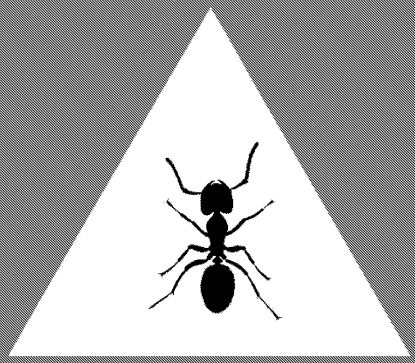
Zig Zag Education

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

PYTHON³ 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 12895

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

Product Support from ZigZag Education ii

Terms and Conditions of Use iii

Teacher's Introduction iv

Printouts of electronic resources (for reference)

- Code Breakdown (16 pages)
- Grid Example (1 page)
- Theory Questions: Non-write-on Version (2 pages)
- Theory Questions: Write-on Version (5 pages)
- Coding Tasks (23 pages)
- Additional Tasks (Extension) (2 pages)
- Theory Questions: Mark Scheme (4 pages)*
- Coding Tasks: Mark Scheme (66 pages)*
- Electronic Answer Document (EAD) (3 pages)
- UML Class Diagram: Complete (1 page)*
- UML Class Diagram: Activity (1 page)**

** 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

Skel' on Code Breakdo

Static Methods

Identifier / Data		Description
GetCellReference		
Parameters		This method asks the user for the row a inputs are cast to integers and assigne then returned.
Return values	ow : Int umn : Int	
GetChoice		
Parameters	n/a	This method assigns a string input from
Return values	**Choice** : String	
DisplayMenu		
Parameters	n/a	This method displays the main program
Return values	n/a	
Main		
Parameters	n/a	Th s me hc ets up the simulation. Th int list. The method asks for user i elect which configuration list to use w
Return values	n/a	
		The method then enters the main prog simulation options menu. The method t **GetChoice()** method ranging from 1 to **ThisSimulation** object. If the user selects 9 from the **GetChoice**

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Class: Simulation

Identifier / Data		Description
<<constructor>>		
Parameters	**SimulationParameters** : Int List	Initialises [illegible]ow[illegible]g protected attribute[illegible] • **[illegible]tar[illegible]mberOfNests** to the valu[illegible] • **[illegible]mberOfRows** to the value in inde[illegible] • **NumberOfColumns** to the value in [illegible] • **StartingFoodInNest** to the value in[illegible] • **StartingNumberOfFoodCells** to the[illegible] • **StartAntsInNest** to the value in inde[illegible] • **NewPheromoneStrength** to the val[illegible] • **PheromoneDecay** to the value in in[illegible] The method first instantiates the four ma[illegible] **Pheromones** and **Grid** as protected, em[illegible] and **Column** and uses these to perform [illegible] **Grid** represents the simulation "world". E[illegible] position in the "world". The bounds of thi[illegible] The method then calls the **SetUpANest**[illegible] the default **Row**,**Column** position of the [illegible] The method then uses a For loop to add [illegible] is governed by the variable **StartingNum**[illegible] which is initialised to 2. This means that [illegible] run, which me[illegible]at only one nest will [illegible] The loop [illegible]ates random values for **R**[illegible] 1 [illegible]**umberOfColumns** to keep the [illegible] positions selected are the same as the p[illegible] new random position. Once a valid posit[illegible] method to create a new nest. The method then enters another For loo[illegible] **StartingNumberOfFoodCells** to add ra[illegible] cells cannot be added into a cell contain[illegible] random locations until a valid location is [illegible] to add a hard-coded amount of 500 food [illegible] already present in the cell; therefore, it is [illegible]

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 n[illegible] f a **Nest** is at the sa[illegible] the n[illegible] ca[illegible] the **ChangeFood()**
Return values	n/a	
AddFoodToCell (public)		
Parameters	F[illegible] [illegible]n : Int **Quantity** : Int	This method uses the **GetIndex()** me its **Row** and **Column** position in the s method for the cell at that index and p that cell. The skeleton program often amount of food in a cell.
Return values	n/a	
AdvanceStage (public)		
Parameters	**NumberOfStages** : Int	This method contains a For loop whic from one stage to the next. The loop i Inside the loop, the method first instan **PheromonesToDelete**. It then iterat calling the **AdvanceStage()** method the **GetStrength()** method to query t reached 0, the **Pheromone** is added Once the [illegible]p [illegible]as completed, the me [illegible]on [illegible]esToDelete list, removing list [illegible]is because you can't remove with a Foreach loop. Once the loop has completed, the me the **AdvanceStage** method on each object called **TempCell** as a referenc investigated. The method then queries method calls the **AddFoodToNest()** together with its row and location. Th **UpdateFoodCarried()** method is call carrying, which will reduce the amoun
Return values	n/a	

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Identifier / Data		Description
		If the ant, however, is in a cell which c a food capacity greater than zero, the capacity of the ant. AQA updated this *(derived from the Ant class)* with a foc that it can pick up food, the method has what is available in the cell or gre **UpdateFoodInCell()** on the **Current** taken by the ant. Finally, it calls the U ant to carry back to the nest. If the ant is not in a food cell, the code its way back to the nest. If so, the met ant to reapply or generate new pheron statement then calls the **ChooseCell** **GetIndicesOfNeighbours()** and **Ge** controls whether the ant is moving ran The final task in the method is to itera each **Nest** in the **Nests** list.
GetAreaDetails (public)		
Parameters	**StartRow** : Int **StartColumn** : Int **EndRow** : Int **EndColumn** : Int	Uses nested iteration to enumerate t the top left and bottom right corners The method uses concatenation to bu **Details** string represents a single cell **Temp**[illegible] a reference of the cell in uses the **GetNestInCell()**, **GetNumbe** **GetAmountOfFood()** helper methods Once the loop has completed, the **D**[illegible]
Return values	**Details** : String	

INSPECTION COPY

COPYRIGHT PROTECTED

Identifier / Data		Description
GetCellDetails (public)		
Parameters	**Row** : Int **Column** : Int	This method is used to build up a m The meth tantiates a local cell o the l be g investigated. It firs nsta es a copy of the **Nest** in that FoodLevel() method on that cell The method then uses the same tech method, passing in the **CurrentCell** o find out how many ants are in that cel uses the **GetNumberOfPheromone** and how strong they are. Finally, the method adds some carria and returns the string.
Return values	**Details** : String	
GetDetails (public)		
Parameters	n/a	Uses nested iteration to enumerate t The method uses concatenation to b in the **Details** string represents a si called **TempCell** as a reference of th then uses the **GetNestInCell()**, **Get** and **GetAmountOfFood()** helper me Once the as 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

Identifier / Data		Description
GetIndexOfNeighbourWithStrongestPheromone (private)		
Parameters	**Row** : Int **Column** : Int	This method first instantiates the follo • **St** **estPheromone** to 0 **ex** **StrongestPherom**
Return values	**IndexOfStrongestPheromone** : Int	
		hen uses a Foreach loop to iterate cell at position **Row**, **Column**. This is method. If a neighbour is not valid (fo is represented as -1. In the Foreach l the strength of the pheromone in tha so, the method updates the **IndexOf** what the new **StrongestPheromon** After the loop has completed, the **Ind** contain any pheromones, the **IndexO** continue moving randomly.
GetIndicesOfNeighbours (private)		
Parameters	**Row** : Int **Column** : Int	This method builds up and returns a l **Grid** indices of the neighbouring cells at the position **Row**, **Column**. The m uses nested iteration to look at the ce 3×3 grid around the cell given at posi **Column**. For example, if **Row** was 3 **Column** was 2 in a standard 5×5 sim "world", t hod would return a lis [ini 5,6 ,-1,12,15,16,17]. If a cell is outside the bounds of the s "world", or if the cell being investigate the index is given as -1.
Return values	**ListOfNeighbours** : Int List	

INSPECTION COPY

COPYRIGHT PROTECTED

Identifier / Data		Description
GetNestInCell (public)		
Parameters	**C** : Cell	This method iterates through the **Ne** parameter **C**, they match, the **Nes** the **C** gives a none return.
Return values	**N** : Nest	
GetNumberOfAntsInCell (public)		
Parameters	**C** : Cell	This method initialises an integer va comparing the location of the **Ant** w variable is incremented. After the lo
Return values	**Count** : Int	
GetNumber[illegible]mo[illegible]esInCell (public)		
Parameters	**C** : Cell	This method initialises an integer va comparing the location of the **Phero** **Count** variable is incremented. Afte
Return values	**Count** : Int	
GetStrongestPheromoneInCell (private)		
Parameters	**C** : Cell	This method initialises an integer va list comparing the location of the **Ph** and if the **Count** variable and the st value in the variable **Strongest**, the the **Pheromone**. After the loop, **Str**
Return values	**Strongest** : Int	
SetUpANestAt (public)		
Parameters	**Row** : Int **Column** : Int	This meth[illegible]st adds a new **Nest** o the [illegible]ist. [illegible] then adds a new **Q** **An**[illegible]. Finally, it uses a For loop t [illegible]ver bound of 2 and the upper bou **StartingAntsInNest** must also inclu
Return values	n/a	
UpdateAntsPheromoneInCell ([illegible]		
Parameters	**A** : [illegible]	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 value	n/a	

INSPECTION COPY

COPYRIGHT PROTECTED
Zig Zag Education

Class: Cell

Identifier / Data		Description
<<constructor>>		
Parameters	**StartRow** : Int **StartColumn** : Int	The c ct r passes the parame then sets the protected attribute
Return values	n/a	
GetAmountOfFood (public)		
Parameters		This is an accessor method which
Return values	**AmountOfFood** : Int	
GetDetails (public) <<override>>		
Parameters	n/a	This method is only called from the G 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 upd **Change** to it.
Return values	n/a	

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

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 Ne • **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 **ColumnToChange** : Int	This method updates the position of t on the **NewCellIndicator** parameter the right s[illegible] the location of neighb arou[illegible]nt. [illegible]he **NewCellIndicato** nde[illegible]n. If the parameter is gre [illegible] 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 decrements the **ColumnToChange** or 8, the new direction must be right parameter to move the ant right. A c move diagonally within the table sho **RowToChange** and **ColumnToCha**
Return values	n/a	
ChooseCellToMoveTo (public) <<virtual>>		
Parameters	**ListOfNeighbours** : Int List **IndexOfNeighbourWithStronge stPheromone** : int	This is an empty base class virtual m
Return values	n/a	
ChooseRandomNeighbour (protected)		
Parameters	**ListOfNeighbours** : Int List	This [illegible] oses the index of a [illegible]od [illegible]he **Simulation** class and [illegible] select one of the neighbours does not contain -1. If an index cont location which is outside the bounds
Return values	Int	
GetDetails (public) <<override>>		
Parameters	[illegible]	This method is only called from the comprises the return value from the the **Ant**, the **TypeOfAnt**, and how m method in the base class, however,
Return value	String	
GetFoodCapacity (public) <<override>>		
Parameters	n/a	This is an accessor method which r
Return values	Int	

INSPECTION COPY

COPYRIGHT PROTECTED
Zig Zag Education

Identifier / Data		Description
GetFoodCarried (public) <<virtual>>		
Parameters	n/a	This is an accessor method which r
Return values	Int	
GetNestColumn (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 express attribute **NestRow** and if the current
Return values	Bool	
UpdateFoodCarried (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 parameters It then sets the following protected attributes: • **BelongsTo** to the parameter **BelongsToAnt** • **Strength** to the parameter **InitialStrength** • **PheromoneDecay** to the parameter **Decay**
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 returns
Return values	Int	
GetStrength (public)		
Parameters	n/a	This is an accessor method which returns
Return values	Int	
UpdateStrength (public)		
Parameters	**Change** : Int	This is a mutator method which updates
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 S • **NumberOfQueens** to 1 • **ID** to the static attribute **NextN** 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 i exits and the nest doesn't advance. The method then initialises three int setting them all to 0. It then iterates Note that this DOESN'T check if the is a **Queen** ant, the **Count** variable normal worker ant, the **Count** varia incremented by 1. This is perhaps s ***WorkerAntsBelongingToNestCo*** In reality it represents the amount of food *amount of food in the nest decrease ants that belong to the nest*", which regardless of their cell location in th **ChangeFood()** method is called, pa amount of food in the nest. The method then performs a numbe in the nest is 0 and the **AntsInNest** *simulation belonging to this nest, b* incremented. If the **FoodLevel** of th this nest, the **AntsToCull** variable i
Return values	**Nests** : Nest List **Ants** : Ant List **Pheromones** : Pheromone List	

INSPECTION COPY

COPYRIGHT PROTECTED
Zig Zag Education

Identifier / Data		Description
		The method then performs a wider number of worker ants belonging to incremented. Within this scenario, t greater [illegible] e number of worker a [illegible] ipp[illegible]tion from attempting to re Th[illegible]thod then uses a For loop, w ants from the **Ants** list. The method The loop will not select an ant whic ant selected could be anywhere in t nest. This means that any ant belo food cell or carrying food back to th **NumberOfQueens** variable is decr If the **FoodLevel** is not less than fiv is deemed to be healthy, and there f there is a 50% chance that the **Que** will be a new **Queen**. Combined, th a new **Queen** is added to the simul ants are added to the **Ants** list. The three modified lists – **Nests**, **A**
ChangeFood (public)		
Parameters	**Change** : Int	This method adjusts the **FoodLeve** It then chec[illegible]he updated **FoodLe** back [illegible]
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 con[illegible]r sets the following p[illegible] • **Ro**[illegible] to th[illegible] parameter **StartRo**[illegible] • [illegible]**umn** to the parameter **Sta**[illegible] • [illegible] **ID** to None *(so it can be updat*[illegible] Both of these attributes are updatec
Return values	n/a	
AdvanceStage (public) <<vi[illegible]		
Parameters	[illegible] Nest List **Ants** : Ant List **Pheromones** : Pheromone List	This is an empty base class virtual [illegible]
Return values	n/a	
GetColumn (public)		
Parameters	n/a	This is an accessor method which [illegible]
Return values	Int	
GetDetails (public) <<virtual>>		
Parameters	n/a	This is a base class virtual method w[illegible]
Return values	String	
GetID (public)		
Parameters	n/a	This [illegible] [illegible]ccessor method which [illegible]
Return values	Int	
GetRow (public)		
Parameters	n/a	This is an accessor method which [illegible]
Return values	[illegible]	
InSameLoca[illegible]**lic**		
Parameters	**E** : Entity	This returns the result of an express[illegible] parameter **E** and the attribute **Colu**[illegible] entities are in the same place).
Return values	Bool	

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Class: Queen

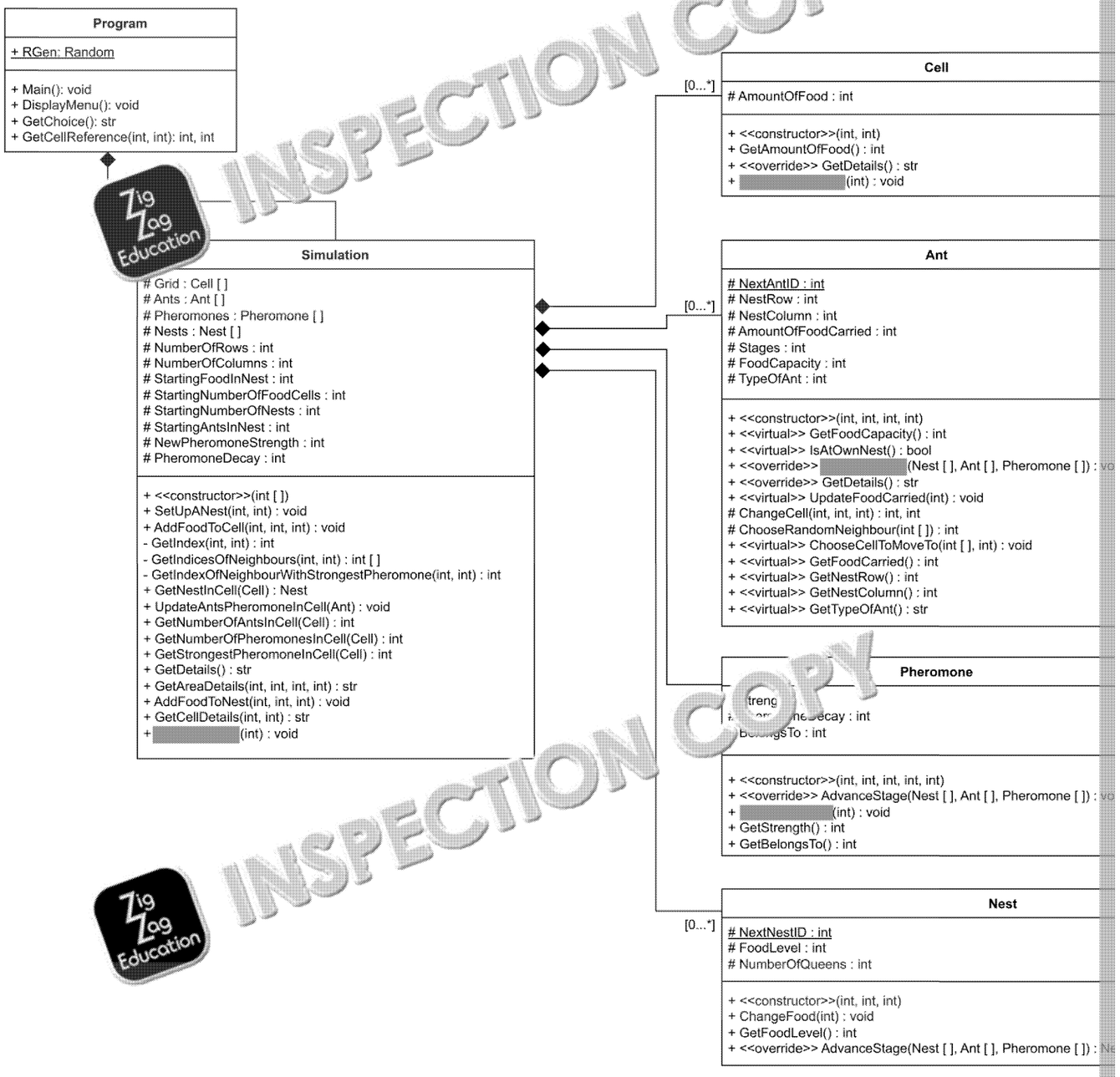
Identifier / Data		Description
<<constructor>>		
Parameters	**StartRow** : Int **StartColumn** : Int **NestInRow** : Int **NestInColumn** : Int	The constructor passes all four para It then sets the **TypeOfAnt** attribute
Return values	n/a	

Class: WorkerAnt

Identifier / Data		Description
<<constructor>>		
Parameters	**StartRow** : Int **StartColumn** : Int **NestInRow** : Int **NestInColumn** : Int	The constructor passes all four para It then sets the **TypeOfAnt** attribute attribute (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 is carrying any food. If it is, the **Column** of the worker ant are less respectively. The **Row** and **Column** to move the worker ant one cell per Alternatively, if the worker ant is not **IndexOfNeighbourWithStrongestP** currently searching for food and the contains pheromone. In this case the passing in the **ListOfNeighbours** p calls the **ChangeCell()** method usin If the **IndexOfNeighbourWithStro** next to a cell or cells with pheromo the **IndexOf()** method in the built-in **IndexOfNeighbourWithStrongestP** the **ChangeCell()** method using this
Return values	n/a	
GetDetails (public) <<override>>		
Parameters	n/a	This method is only called from the which comprises the return value fr **AmountOfFoodCarried** of the **Wo**
Return values	String	

INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Ant Simulation



INSPECTION COPY
INSPECTION COPY
INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

Simulation 1

	1	2	3	4	5
1	*Grid Index 0*	*Grid Index 1* **Food:**	*Grid Index 2*	*Grid 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 19*
5	*Grid Index 20*	*Grid Index 21*	*Grid Index 22*	*Grid Index 23*	*Grid Index 24*

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

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

Ant Simulati

Python Theory Questions

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

TOTAL MARKS: 54

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

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

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

(b) The code contains a prompt on line 11: **Enter simulation numbe**

What error would occur during runtime if the user typed "four" whe
from line 11?

(c) How would a programmer resolve this issue?

2. Give an identifier used for:

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

(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

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 ma
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

COPYRIGHT PROTECTED

Zig Zag Education

Ant Simulati

Python Theory Questions

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

TOTAL MARKS: 54

1. The **Main** procedure in the pr[illegible] program asks the user to ente
One of those is **the simulation number**.

Current[illegible] assumes the user will always enter valid nume
1–4 in[illegible].

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

.....

.....

.....

(b) The code contains a prompt on line 11: **Enter simulation numbe**

What error would occur during runtime if the user typed "four" whe
from line 11?

.....

.....

(c) How would a programmer resolve this issue?

.....

.....

2. Give an identifier used for:

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

.....

.....

(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

Programming Tasks

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

Note that any alternative or additional code changes that [illegible] eemed appropriate [illegible] ensuring that it is clear where in the [illegible] Program those change

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

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

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

These questions are based on Version 2 of the Ant Simulation code released by [illegible] you are using Version 2 of the code from Centre Ser[illegible]

INSPECTION COPY

INSPECTION COPY

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

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 a valid simulation 1.
- Enter 9 to quit the simulation.
- Show the program correctly displaying a check to confirm that the user
- 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
Zig Zag Education

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 t 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 1 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 protec 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 (Extens

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 re "food source exhausted" pheromone should quadruple the decay level of t help erase the path more quickly.

Extens 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 th **ChooseCellToMoveTo** method as additional weighting for selecting wheth pheromone trail back to a food source.

INSPECTION COPY

COPYRIGHT PROTECTED

Zig Zag Education

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 w
WorkerAnt class so that an ant won't collect a particular food source if the
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 nest
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 **ShareFoodWithNear**
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 nest
main menu to allow the user to place food into cells. Demonstrate ants igno
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 grea
time.

Extension 14

Introduce the concept of "[illegible]ality" whereby a worker ant defects to a dif
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 of
to the nest, it is assessed by the ants in the nest. If it is lower than other foo
at, a **ReviewerAnt** follows the pheromone trail out to the food source, remo

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 in 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 an 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 effectively to this question; however, it could be done with a

Example Solution

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

```
class WorkerAnt(Ant):
    def __init__(self, StartRow, StartColumn, NestInRow, NestInColumn):
        super().__init__(StartRow, StartColumn, NestInRow, NestInColumn)
        self._TypeOfAnt = "worker"
        self._FoodCapacity = 30
        #CHANGE
        self.__MovementHistory = list[tuple[int, int]]()
        self.__ForagingStages = 0
        self.__BackTracking = False
        #END CHANGE
```

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

```
    def ChooseCellToMoveTo(self, ListOfNeighbours, IndexOfNeighbourWithStrongestP
        if self._AmountOfFoodCarried > 0:
            #CHANGE
            self.__ForagingStages = 0   #Covers the scenario of ant finds food b
            self.__MovementHistory.clear()
            if self._Row > self._NestRow:
                self._Row -= 1
            elif self._Row < self._NestRow:
                self._Row += 1
            elif self._Column > self._NestColumn:
                self._Column -= 1
            elif self._Column < self._NestColumn:
                self._Column += 1
        #CHANGE
        elif self.__ForagingStages >= 7:
```


INSPECTION COPY
COPYRIGHT PROTECTED
Zig Zag Education

```
            self.BacktrackToPreviousCell()
        elif IndexOfNeighbourWithStrongestPheromone == -1:
            IndexToUse = self._ChooseRandomNeighbour(ListOfNeighbours)
            self._Row, self._Column = self._ChangeCell(IndexToUse, self._Row, se
            #print(f"ForagingStages for this Ant: {self.__ForagingStages}")
            self.__MovementHistory.append((self._Row, self._Column))    #The ant
trail
            self.__ForagingStages += 1
        else:
            IndexToUse = ListOfNei[illegible]ex(IndexOfNeighbourWithStrongestPhe
            self._Row, self.[illegible] = self._ChangeCell(IndexToUse, self._Row, se
            self.__Forag[illegible] = 0   #Once it finds a pheromone trail, the an
            [illegible]_M[illegible]istory.clear()
        #EN[illegible]
```

Creation of Back[illegible]ckToPreviousCell method which is called when ant reaches seven stages of mov

```
        #CHANGE
        elif self.__ForagingStages >= 7:
            self.BacktrackToPreviousCell()
        #END CHANGE
        elif IndexOfNeighbourWithStrongestPheromone == -1:
            IndexToUse = self._ChooseRandomNeighbour(ListOfNeighbours)
            self._Row, self._Column = self._ChangeCell(IndexToUse, self._Row, se
        else:
            IndexToUse = ListOfNeighbours.index(IndexOfNeighbourWithStrongestPhe
            self._Row, self._Column = self._ChangeCell(IndexToUse, self._Row, se

    #CHANGE
    def BacktrackToPreviousCell(self):
        self.__BackTracking = True
        if len(self.__MovementHistory) [illegible]
            print(f"Ant {self._ID[illegible] from {self._Row}, {self._Column}"
            PreviousPositio[illegible]_MovementHistory.pop()
            self._Ro[illegible]ous[illegible]osition[0]
            [illegible]Co[illegible]reviousPosition[1]
            [illegible]"t[illegible] {self._Row}, {self._Column}")
            [illegible]._Row == self._NestRow and self._Column == self._NestColumn:
                self.__BackTracking = False
                self.__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 [illegible] of stages to advance by: 15
Ant 3 [illegible]acked from 1, 3 to 1, 3
Ant 8 b[illegible]tracked 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 [illegible]
Ant 3 backtracked from 3, [illegible]
Ant 8 backtracked fr[illegible] 5 t[illegible], 5
Ant 10 backtrack[illegible]m [illegible], 6 to 5, 5
Ant 3 [illegible]ac[illegible]om 2, 5 to 2, 4
Ant 8 [illegible]acked from 4, 5 to 5, 4
Ant 10 [illegible]racked 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:

 © ZigZag Education 2025

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			

 © ZigZag Education, 2025

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