </pre> Alternative answer for taking a loan In ModifyCompany method in Simulation class <pre> ... else if Choice = '4' then if Companies[Index].GetLoanBalance() <= 0 then begin write('Enter the interest rate for the loan: '); readln(Rate); Self.Companies[Index].GetLoan(Rate); end ... </pre> Company Class <pre> function GetLoanBalance(): Real; ... function Company.GetLoanBalance() : Real; begin GetLoanBalance := LoanBalance; end; procedure Company.GetLoan(InterestRate : Real); begin Self.Balance += 10000; Self.LoanBalance := 10000; Self.InterestRate := Rate; end; </pre>	
13	1	<pre> function Company.GetOrderedListOfOutlets() : TIntegerArray; var OrderedList : TIntegerArray; </pre>	11

		<pre> NearestOutlet, Count, Index, Temp : Integer; ShortestDistanceSoFar : Real; ItemInList : Boolean; begin SetLength(OrderedList, 1); OrderedList[0] := 0; while length(OrderedList) < length(Self.Outlets) do begin ShortestDistanceSoFar := 1000000; for Count := 1 to length(Self.Outlets) - 1 do begin ItemInList := False; for Index := low(OrderedList) to high(OrderedList) do begin if OrderedList[Index] = Count then ItemInList := True; end; if ItemInList = false then begin Temp := trunc(GetDistanceBetweenTwoOutlets(OrderedList[length(OrderedList) - 1], Count)); if Temp < ShortestDistanceSoFar then begin NearestOutlet := Count; ShortestDistanceSoFar := Temp; end; end; end; SetLength(OrderedList, length(OrderedList) + 1); OrderedList[high(OrderedList)] := NearestOutlet; end; GetOrderedListOfOutlets := OrderedList; end; function Company.CalculateDeliveryCost() : Real; var ListOfOutlets : TIntegerArray; TotalDistance, TotalCost : Real; Current : Integer; begin ListOfOutlets := GetOrderedListOfOutlets(); TotalDistance := 0; for Current := 0 to length(ListOfOutlets) - 2 do TotalDistance += GetDistanceBetweenTwoOutlets(ListOfOutlets[Current], ListOfOutlets[Current + 1]); TotalCost := TotalDistance * Self.FuelCostPerUnit; CalculateDeliveryCost := TotalCost; end; </pre>	
--	--	---	--

JAVA

Question			Marks
05	1	<pre> int modeFrequency = 0, input; int[] frequencies = new int[10]; boolean multimodal = false; Console.WriteLine("Enter the number of digits you would like to enter:"); int numOfDigits = Integer.parseInt(Console.ReadLine()); for (int i = 0; i < numOfDigits; i++) { Console.WriteLine("Enter a digit:"); input = Integer.parseInt(Console.ReadLine()); frequencies [input]++; } modeFrequency = frequencies [0]; for (int i = 1; i < 10; i++) { if (modeFrequency == frequencies [i]) { multimodal = true; } else if (frequencies [i] > modeFrequency) { multimodal = false; modeFrequency = frequencies [i]; } } if (multimodal) { Console.WriteLine("Data was multimodal"); } else { Console.WriteLine("The modal digit appeared " + modeFrequency + " times"); </pre>	12
10	1	<pre> public void addCompany() { int balance, x, y; String companyName, typeOfCompany; do { Console.write("Enter a name for the company: "); companyName = Console.ReadLine(); if (companyName.equals("")) { Console.WriteLine("You must enter a name."); } else if (getIndexOfCompany(companyName) != -1) { Console.WriteLine("That name is already being used."); } } while (companyName.equals("")) getIndexOfCompany(companyName) != -1; </pre> Alternative answer <pre> public void addCompany() { int balance, x, y; String companyName, typeOfCompany; Console.write("Enter a name for the company: "); companyName = Console.ReadLine(); while (companyName.equals("")) getIndexOfCompany(companyName) != -1) { if (companyName.equals("")) { Console.WriteLine("You must enter a name."); } else if (getIndexOfCompany(companyName) != -1) { Console.WriteLine("That name is already being used."); } } </pre>	5

		<pre> Console.write("Enter a name for the company: "); companyName = Console.readLine(); } </pre>	
11	1	<pre> class AffluentHousehold extends Household { public AffluentHousehold(int x, int y) { super(x, y); chanceEatOutPerDay = 1; } } public void addHousehold() { int x, y; int[] tempLocation = getRandomLocation(); x = tempLocation[0]; y = tempLocation[1]; if (x < 100) { AffluentHousehold temp = new AffluentHousehold(x, y); households.add(temp); } else { Household temp = new Household(x, y); households.add(temp); } } </pre>	7
12	1	From the simulation class <pre> public void modifyCompany(int index) { String choice; int outletIndex, x, y; boolean closeCompany; Console.WriteLine(System.lineSeparator() + "*****"); Console.WriteLine("***** MODIFY COMPANY *****"); Console.WriteLine("*****"); Console.WriteLine("1. Open new outlet"); Console.WriteLine("2. Close outlet"); Console.WriteLine("3. Expand outlet"); Console.WriteLine("4. Get Loan"); Console.WriteLine("5. Pay back loan"); Console.write(System.lineSeparator() + "Enter your choice: "); choice = Console.readLine(); Console.WriteLine(); if (choice.equals("2") choice.equals("3")) { ... } else if (choice.equals("1")) { ... } else if (choice.equals("4")) { Console.write("Enter the interest rate for the loan: "); float rate = Float.parseFloat(Console.readLine()); companies.get(index).getLoan(rate); } else if (choice.equals("5")) { Console.write("Enter the amount to pay back: "); float payBackAmount = Float.parseFloat(Console.readLine()); companies.get(index).payBackLoan(payBackAmount); } Console.WriteLine(); } </pre>	10

```
}

```

From the company class

```
class Company {
    protected String name, category;
    protected float balance, reputationScore, avgCostPerMeal,
    avgPricePerMeal, dailyCosts, familyOutletCost, fastFoodOutletCost,
    namedChefOutletCost, fuelCostPerUnit, baseCostOfDelivery,
loanBalance, interestRate;
    protected List<Outlet> outlets = new ArrayList();
    protected int familyFoodOutletCapacity, fastFoodOutletCapacity,
    namedChefOutletCapacity;
    private static Random rnd = new Random();
    ...

    public String processDayEnd() {
        ...
    }
    details += "Previous balance for company: " + balance +
    System.lineSeparator();
    balance += profitLossFromOutlets - dailyCosts - deliveryCosts -
(loanBalance * interestRate);
    details += "New balance for company: " + balance;
    return details;
}
```

```
public void getLoan(float rate)
{
    if (loanBalance == 0)
    {
        balance += 10000;
        loanBalance = 10000;
        interestRate = rate;
    }
}

public void payBackLoan(float amount)
{
    loanBalance -= amount;
    balance -= amount;
}
```

Alternative answer for taking a loan

In modifyCompany method in Simulation class

```
...
} else if (choice.equals("4")) {
    if (companies.get(index).getLoanBalance() <= 0) {
        Console.write("Enter the interest rate for the loan: ");
        float rate = Float.parseFloat(Console.readLine());
        companies.get(index).takeOutLoan(rate);
    }
}
```

		Methods in Company Class <pre> public float getLoanBalance() { return loanBalance; } public void takeOutLoan(float rate) { balance += 10000; loanBalance = 10000; interestRate = rate; } </pre>	
13	1	<pre> public List<Integer> getOrderedListOfOutlets() { List<Integer> orderedList = new ArrayList(); int nearestOutlet = 0; orderedList.add(nearestOutlet); while (orderedList.size() < outlets.size()) { float shortestDistanceSoFar = 1000000f; for (int count = 1; count < outlets.size(); count++) { if (!orderedList.contains(count)) { float temp = getDistanceBetweenTwoOutlets(orderedList.get(orderedList.size()-1), count); if (temp < shortestDistanceSoFar) { nearestOutlet = count; shortestDistanceSoFar = temp; } } } orderedList.add(nearestOutlet); } return orderedList; } public float calculateDeliveryCost() { List<Integer> listOfOutlets = getOrderedListOfOutlets(); float totalDistance = 0; float totalCost; for (int current = 0; current < listOfOutlets.size() - 1; current++) { totalDistance += getDistanceBetweenTwoOutlets(listOfOutlets.get(current), listOfOutlets.get(current + 1)); } totalCost = totalDistance * fuelCostPerUnit; return totalCost; } </pre>	11