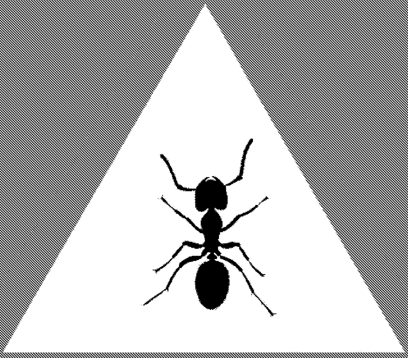
Computer Science

A Level | AQA | 7517

2015 specification
for the 2026 exam



Ant Simulation

PAPER 1 EXAM RESOURCE PACK 2026

for A Level AQA Computer Science

C# EDITION

- DIGITAL RESOURCE -

This pack includes paper versions of the electronic files.

Go to **zzed.uk/ProductSupport** to download the electronic files.

zigzageducation.co.uk

POD
12893

Publish your own work... Write to a brief...
Register at **publishmenow.co.uk**

Follow us on Bluesky or X **@ZigZagComputing**

Photocopiable/digital resources may only be copied by the purchasing institution on a single site and for their own use

Contents

Printouts of electronic resources (for reference)

** The electronic PDF versions of these files are password-protected, so that students can only access them with your permission. Passwords can be found in the Teacher's Introduction on page iv.*

*** Note there are also electronic copies of the UML Diagrams ('Complete' & 'Activity' versions) provided.*

Teacher's Introduction

The resource pack consists of the following sections:

- **Code breakdown**: a detailed technical overview of the Skeleton Program, describing in detail each class and method in turn – including their purpose/function, parameters and return values. Note that this is intended as a helpful reference document only, and not as a substitute for exploring the code in a practical manner. There is also a 'Grid Example' file which can be used as a printout to help students to understand the simulation.
- **UML class diagram activity**: requires you to study the program and fill in the gaps with the missing class/method names, data types, associations and access levels.
- **Video**: a quick overview of the ***Ant Simulation*** application mechanics – intended as a visual aid to accompany the notes in the official AQA pre-release material.
- **Theory questions**: designed to test your understanding of the Skeleton Program. These questions require access to the program, but no modifications need to be made to the program. Write-on (with answer lines) and non-write-on versions are available.
- **Coding tasks**: there are 24 modification tasks to test your programming skills – as well as an additional 15 modification ideas that you may also want to try as extension tasks.
- **Solutions / Mark schemes**: included for UML class diagram activity, theory questions, and coding tasks.

This resource is intended to supplement your teaching only. **Please read full disclaimer (p. iii) before using it.**

DIGITAL RESOURCE

Once you have downloaded the files for this resource via (**zzed.uk/ProductSupport**) you will have access to the following:

🗀 `AntSimulation`	this folder contains all of the content (PDF/DOCX) accessible via a HTML interface
🗎 `Passwords.txt`	for teacher use – this file contains all of the passwords for the protected PDFs (also listed below)

* PRINTED COPIES OF ALL THE MATERIALS IN THIS DIGITAL RESOURCE PACK ARE INCLUDED FOR REFERENCE.

Installation: Extract the files from the downloaded ZIP file and move the entire `TargetClear` folder onto a network location that is accessible for students, and provide them with a shortcut to the index.html file. All content can be accessed from this page.

Passwords: All of the PDFs accessible via the *Solutions* web page are password-protected, so that students can only access them with your permission. Each password is a four-digit code, as follows:

This pack is based on Version 2 of the Ant Simulation code released by AQA on 21/11/25.
Please ensure you are using Version 2 of the code from Centre Services.

Ant Simulation

Ant Simulation represents a small world of ants showing their interaction with nests, food and pheromone trails.

The user can experiment with four different simulations, each representing different configurations of the small world. The world is represented by either a 5×5 or 10×10 grid. Simulations 1–3 contain a single ant nest, and simulation 4 contains two ant nests. At the start of the simulation, each nest contains a single queen ant and multiple worker ants (configured by parameters when the application starts up).

The application uses the concept of stages to advance, each stage essentially being a snapshot of the current state of the simulation world. As the user advances the stages, the entities within the simulation world, be those ants, pheromones or the nests themselves, all advance one stage at a time, each performing one of their preset operations. Examples of these operations are move, pick up food, follow a pheromone trail, or deposit food at the nest.

Initially, worker ants move at random around the simulation world, moving to one of their neighbouring cells per stage. If an ant finds a cell containing food, the ant picks up a small amount of that food, and then as the stages are advanced by the user, makes its way directly back to the nest. As it moves back towards the nest, the worker ant lays down a pheromone trail. If other worker ants are near to that pheromone trail, they will follow it to the food source to also collect food and take it back to the nest. This represents how ants behave in the real world.

The application also represents the relationship between food available at a nest and ant population. At each stage, if the food in a nest gets too low, or the population gets too high for the food available, some of the ants belonging to the nest are culled.

When the nest has plentiful food levels, however, the queen ant will give birth to new worker ants and even on a very small chance, another queen.

During operation, the user can display the simulation world in its entirety, view part of it, or even view an individual cell within the world to see it in detail.

The objective of the application is to experiment with different configurations of the simulation world, advancing the simulation stage by stage to see how the ants interact with each other, the food around them, and the nest.

This resource aims to help you get to grips with and prepare for the A Level Paper 1 examination for summer 2026, which is partly based on the ***Ant Simulation*** pre-release material.

Ant Simulat

Skeleton Code Breakdo

Static Methods

Identifier / Data		Description
GetCellReference		
Parameters	Row : Int (passed by ref) Column : Int (passed by ref)	This method asks the user for the row and c are cast to integers and assigned to the vari reference rather than by value, they do not r
Return values		
GetChoice		
Parameters	n/a	This method assigns a string input from the
Return values	Choice : String	
DisplayMenu		
Parameters	n/a	This method displays the main program mer
Return values	n/a	
Main		
Parameters	n/a	This method sets up the simulation. The sim [illegible]er [illegible] The method asks for user input t which configuration list to use when instanti The method then enters the main program l options menu. The method then sets the Ch ranging from 1 to 4. This is used to call diffe If the user selects 9 from the GetChoice() m
Return values	n/a	

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Class: Simulation

Identifier / Data		Description
<<constructor>>		
Parameters	SimulationParameters : Int List	Initialises the following protected at SimulationParameters • StartingNumberOfNests to th • NumberOfRows to the value i • NumberOfColumns to the val • StartingFoodInNest to the val • StartingNumberOfFoodCells • StartAntsInNest to the value i • NewPheromoneStrength to th • PheromoneDecay to the value The method then initialises the two nested loop to populate the list Gri Each cell is given a Row and Colu of this are from 1,1 to NumberOfR The method then calls the SetUpA which is the default Row,Column p The method then uses a For…Next positions. The loop is governed by t the variable Count, which is initialis to 1, the loop does not run, which m The loop generates random values NumberOfRows + 1, or 1 to Numb simulation "world". If the random po present, the program loops to selec chosen, the method calls the SetU The method then enters another Fo bound of StartingNumberOfFoodC "world". Food cells cannot be addec to repeatedly select random locatio AddFoodToCell() method to add a method does not check if food is al added multiple times to the same c
Return values	n/a	

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Identifier / Data		Description
AddFoodToNest (public)		
Parameters	Food : Int Row : Int Column : Int	This method uses a Foreach loop to food to a nest. If a Nest is at the sa the method calls the ChangeFood(
Return values	n/a	
AddFoodToCell (public)		
Parameters	Row : Int Column : Int Quantity : Int	This method uses the GetIndex() m its Row and Column position in the method for the cell at that index and that cell. The skeleton program ofte amount of food in a cell.
Return values	n/a	
AdvanceStage (public)		
Parameters	NumberOfStages : Int	This method contains a For…Next lo simulation from one stage to the nex Inside the loop, the method first insta PheromonesToDelete. It then itera calling the AdvanceStage() method the GetStrength() method to query reached 0, the Pheromone is adde Once the loop has completed, the m PheromonesToDelete list, removin list. This is because you can't remov with a Foreach loop. Once the loop has completed, the m the AdvanceStage method on each cell object called TempCell as a ref investigated. The method then queri method calls the AddFoodToNest() together with its row and location. Th UpdateFoodCarried() method is ca carrying, which will reduce the amou
Return values	n/a	

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Identifier / Data		Description
		If the ant, however, is in a cell which a food capacity greater than zero, th capacity of the ant. AQA updated thi *(derived from the Ant class)* with a fo If the ant can pick up food, the method greater than what is available in the c calls the **UpdateFoodInCell()** on th amount taken by the ant. Finally, it c food to the ant to carry back to the n If the ant is not in a food cell, the cod its way back to the nest. If so, the m ant to reapply or generate new pher statement then calls the **ChooseCel** **GetIndicesOfNeighbours()** and **G** controls whether the ant is moving r The final task in the method is to iter each **Nest** in the **Nests** list.
GetAreaDetails (public)		
Parameters	**StartRow** : Int **StartColumn** : Int **EndRow** : Int **EndColumn** : Int	Uses nested iteration to enumerate the top left and bottom right corners The method uses concatenation to in the **Details** string represents a si called **TempCell** as a reference of th then uses the **GetNestInCell()**, **Get** and **GetAmountOfFood()** helper m Once the loop has completed, the D
Return values	**Details** : String	

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED

Identifier / Data		Description
GetCellDetails (public)		
Parameters	Row : Int Column : Int	This method is used to build up a m The method instantiates a local cell at the location being investigated. It f instantiates a copy of the Nest in tha GetFoodLevel() method on that cell The method then uses the same tec method, passing in the CurrentCell find out how many ants are in that c uses the GetNumberOfPheromone and how strong they are. Finally, the method adds some carri read and returns the string.
Return values	Details : String	
GetDetails (public)		
Parameters	n/a	Uses nested iteration to enumerate The method uses concatenation to in the Details string represents a si called TempCell as a reference of th then uses the GetNestInCell(), Get and GetAmountOfFood() helper m Once the loop has completed, the D
Return values	Details : String	
GetIndex (private)		
Parameters	Row : Int Column : Int	Uses the Row and Column paramet
Return values	Int	

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Identifier / Data		Description
GetIndexOfNeighbourWithStrongestPheromone (private)		
Parameters	Row : Int Column : Int	This method first instantiates the fol • StrongestPheromone to 0 • IndexOfStrongestPheromon It then uses a Foreach loop to iterat cell at position Row, Column. This method. If a neighbour is not valid (it is represented as -1. In the Foreac if the strength of the pheromone in t If so, the method updates the Index what the new StrongestPheromon After the loop has completed, the In contain any pheromones, the Index continue moving randomly.
Return values	IndexOfStrongestPheromone : Int	
GetIndicesOfNeighbours (private)		
Parameters	Row : Int Column : Int	This method builds up and returns a Grid indices of the neighbouring cell at the position Row, Column. The n nested iteration to look at the cells i around the cell given at position Ro For example, if Row was 3 and Col in a standard 5×5 simulation "world", method would return a list containing [5,6,7,10,11,12,15,16,17]. If a cell is outside the bounds of the "world", or if the cell being investiga the index is given as -1.
Return values	ListOfNeighbours : Int List	
GetNestInCell (public)		
Parameters	C : Cell	This method iterates through the Ne parameter C. If they match, the Nes the method gives a null return.
Return values	N : Nest	

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Identifier / Data		Description
GetNumberOfAntsInCell (public)		
Parameters	C: Cell	This method initialises an integer var comparing the location of the Ant wi variable is incremented. After the loo
Return values	Count : Int	
GetNumberOfPheromonesInCell (public)		
Parameters	C: Cell	This method initialises an integer var comparing the location of the Phero Count variable is incremented. After
Return values	Count : Int	
GetStrongestPheromoneInCell (private)		
Parameters	C: Cell	This method initialises an integer var list comparing the location of the Ph and if the Count variable and the str value in the variable Strongest, the the Pheromone. After the loop, Str
Return values	Strongest : Int	
SetUpANestAt (public)		
Parameters	Row : Int Column : Int	This method first adds a new Nest o the Nests list. It then adds a new Qu Ants list. Finally, it uses a For…Next has a lower bound of 2 and the uppe the StartingAntsInNest must also i
Return values	n/a	
UpdateAntsPheromoneInCell (public)		
Parameters	A : Ant	This method iterates through the Ph parameter A, and it has the same ID the UpdateStrength() method on th parameter, and then exits the metho list without finding a match, it means original path; therefore, the method i the Pheromone list.
Return values	n/a	

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Class: Cell

Identifier / Data		Description
<<constructor>>		
Parameters	StartRow : Int StartColumn : Int	The constructor passes the parame It then sets the protected attribute A
Return values	n/a	
GetAmountOfFood (public)		
Parameters		This is an accessor method which r
Return values	AmountOfFood : Int	
GetDetails (public) <<override>>		
Parameters	n/a	This method is only called from the comprises the return value from the AmountOfFood in the cell. The Ge empty string.
Return values	String	
UpdateFoodInCell (public)		
Parameters	Change : Int	This is a mutator method which upda Change to it.
Return values	n/a	

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Class: *Ant*

Identifier / Data		Description
<<constructor>>		
Parameters	StartRow : Int StartColumn : Int NestInRow : Int NestInColumn : Int	The constructor passes the parame It then sets the following protected a • NestRow to the parameter Nes • NestColumn to the parameter • ID to the static attribute NextA • Stages to 0 • AmountOfFoodCarried to 0 • FoodCapacity to 0 • TypeOfAnt to “” Many of these attributes are update The constructor also increments the new ID number.
Return values	n/a	
AdvanceStage (public) <<virtual>>		
Parameters	Nests : Nest List Ants : Ant List Pheromones : Pheromone List	This method increments the Stages
Return values	n/a	

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Identifier / Data		Description
ChangeCell (protected)		
Parameters	NewCellIndicator : Int RowToChange : Int (passed by reference) ColumnToChange : Int (passed by reference)	This method updates the position of t on the NewCellIndicator parameter the right below the location of neighb around the ant. The NewCellIndicat indicates position. If the parameter is gre then the new direction must be down therefore, the method increments the RowToChange parameter to move t downwards. If the NewCellIndicator 3, the new direction must be upwards the method decrements the RowToC If the NewCellIndicator is 0, 3 or 6, the ColumnToChange parameter to direction must be right; therefore, the the ant right. A combination of these s table shown above. RowToChange and ColumnToCha be returned.
Return values	n/a	
ChooseCellToMoveTo (public) <<virtual>>		
Parameters	ListOfNeighbours : Int List IndexOfNeighbourWithStrongestPheromone : int	This is an empty base class virtual m
Return values	n/a	
ChooseRandomNeighbour (protected)		
Parameters	ListOfNeighbours : Int List	This method chooses the index of a AdvanceStage method in the Simu for food. It uses a loop to select one while the neighbour does not contai location of the ant, or it is a location
Return values	Int	
GetDetails (public) <<override>>		
Parameters	n/a	This method is only called from the which comprises the return value fr ID of the Ant, the TypeOfAnt, and GetDetails() method in the base cl
Return values	String	

INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Identifier / Data		Description
GetFoodCapacity (public) <<override>>		
Parameters	n/a	This is an accessor method which r
Return values	Int	
GetFoodCarried (public) <<virtual>>		
Parameters	n/a	This is an accessor method which r
Return values	Int	
GetNestCol (public) <<virtual>>		
Parameters	n/a	This is an accessor method which r
Return values	Int	
GetNestRow (public) <<virtual>>		
Parameters	n/a	This is an accessor method which r
Return values	Int	
GetTypeOfAnt (public) <<virtual>>		
Parameters	n/a	This is an accessor method which r
Return values	String	
IsAtOwnNest (public) <<virtual>>		
Parameters	n/a	This returns the result of an expressi attribute NestRow and if the current
Return values	Bool	
UpdateFood (public) <<virtual>>		
Parameters	Change : Int	This is a mutator method which upda parameter Change to it.
Return values	n/a	

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Class: Pheromone

Identifier / Data		Description
<<constructor>>		
Parameters	Row : Int Column : Int BelongsToAnt : Int InitialStrength : Int Decay : Int	The constructor passes the parame It then sets the following protected a • BelongsTo to the parameter B • Strength to the parameter Initi • PheromoneDecay to the para
Return values	n/a	
AdvanceStage (public) <<override>>		
Parameters	Nests : Nest List Ants : Ant List Pheromones : Pheromone List	This method reduces the Strength It then checks the updated Strength
Return values	n/a	
GetBelongsTo (public)		
Parameters	n/a	This is an accessor method which r
Return values	Int	
GetStrength (public)		
Parameters	n/a	This is an accessor method which r
Return values	Int	
UpdateStrength (public)		
Parameters	Change : Int	This is a mutator method which upda
Return values	n/a	

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Class: Nest

Identifier / Data		Description
<<constructor>>		
Parameters	StartRow : Int StartColumn : Int StartFood : Int	The constructor passes the parame It then sets the following protected a • FoodLevel to the parameter St • NumberOfQueens to 1 • ID to the static attribute NextNe The constructor also increments the a new ID number.
Return values	n/a	
AdvanceStage (public) <<override>>		
Parameters	Nests : Nest List Ants : Ant List Pheromones : Pheromone List	This method performs a number of It first checks to see if the Ants list exits and the nest doesn't advance. The method then initialises three inte setting them all to 0. It then iterates t Note that this DOESN'T check if the Queen ant, the Count variable is inc worker ant, the Count variable is inc This is perhaps slightly misleading a would have been more descriptive. around. The preliminary material *stage by an amount determined by t* Count variable is incremented for e "world". After looping through the A inverse of the Count variable which The method then performs a numbe in the nest is 0 and the AntsInNest *simulation belonging to this nest, bu* incremented. If the FoodLevel of th this nest, the AntsToCull variable i
Return values	n/a	

INSPECTION COPY

COPYRIGHT PROTECTED

Identifier / Data		Description
		The method then performs a wider c number of worker ants belonging to incremented. Within this scenario, t greater than the number of worker a the application from attempting to re The method then uses a For…Next removing ants from the **Ants** list. Th **Ants** list. The loop will not select an however, the ant selected could be be in the nest. This means that any located in a food cell or carrying foo the **NumberOfQueens** variable is d If the **FoodLevel** is not less than fiv is deemed to be healthy, and theref there is a 50% chance that the Que will be a new **Queen**. Combined, th a new **Queen** is added to the simula ants are added to the **Ants** list.
ChangeFood (public)		
Parameters	**Change** : Int	This method adjusts the **FoodLeve** It then checks the updated **FoodLe** back to 0.
Return values	n/a	
GetFoodLevel (public)		
Parameters	n/a	This is an accessor method which r
Return values	Int	

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Class: Entity

Identifier / Data		Description
<<constructor>>		
Parameters	StartRow : Int StartColumn : Int	The constructor sets the following pr • Row to the parameter StartRo • Column to the parameter Star Both of these attributes are updated
Return values	n/a	
AdvanceStage (public) <<virtual>>		
Parameters	Nests : Nest List Ants : Ant List Pheromones : Pheromone List	This is an empty base class virtual
Return values	n/a	
GetColumn (public)		
Parameters	n/a	This is an accessor method which r
Return values	Int	
GetDetails (public) <<virtual>>		
Parameters	n/a	This is a base class virtual method w
Return values	String	
GetID (public)		
Parameters	n/a	This is an accessor method which r
Return values	Int	
GetRow (public)		
Parameters	n/a	This is an accessor method which r
Return values	Int	
InSameLocation (public)		
Parameters	E : Entity	This returns the result of an expressi parameter E and the attribute Colum entities are in the same place).
Return values	Bool	

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Class: Queen

Identifier / Data		Description
<<constructor>>		
Parameters	StartRow : Int StartColumn : Int NestInRow : Int NestInColumn : Int	The constructor passes all four parame It then sets the TypeOfAnt attribute (fr
Return values	n/a	

Class: WorkerAnt

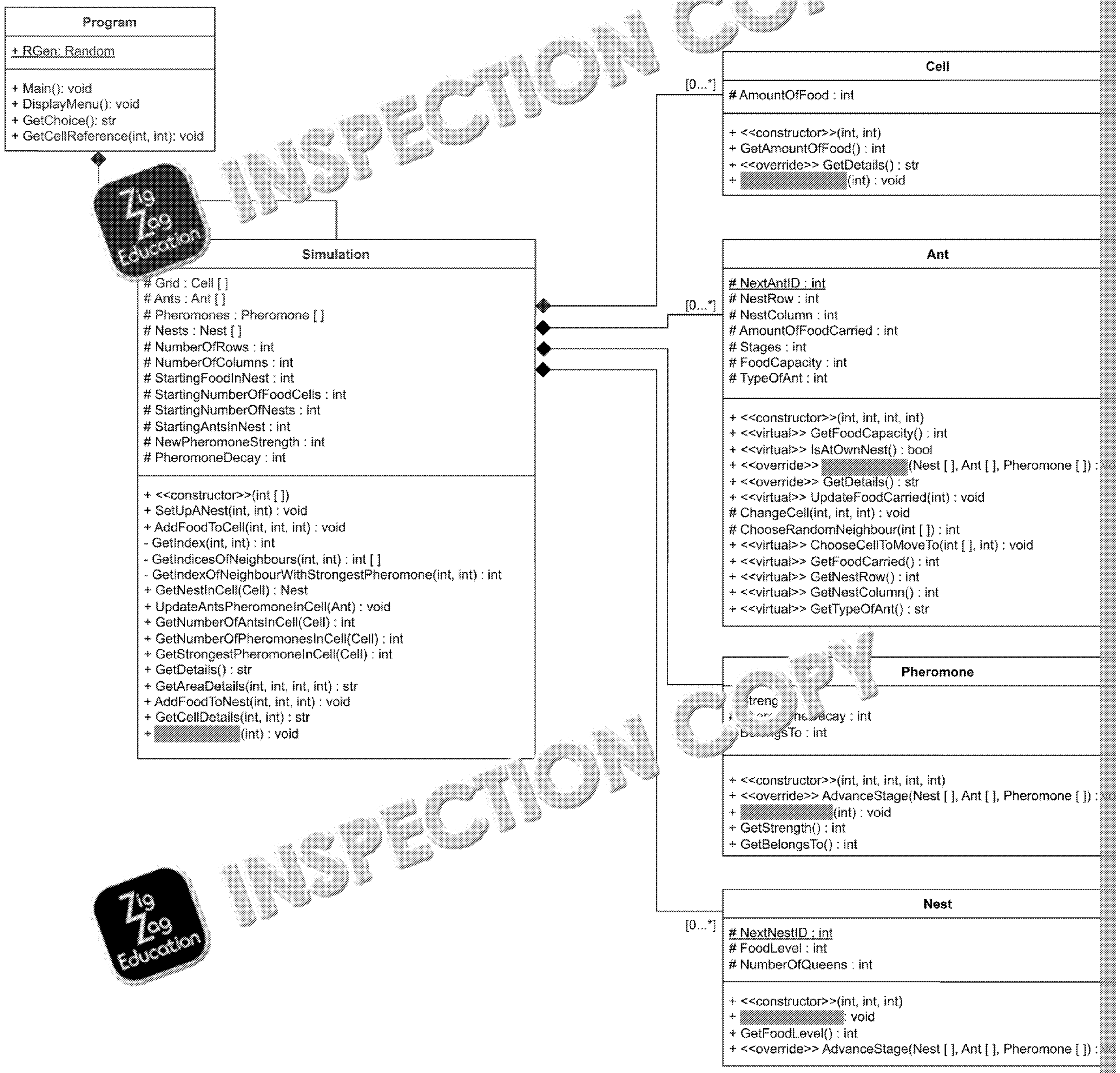
Identifier / Data		Description
<<constructor>>		
Parameters	StartRow : Int StartColumn : Int NestInRow : Int NestInColumn : Int	The constructor passes all four parame It then sets the TypeOfAnt attribute (fr (from the base class) to 30.
Return values	n/a	
ChooseCellToMoveTo (public) <<override>>		
Parameters	ListOfNeighbours : Int List IndexOfNeighbourWithStrongestPheromone : Int	This method decides how a worker ant carrying any food. If it is, the method us worker ant are less than or greater than Column attributes (from the base class cell per stage back towards the nest. Alternatively, if the worker ant is not car IndexOfNeighbourWithStrongestPhe currently searching for food and there a contain pheromone. In this case the me in the ListOfNeighbours parameter to ChangeCell() method using this index t If the IndexOfNeighbourWithStrongest cell or cells with pheromone in them, whi method in the built-in List class to return t IndexOfNeighbourWithStrongestPhero ChangeCell() method using this index to
Return values	n/a	
GetDetails (public) <<override>>		
Parameters	n/a	This method is only called from the Get comprises the return value from the Ge AmountOfFoodCarried of the Worker
Return values	String	

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Ant Simulation



Program

+ RGen: Random

+ Main(): void
+ DisplayMenu(): void
+ GetChoice(): str
+ GetCellReference(int, int): void

Simulation

Grid : Cell [ ]
Ants : Ant [ ]
Pheromones : Pheromone [ ]
Nests : Nest [ ]
NumberOfRows : int
NumberOfColumns : int
StartingFoodInNest : int
StartingNumberOfFoodCells : int
StartingNumberOfNests : int
StartingAntsInNest : int
NewPheromoneStrength : int
PheromoneDecay : int

+ <<constructor>>(int [ ])
+ SetUpANest(int, int) : void
+ AddFoodToCell(int, int, int) : void
- GetIndex(int, int) : int
- GetIndicesOfNeighbours(int, int) : int [ ]
- GetIndexOfNeighbourWithStrongestPheromone(int, int) : int
+ GetNestInCell(Cell) : Nest
+ UpdateAntsPheromoneInCell(Ant) : void
+ GetNumberOfAntsInCell(Cell) : int
+ GetNumberOfPheromonesInCell(Cell) : int
+ GetStrongestPheromoneInCell(Cell) : int
+ GetDetails() : str
+ GetAreaDetails(int, int, int, int) : str
+ AddFoodToNest(int, int, int) : void
+ GetCellDetails(int, int) : str
+ [redacted](int) : void

Cell

AmountOfFood : int

+ <<constructor>>(int, int)
+ GetAmountOfFood() : int
+ <<override>> GetDetails() : str
+ [redacted](int) : void

Ant

NextAntID : int
NestRow : int
NestColumn : int
AmountOfFoodCarried : int
Stages : int
FoodCapacity : int
TypeOfAnt : int

+ <<constructor>>(int, int, int, int)
+ <<virtual>> GetFoodCapacity() : int
+ <<virtual>> IsAtOwnNest() : bool
+ <<override>> [redacted](Nest [ ], Ant [ ], Pheromone [ ]) : vo
+ <<override>> GetDetails() : str
+ <<virtual>> UpdateFoodCarried(int) : void
ChangeCell(int, int, int) : void
ChooseRandomNeighbour(int [ ]) : int
+ <<virtual>> ChooseCellToMoveTo(int [ ], int) : void
+ <<virtual>> GetFoodCarried() : int
+ <<virtual>> GetNestRow() : int
+ <<virtual>> GetNestColumn() : int
+ <<virtual>> GetTypeOfAnt() : str

Pheromone

Strength : int
PheromoneDecay : int
BelongsTo : int

+ <<constructor>>(int, int, int, int, int)
+ <<override>> AdvanceStage(Nest [ ], Ant [ ], Pheromone [ ]) : vo
+ [redacted](int) : void
+ GetStrength() : int
+ GetBelongsTo() : int

Nest

NextNestID : int
FoodLevel : int
NumberOfQueens : int

+ <<constructor>>(int, int, int)
+ [redacted]: void
+ GetFoodLevel() : int
+ <<override>> AdvanceStage(Nest [ ], Ant [ ], Pheromone [ ]) : vo

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Simulation 1

	1	2	3	4	5
1	*Grid Index 0*	*Grid Index 1* **Food:** [illegible]	*Grid Index* [illegible]	*Index 3*	*Grid Index 4*
2	*Grid Index 5*	*Grid Index 6*	*Grid Index 7*	*Grid Index 8* **Nest** **Ants: 5**	*Grid Index 9* **Food:** **500**
3	*Grid Index 10* **Food:** **500**	*Grid Index 11*	*Grid Index 12*	*Grid Index 13*	*Grid Index 14*
4	*Grid Index 15*	*Grid Index 16*	*Grid Index 17*	*Grid Index 18*	*Grid Index 1*[illegible]
5	*Grid Index 20*	*Grid Index 2*[illegible]	*Grid Index 22*	*Grid Index 23*	*Grid Index 24*

Starting N of Ne
Number
Numb Colu
Starting A of Food i
Starting N of Food
Starting N of Ants i
Strength Pherom
Pheron Decay

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Ant Simulati

C# Theory Questions

These questions refer to the **Preliminary Material** and the **Sk**
but **do not** require any additional programmin

TOTAL MARKS: 54

INSPECTION COPY

1. The Main procedure in the pre-release program asks the user to enter of those is **the simulation number**.

   Currently the program assumes the user will always enter valid numer 1–4 inclusive.

   (a) Explain why this program is not robust when processing user input

   (b) The code contains a prompt on line 20: **Enter simulation number**

   What error would occur during runtime if the user typed "four" whe from line 20?

   (c) How would a programmer resolve this issue?

2. Give an identifier used for:

   (a) A local variable used within the **GetCellReference** procedure.

   (b) A class variable.

   (c) The loop counter variable used when nests are being created wit **Simulation** class.

   (d) A variable that is used to store the neighbouring cell indices as a li

3. This program makes use of **Encapsulation**.

   (a) Explain what encapsulation means in object-oriented programmin

   (b) Describe how this concept is used within the **Cell** class.

4. This program makes use of **Inheritance**.

   (a) Explain what inheritance means in object-oriented programming.

   (b) Give an advantage of using this technique.

   (c) Explain how inheritance is used between the Entity, Ant, and Wor

5. What data structure is used to represent the grid of cells, and how mar contain in Simulation 3?

6. Describe how the **AdvanceStage** methods demonstrate polymorphism

7. Explain how the simulation demonstrates the use of abstraction by mo advantage of using this technique to simulate the behaviour of ant colo

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Ant Simulati

C# Theory Questions

These questions refer to the **Preliminary Material** and the **Sk**
but **do not** require any additional programmin

TOTAL MARKS: 54

INSPECTION COPY

1. The **Main** procedure in the provided program asks the user to enter
of those is **the simulation number**.

Currently the program assumes the user will always enter valid numer
1–4 inclusive.

(a) Explain why this program is not robust when processing user input

....

....

....

(b) The code contains a prompt on line 20: **Enter simulation numbe**

What error would occur during runtime if the user typed “four” whe
from line 20?

....

....

(c) How would a programmer resolve this issue?

....

....

2. Give an identifier used for:

(a) A local variable used within the **GetCellReference** procedure.

....

....

INSPECTION COPY

(b) A class variable.

....

....

(c) The loop counter variable used when nests are being created wit
Simulation class.

....

....

QUESTION 2 CONTINUES OVERLEAF

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Ant Simulati

INSPECTION COPY

Programming Tasks

These questions require you to load the **Skeleton Program** and to make

Note that any alternative or additional code changes that are deemed appropriate ensuring that it is clear where in the Skeleton Program those chang

The objective of this resource is to provide you with a selection of different que questions. Some questions are more prescriptive than others in how the task sh range of levels. Questions which have a similar theme may use different techn options on how to solve problems. Questions in this resource are not necessarily p

Unless specified by the question, the solutions assume valid input and therefore beyond that supplied by the original pre-release material f

Students are recommended to start with a clean copy of the pre-release code questions in this resource. This will prevent modifications made for one question different question.

These questions are based on Version 2 of the Ant Simulation code released by you are using Version 2 of the code from Centre Serv

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Task 1

This question extends the Skeleton Program to confirm that the user has s simulation. Modify the application to confirm that the user would like to end when they input option 9 from the main menu. On confirmation, the progra the simulation has ended.

What you need to do

Task 1.1

Update the **Main** method. When the user selects option 9 (quit), confirm th quit the simulation as usual and display that the simulation has ended.

Task 1.2

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter 1 to start simulation 1.
- Enter 9 to quit the simulation.
- Show the program correctly displaying a check to confirm that the use
- When prompted confirm to quit the simulation
- Show the program displaying a message confirming that the simulatio

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE showing the modifications to the M
- SCREEN CAPTURE(S) showing the required tests.

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Task 2

This question extends the **Ant** class and the **Nest** class to show a worker a reaches the age of 14 (stages). Modify the **Ant** class to include a getter fo **Stages** to ask an ant its age. Modify the **AdvanceStage** method in the Ne from the **Ants** list which are older than 14 stages. This should happen at th method. Confirm to the user as each old worker ant dies.

What you need to do

Task 2.1

Update the **Ant** class to include a new method which exposes the protecte **AdvanceStage** method in the **Nest** class to check if an ant is older than 14 the **Ants** list and confirm to the user that the ant has died of old age.

Task 2.2

Test that the changes you have made work:

- Run the Skeleton Program.
- Enter `1` to start simulation 1.
- Enter `1` to display five ants in the nest at the start of the simulation.
- Enter `5` followed by `15` to move the simulation on 15 stages.

Evidence that you need to provide:

- Your PROGRAM SOURCE CODE showing you exposing the protecte and modification to the **AdvanceStage** method in the **Nest** class.
- SCREEN CAPTURE(S) showing the required tests.

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Ant Simulati

Programming Tasks (Extensi

Extension 1

In the standard simulation, ants will continue to follow a pheromone trail to when that food source has been exhausted. Introduce a second pheromon to the Ant class. When an ant finds that the food source has less than 10% lay down the new pheromone rather than the standard pheromone as it ret "food source exhausted" pheromone should quadruple the decay level of th help erase the path more quickly.

Extension 2

In the standard simulation it is possible for the Queen ant to be culled at ra method into the Nest class which detects if a Queen has not been present promotes a normal WorkerAnt to a new Queen.

Extension 3

Introduce the concept of a soldier ant which can spray formic acid. Create a moves using the normal random movement around the simulation world. A should review all of its neighbouring cells. If the soldier ant finds an ant or a different nest, it should spray formic acid towards that cell, killing all of the a

Extension 4

Introduce a "message-passing" system whereby ants can leave messages heads as they move past each other. Modify the Cell class to include the c scale of 1–5 (1 being poor, 5 being highly nutritious). As ants pass each ot nest, the returning ant, carrying food, "bumps heads" with other ants to info food source. Modify the ChooseCellToMoveTo method to use this informa for deciding if an ant will follow the pheromone trail.

Extension 5

Introduce the concept that a nest can have a maximum of 30 ants belongin been reached, create a new Queen ant which leaves the nest and moves a simulation to create a new nest. Two nests cannot be in neighbouring cells.

Extension 6

Introduce two new options, 6 and 7, to the main menu to give the user the a simulation to a CSV file.

Extension 7

Implement a dictionary data structure into the WorkerAnt class which store source found and how much food there is. At each stage, use the strength to the food source to make a guess at how many ants have visited the food much food is being consumed per stage, and update the dictionary. Use thi ChooseCellToMoveTo method as additional weighting for selecting wheth pheromone trail back to a food source.

INSPECTION COPY

COPYRIGHT PROTECTED

Extension 8

Implement the A* algorithm for a worker ant to find the shortest path back t
Use Manhattan distance as the heuristic.

Extension 9

Modify the **Cell** class to allow different types of food – for example, food an
concept to the **Nest** class to monitor how much has been collected by the v
WorkerAnt class so that an ant won't collect a particular food source if ther
nest.

Extension 10

Introduce polydomous colonies which can have multiple nests. Track share
deposit food at nests within their colony rather than just at one specific nes
there is already plenty of food then it should go to another nest.

Extension 11

Modify the **Nest** class to create a new method called **ShareFoodWithNea**
one nest to take food to the ants in another nest if it is running low.

Extension 12

Implement Tandem Running by implementing new **LeadAnt** and **Follower**
been found, instantiate two ants in the nest: one "Lead" and one "Follow". T
pheromone, instead moving randomly, but has double the food-carrying ca
pheromone trail, moving slowly (one cell per two stages). At each stage it s
follower is within one cell distance. If not, it waits. The follower ant has a 90
the leader's current position. It stops if already adjacent. When both ants re
ant picks up the food and takes it back to the nest.

Extension 13

Introduce the concept of "most food per unit of time" (stage). Modify the **Ce**
allowing only three ants present at any time. Introduce a new **Bridge** class
cell which has a capacity of 10 ants present at any time. Add a new option
the user to place bridges into the simulation at any location except the nes
main menu to allow the user to place food into cells. Demonstrate ants ign
being placed in a cell at a location which can't be navigated to using bridge
amount of food that is easier to get to because overall the nest gains a gre
time.

Extension 14

Introduce the concept of "[illegible]" whereby a worker ant defects to a di
more food in it and therefore offers the ant a greater chance of survival.

Extension 15

Introduce the concept of "nutritional value" in a food source. Once a unit o
to the nest, it is assessed by the ants in the nest. If it is lower than other fo
at, a **ReviewerAnt** follows the pheromone trail out to the food source, rem

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Preview of Questions Ends Here

This is a limited inspection copy. Sample of questions ends here to avoid students previewing questions before they are set. See contents page for details of the rest of the resource.

Question		Suggested Solution
13		The numbers are not truly random; they are produced by a deterministic algorithm using a seed value **[1]** This means the same seed will always produce the same sequence of "random" numbers **[1]**
14	(a)	Dictionary lookups are O(1) on average, so finding pheromones at a cell would be faster than scanning the whole list **[1]** They provide direct access by coordinates rather than searching sequentially through a list **[1]**
	(b)	Dictionary keys are hashed to compute an index position. If a key is not hashable (mutable) its hash value could change, breaking dictionary lookups **[1]** Immutable types (tuples like (row, column)) are hashable and safe to use as keys **[1]**
15		In object-oriented programming, code is organised into classes which contain both data and behaviour. In a procedural program, these would instead be handled by separate functions and shared data structures, rather than being combined in a class. **[1]** Example, the **Ant** and **Nest** classes in this program store attributes like **Row** and **FoodLevel**, and contain methods such as **AdvanceStage** and **ChangeFood** **[1]** OOP uses encapsulation to protect data by keeping attributes private and controlling access through methods. In a procedural approach, this value might be stored in a global variable, meaning it could be accidentally changed by any part of the program. **[1]** For example, the nest's food level is stored in **FoodLevel** and can only be changed using the **ChangeFood** method **[1]**

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Task 20

Coding

- Add a suitable data structure to the WorkerAnt class which enables the ant to store its movement his
- Store the movement history of the worker ant from one stage to another. [1 mark]
- Override the normal worker ant movement behaviour to backtrack through the movement history co
- Reset the history if the worker ant finds food. [1 mark]

Teacher Notes

The LIFO structure of a stack lends itself very effectively to this question; however, it could be done with a solution uses a variable "ForagingStages" to keep track of the number of stages. This could also be do backtracking.

Example Solution

Modification of the WorkerAnt class to introduce a new data structure to store the movement history:

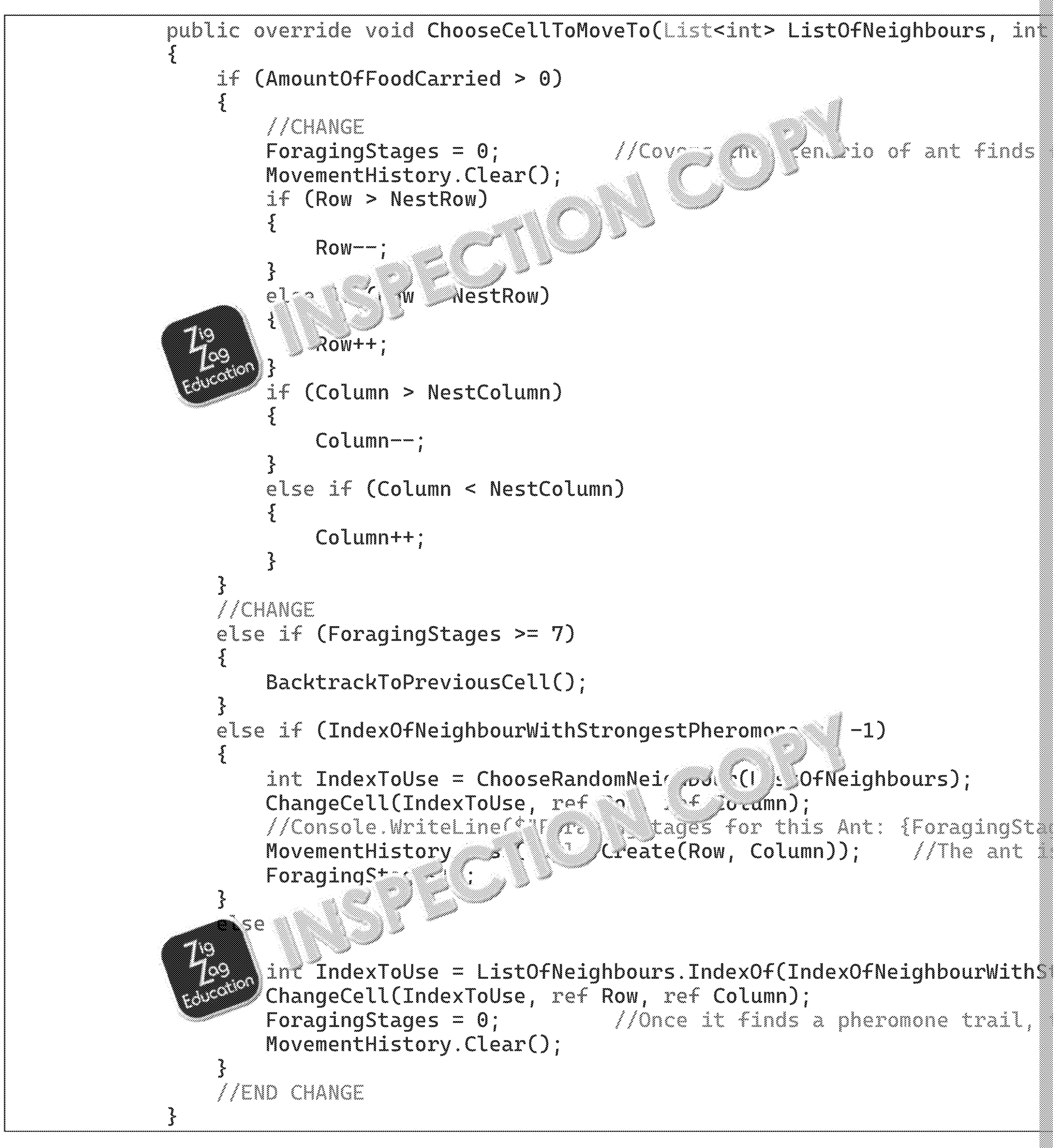
```
        class WorkerAnt : Ant
        {
            //CHANGE
            private Stack<Tuple<int, int>> MovementHistory;
            private int ForagingStages;
            private bool BackTracking = false;

            public WorkerAnt(int StartRow, int StartColumn, int NestInRow, int Nest
                : base(StartRow, StartColumn, NestInRow, NestInColumn)
            {
                TypeOfAnt = "worker";
                FoodCapacity = 30;
                MovementHistory = new Stack<Tuple<int, int>>();
                ForagingStages = 0;
            }
            //END CHANGE
```

INSPECTION COPY

COPYRIGHT PROTECTED

Modification of the ChooseCellToMoveTo method in the WorkerAnt class to push movement history ont

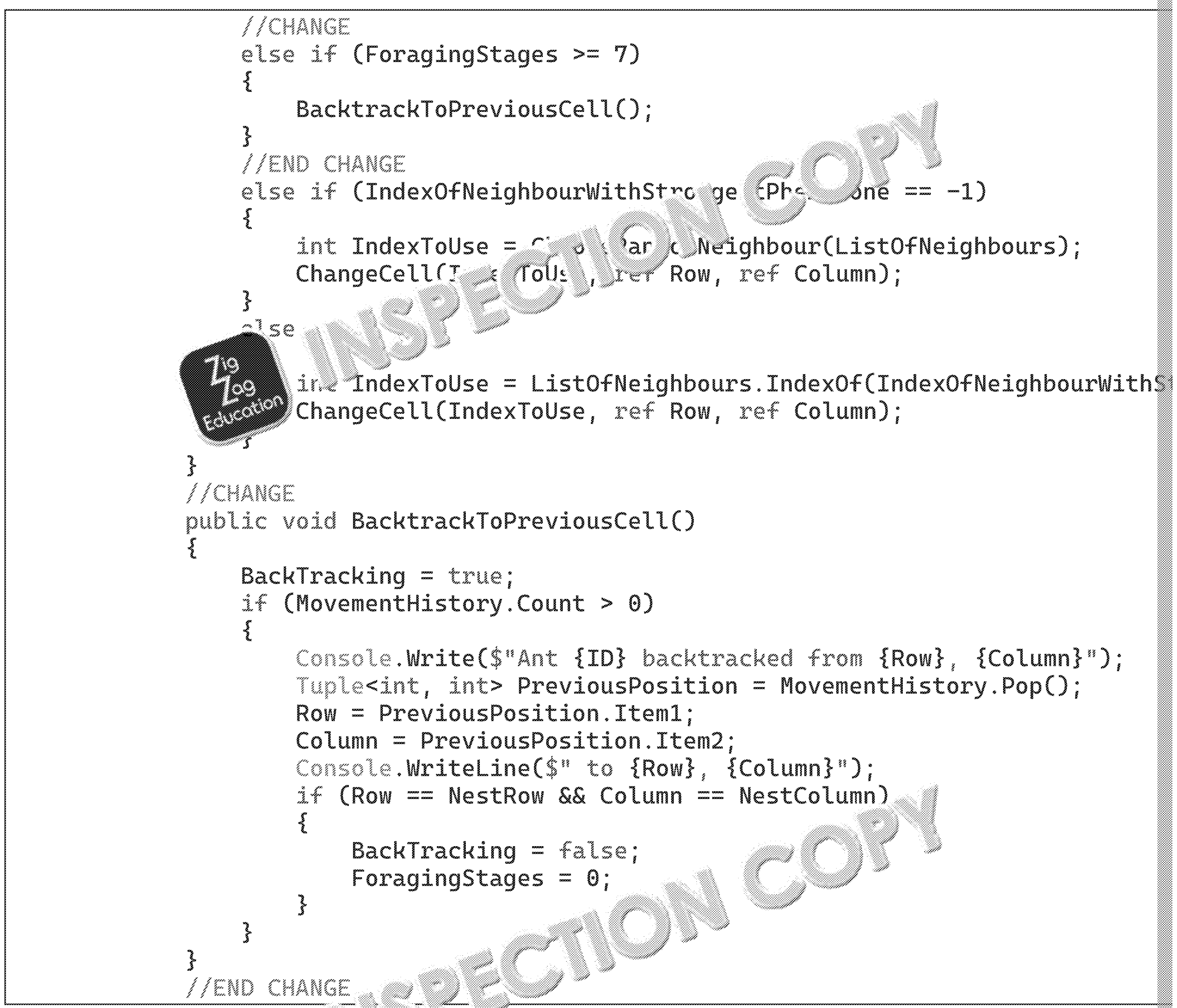


```
        public override void ChooseCellToMoveTo(List<int> ListOfNeighbours, int
        {
            if (AmountOfFoodCarried > 0)
            {
                //CHANGE
                ForagingStages = 0;         //Covers the scenario of ant finds
                MovementHistory.Clear();
                if (Row > NestRow)
                {
                    Row--;
                }
                else if (Row < NestRow)
                {
                    Row++;
                }
                if (Column > NestColumn)
                {
                    Column--;
                }
                else if (Column < NestColumn)
                {
                    Column++;
                }
            }
            //CHANGE
            else if (ForagingStages >= 7)
            {
                BacktrackToPreviousCell();
            }
            else if (IndexOfNeighbourWithStrongestPheromone == -1)
            {
                int IndexToUse = ChooseRandomNeighbour(ListOfNeighbours);
                ChangeCell(IndexToUse, ref Row, ref Column);
                //Console.WriteLine($"Foraging Stages for this Ant: {ForagingSta
                MovementHistory.Push(Tuple.Create(Row, Column));    //The ant i
                ForagingStages++;
            }
            else
            {
                int IndexToUse = ListOfNeighbours.IndexOf(IndexOfNeighbourWithSt
                ChangeCell(IndexToUse, ref Row, ref Column);
                ForagingStages = 0;         //Once it finds a pheromone trail,
                MovementHistory.Clear();
            }
            //END CHANGE
        }
```

INSPECTION COPY

COPYRIGHT PROTECTED

Creation of BacktrackToPreviousCell method which is called when ant reaches seven stages of mov

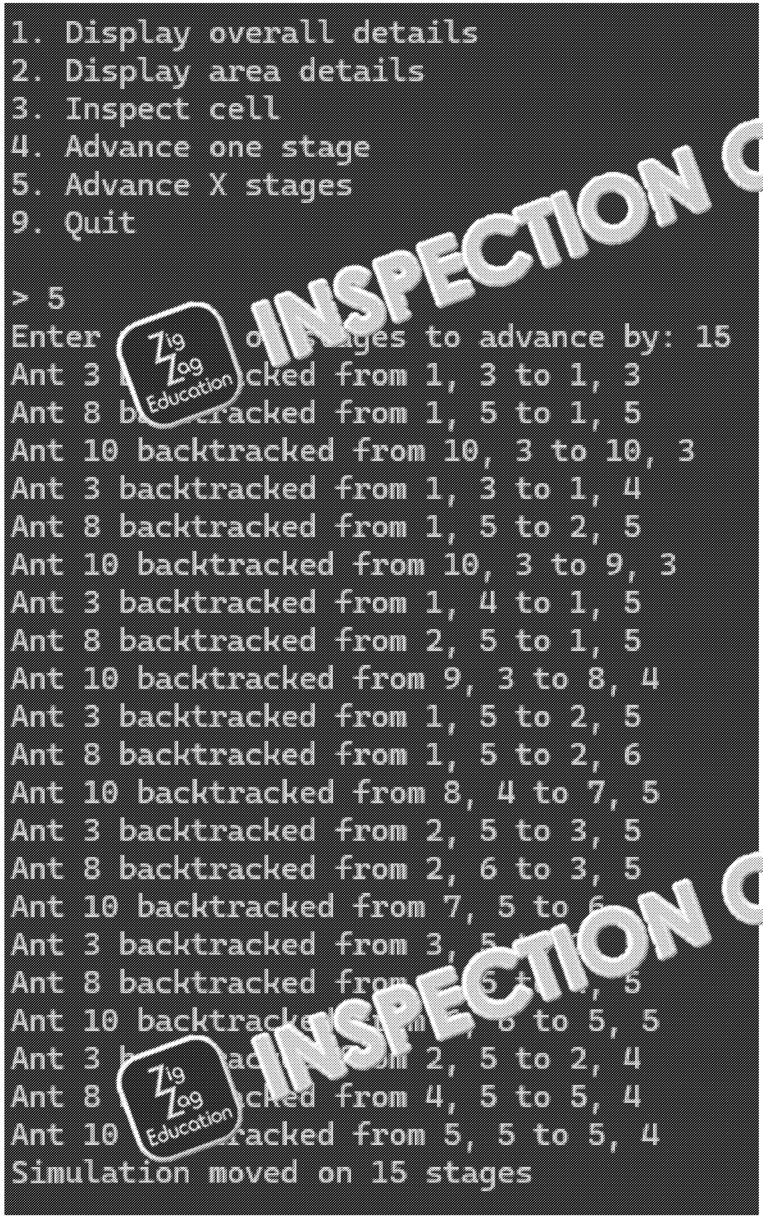


```
            //CHANGE
            else if (ForagingStages >= 7)
            {
                BacktrackToPreviousCell();
            }
            //END CHANGE
            else if (IndexOfNeighbourWithStrongestPheromone == -1)
            {
                int IndexToUse = ChooseRandomNeighbour(ListOfNeighbours);
                ChangeCell(IndexToUse, ref Row, ref Column);
            }
            else
            {
                int IndexToUse = ListOfNeighbours.IndexOf(IndexOfNeighbourWithSt
                ChangeCell(IndexToUse, ref Row, ref Column);
            }
        }
        //CHANGE
        public void BacktrackToPreviousCell()
        {
            BackTracking = true;
            if (MovementHistory.Count > 0)
            {
                Console.Write($"Ant {ID} backtracked from {Row}, {Column}");
                Tuple<int, int> PreviousPosition = MovementHistory.Pop();
                Row = PreviousPosition.Item1;
                Column = PreviousPosition.Item2;
                Console.WriteLine($" to {Row}, {Column}");
                if (Row == NestRow && Column == NestColumn)
                {
                    BackTracking = false;
                    ForagingStages = 0;
                }
            }
        }
        //END CHANGE
```

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Testing

- Show the program displaying backtracking route(s) of ants which have not successfully found food.



```
1. Display overall details
2. Display area details
3. Inspect cell
4. Advance one stage
5. Advance X stages
9. Quit

> 5
Enter number of stages to advance by: 15
Ant 3 backtracked from 1, 3 to 1, 3
Ant 8 backtracked from 1, 5 to 1, 5
Ant 10 backtracked from 10, 3 to 10, 3
Ant 3 backtracked from 1, 3 to 1, 4
Ant 8 backtracked from 1, 5 to 2, 5
Ant 10 backtracked from 10, 3 to 9, 3
Ant 3 backtracked from 1, 4 to 1, 5
Ant 8 backtracked from 2, 5 to 1, 5
Ant 10 backtracked from 9, 3 to 8, 4
Ant 3 backtracked from 1, 5 to 2, 5
Ant 8 backtracked from 1, 5 to 2, 6
Ant 10 backtracked from 8, 4 to 7, 5
Ant 3 backtracked from 2, 5 to 3, 5
Ant 8 backtracked from 2, 6 to 3, 5
Ant 10 backtracked from 7, 5 to 6, 6
Ant 3 backtracked from 3, 5 to 2, 5
Ant 8 backtracked from 3, 5 to 4, 5
Ant 10 backtracked from 6, 6 to 5, 5
Ant 3 backtracked from 2, 5 to 2, 4
Ant 8 backtracked from 4, 5 to 5, 4
Ant 10 backtracked from 5, 5 to 5, 4
Simulation moved on 15 stages
```

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Preview of Answers Ends Here

This is a limited inspection copy. Sample of answers ends here to stop students looking up answers to their assessments. See contents page for details of the rest of the resource.

Name	

ZigZag Education supporting

A Level AQA Computer Science Paper 1

Summer 2026

Ant Simulation

Electronic Answer Document (EAD)

Instructions

- Enter your name in the box at the top of this page
- Answer **all** questions by entering your answers into this document
- Remember to **save** this document regularly
- Save and print this document and any additional pages

- Answer **all** questions
- The marks available for each question are shown in brackets

- You will need:
  - □ access to a computer
  - □ access to a printer
  - □ access to appropriate software
  - □ electronic copies of the required skeleton code
  - □ EAD (Electronic Answer Document)

Total marks:

Exam-style Questions

Answer all questions.   Remember to save this document regularly.

Q	Answer		*Mark* *(leave blank)*
1	(a)		
	(b)		
	(c)		
2	(a)		
	(b)		
	(c)		
	(d)		
3	(a)		
	(b)		
4	(a)		
	(b)		
	(c)		
5			
6			
7			
8	(a)		
	(b)		
9	(a)		
	(b)		
10	(a)		
	(b)		
	(c)		
	(d)		
11	(a)		
	(b)		
	(c)		
	(d)		
	(e)		
	(f)		
12			
13			
14	(a)		
	(b)		
15			

Exam-style Programming Tasks

Answer all questions. Remember to save this document regularly.

Q	Answer	*Mark* *(leave blank)*
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		

© ZigZag Education, 2025