Zig Zag Education

Computer Science

A Level | AQA | 7517

AQA PAPER 1 EXAM RESOURCE PACK 2017

RABBITS AND FOXES

for A Level AQA Computer Science

zigzageducation.co.uk

POD
7226

Contents

Thank you for choosing ZigZag Education!

Talk to Us!

Love it as it is?
Let the author and other teachers know what you think

Got a suggestion?
If your improvement leads to an update we will send you a new copy for free

Found a problem?
We will fix it and send you a free updated copy

We ♥ your feedback!
Let us know what you think using the feedback sheet on the next page.
£10 ZigZag Voucher for detailed & complete reviews!

Web:	Email:	Real Person:	Fax:	Post:
zzed.uk/more	**computerscience @zigzageducation.co.uk**	**0117 950 3199**	**0117 959 1695**	**ZigZag Education, Unit 3, Greenway Business Centre, Doncaster Road, Bristol BS10 5PY**

Become a Published Author

ZigZag is a large community of over 6000 teachers & educationalists.
Review new titles, publish your own work or write to a brief.

Fancy being involved?

Then register at...
publishmenow.co.uk
The Professional Publishing Community

Alternatively email new resource ideas directly to...
publishmenow@zigzageducation.co.uk

For more resources go to **zzed.uk/more**
where you can preview every page before you buy

Teacher Feedback Opportunity

£10 ZigZag Voucher for detailed & complete reviews! ◆ Use for problems/areas for improvement/positive feedback

Resource ID & Name	7226 – A Level AQA Computer Science Paper 1 Exam Resource Pack 2017 (VB.NET Edition)
School Name	

Your Name		**Position**	

Overall, what did you think about this resource? ....

....

....

....

I particularly like this resource because... ....

....

....

How does it help you or your students? ....

....

It is better than some other resources because... ....

....

....

What might you say to a colleague in a neighbouring school to persuade them to use this resource? ....

....

....

....

How well does it match your specification (and which specification is this)? ....

....

Other comments, suggestions for improvements, errors found (please give page numbers) etc. ....

....

....

Resources I would like published:	
Resources I might write, *or have written*, for consideration for publication:	

Fax to:	Email to:	Submit online:	Post to:	INTERNAL USE ONLY Feedback logged: ✓□ Complete & detailed: Y /N If detailed, £10 sent: ✓□
0117 959 1695	**feedback@ zigzageducation.co.uk**	**zzed.uk/feedback**	**ZigZag Education, Unit 3, Greenway Business Centre, Doncaster Road, Bristol BS10 5PY**	

Terms and Conditions of Use

Terms and Conditions

Please note that the **Terms and Conditions** of this resource include point 5.3, which states:

> **"You acknowledge that you rely on your own skill and judgement in determining the suitability of the Goods for any particular purpose."**

"We do not warrant: that any of the Goods are suitable for any particular purpose (e.g. any particular qualification), or the results that may be obtained from the use of any publication, or expected exam grades, or that we are affiliated with any educational institution, or that any publication is authorised by, associated with, sponsored by or endorsed by any educational institution."

Copyright Information

Every effort is made to ensure that the information provided in this publication is accurate and up to date but no legal responsibility is accepted for any errors, omissions or misleading statements. It is ZigZag Education's policy to obtain permission for any copyright material in their publications. The publishers will be glad to make suitable arrangements with any copyright holders whom it has not been possible to contact.

Students and teachers may not use any material or content contained herein and incorporate it into a body of work without referencing/acknowledging the source of the material ("Plagiarism").

Disclaimer

> This resource is intended to supplement your teaching only.
>
> It is the teacher's responsibility to decide how to use this resource to assist themselves and their students appropriately. You may simply wish to read this material to better inform yourself and to help you prepare your lessons and to give you ideas for your teaching. You may also consider whether it is appropriate to hand out some of the sheets for reference and to use some of the activities for classwork or homework. You may also consider whether it is appropriate to hand out the booklet to be worked through by your students more independently.
>
> As with all pre-release material, it is the teacher's responsibility to decide in what way to assist their students, and to decide how this resource in particular can be used to fit into that assistance.
>
> **The resources here are provided as an interpretation of the pre-release material.**
> **The author does not have any special knowledge of what to expect on any particular exam.**

ZigZag Education is not affiliated with AQA, Pearson, Edexcel, OCR, WJEC, CEA, International Baccalaureate Organization or DFE in any way nor is this publication authorised by, associated with, sponsored by or endorsed by these institutions unless explicitly stated on the front cover of this publication.

This publication is designed to supplement teaching only. Practice questions may be designed to follow the content of a specification and may also attempt to prepare students for the type of questions they will meet in the examination, but will not attempt to predict future examination questions. ZigZag Education do not make any warranty as to the results that may be obtained from the use of this publication, or as to the accuracy, reliability or content of the publication.

Where the teacher uses any of the material from this resource to support examinations or similar then the teacher must ensure that they are happy with the level of information and support provided pertaining to their personal point of view and to the constraints of the specification and to others involved in the delivery of the course. It is considered essential that the teacher adapt, extend and/or censor any parts of the contained material to suit their needs, the needs of the specification and the needs of the individual or group concerned. As such, the teacher must determine which parts of the material, if any, to provide to the students and which parts to use as background information for themselves. Likewise, the teacher must determine what additional material is required to cover all points on the specification and to cover each specification point to the correct depth.

Teacher's Introduction

This pack is designed to help you support your students taking the A Level Computer Science Paper 1 examination. It is based on the 'Rabbits & Foxes' preliminary material (VB .NET) – for examination June 2017.

It consists of the following:

① **Pre-release Commentary** (for teachers)
A detailed overview of the skeleton program, describing all VB code elements and routines.

This section is designed to help you get to grips with the program, so that you can feel confident helping your students. This commentary is not designed to be given to students before they have explored the code for themselves, and if used in this way could lead to misconceptions of how the program works.

② **UML Diagram Activity**
A partially incomplete UML class diagram for students to complete while getting to grips with the skeleton program. Any missing operations and attributes must be added to the diagram. A completed version is provided in the solutions section at the back of the resource.

③ **Programming Theory Questions**
Theory questions test students' understanding of the 'Rabbits & Foxes' code, like Section C in the exam. These are provided in both write-on and non-write-on format.

④ **Programming Exercises**
Modification exercises put students' programming skills to the test, like Section D in the exam. An Electronic Answer Document (EAD) and the modified VB code are provided on the CD.

Answers and solutions for the UML Diagram activity, theory questions and programming exercises are provided from page 22 onwards. Note that for the programming exercises in particular, these are example solutions and you must use your discretion to award marks accordingly where there are valid alternative solutions.

The **Appendices** contains some additional resources, including:

- Further modifications worksheet: a template for brainstorming further enhancements to the skeleton program. This is suggested as a group activity, so that students (and the teacher) can share their ideas, thus increasing the likelihood of covering every area that will come up in the exam.
- Electronic Answer Document (EAD) printout: hard copy version of the file on CD (for reference).

Enter the URL zzed.uk/7226 in your web browser to download a folder containing the following:

- **MODIFIED_VB_CODE.txt** – text file containing the new and/or modified program code as shown in the mark scheme for section ④ (from page 25).
- **PAPER1_EAD.docx** – Electronic Answer Document for completing sections ③ and ④

This resource is intended to supplement your teaching only. It is the teacher's responsibility to decide how to use this resource to assist themselves and their students appropriately. You may simply wish to read this material to better inform yourself and to help you prepare your lessons and to give you ideas for your teaching. You may also consider whether it is appropriate to hand out some of the sheets for reference and to use some of the activities for classwork or homework. You may also consider whether it is appropriate to hand out the booklet to be worked through by your students more independently. As with all pre-release material, it is the teacher's responsibility to decide in what way to assist their students, and to decide how this resource in particular can be used to fit into that assistance.

The resources here are provided as an interpretation of the pre-release material. The author does not have any special knowledge of what to expect on any particular exam.

Pre-Release Commentary

Rabbits and Foxes

Description of the Program

The program is a simulation of rabbit population over time and how it is affected by the foxes who hunt them.

The world is represented by a grid in which each square can contain a rabbit warren (a burrow where many rabbits live) or a fox, or both. F designates a fox, and a number designates a rabbit warren (and how many rabbits are in the warren).

The menu holds the following options:

- Run the simulation with default settings
- Run the simulation with custom settings
- Exit

The settings that can be changed in option 2 include:

- Landscape size
- Number or rabbit warrens at start
- Number of foxes at start
- Randomness (as a %)

During the simulation you can advance to the next time period showing detail or not, as well as inspect the current state of a fox or rabbit warren.

Each time a period runs, the rabbits can:

- Be eaten by a fox
- Be killed by something other than a fox
- Die of old age
- Increase in number (a number of new baby rabbits are born)

This information is displayed for each warren.

Each time a period runs there is a report on the foxes' age, how much food the foxes have eaten compared to what they need, and whether they have reproduced. If they have reproduced, the location of the new foxes is displayed at the bottom.

Rabbits and Foxes

Description of Program Classes

This program contains multiple classes used to simulate foxes and rabbits in their natural environment.

The classes have been listed below, along with a very brief description of their purpose.

Class	Description
Location	A class that creates an object corresponding to a location on the grid.
Simulation	The class that drives the main simulation.
Warren	A class that simulates a rabbit warren (where they live).
Animal	An abstract class used for creating foxes and rabbits. It contains all the variables and functions common to both animals.
Fox (inherits Animal)	The class used to model foxes.
Rabbit (inherits Animal)	The class used to model rabbits.

Description of Class Variables

Each class has a number of variables, only accessible in that particular class. For each of the classes above, their variables have been listed, along with a brief description.

Location – Instance variables	Type	Description
Fox	Fox	This value is equal to None when the simulation is started. This value will hold a Fox object, if there is a fox in this particular location.
Warren	Warren	This value is equal to None when the simulation is started. This value will hold a Warren object, if there is a warren in this particular location.

Simulation — Instance variables	Type	Description
ViewRabbits	String	Variable that should either have the value ‘y’ or ‘n’.
TimePeriod	Integer	Counter to store how many iterations of the simulation have occurred.
WarrenCount	Integer	Variable that counts the number of warrens.
FoxCount	Integer	Variable that counts the number of foxes.
ShowDetail	Boolean	If this is true, more detail will be shown about the simulation.
LandscapeSize	Integer	Value that stores the size of the Landscape (the landscape is assumed to be square).
Variability	Integer	Value that determines how differently the simulation can vary. A high variability increases the range of possible other variable values.
FixedInitialLocations	Boolean	If True, the warrens and foxes will start in a fixed location.
Landscape	Array	2D array of locations used to store foxes and warrens

Warren — Instance variables	Type	Description
MaxRabbitsInWarren	Integer	Constant that stores the maximum number of rabbits that can be stored inside a warren.
RabbitCount	Integer	The value that stores the number of rabbits who are alive.
PeriodsRun	Integer	This variable stores how many periods have passed since the start of the program.
AlreadySpread	Boolean	Boolean value used to determine whether a new warren needs to be created (if an existing one has become too large).
Variability	Integer	Value that determines how differently the simulation can vary. A high variability increases the range of possible other variable values.
Rabbits	Array	An array containing the rabbits that are currently alive in the specified warren.

Animal — Instance variables	Type	Description
NaturalLifespan	Double	Integer value stating how long (in iterations) the animal will live for before dying of natural causes.
ProbabilityOfDeathOtherCauses	Double	Decimal value used for calculating the chance of death from other reasons.
IsAlive	Boolean	Boolean value that states whether an animal is alive or not.
ID	Integer	Integer value given to uniquely identify the animal.
Age	Integer	Value used to store the age of an animal (in iterations).
NextID	*Integer*	*Value used to make sure that each new instance is given a unique identification number.* *Note: this is a CLASS VARIABLE, shared by every instance of the class.*

Fox — Instance variables	Type	Description
DefaultLifespan	Integer	Value used for calculating the lifespan of the fox. The actual lifespan is calculated in the Animal class using the variability variable in the Simulation class.
DefaultProbabilityDeathOtherCauses	Double	Probability used for calculating the chance of dying from random causes. The actual probability is calculated in the Animal class using the variability variable in the Simulation class.
FoodUnitsNeeded	Integer	Number of food units needed to stop the fox from aging or dying.
FoodUnitsConsumedThisPeriod	Integer	Number of food units that have been consumed in one iteration of the simulation.

Rabbit — Instance variables	Type	Description
DefaultLifespan	Integer	Constant used for calculating the lifespan of a rabbit. The actual life span is calculated in the Animal class using the Variability variable in the Simulation class.
DefaultProbabilityDeathOtherCauses	Double	Probability used for calculating the chance of dying from random causes. The actual probability is calculated in the Animal class using the variability variable in the Simulation class.
DefaultReproductionRate	Double	Constant used to set the default Reproduction Rate if none is passed into the constructor.
ReproductionRate	Double	Probability used for calculating the chance that any two rabbits reproduce. Inherited from the rabbits parents.
Genders	Enum	The gender of the rabbit, equal to either Male or Female.

Description of Class Methods

Along with class variables, each class has a number of methods unique to that class. For each class, its functions Ⓕ and procedures Ⓟ have been described below.

Location — Methods	Description	
New Ⓟ	Input: None Output: None	Creates a location instance: 1. Initially there are no foxes in the location 2. Initially there are no warrens in the location

Simulation — Methods	Description	
New Ⓟ	Input: Size of landscape (Integer), initial number of warrens (Integer), initial number of foxes (Integer), variability (Integer), whether fixed locations should be used or not (Boolean) Output: None	Creates a simulation instance: 1. Creates an array of Location instances according to the size of the landscape. 2. Adds foxes and warrens to the landscape. 3. Draws the landscape. 4. Starts the main simulation loop giving options to advance the generation, or inspect a fox/warren.
InputCoordinate Ⓕ	Input: Coordinate name ('x' or 'y') Output: Coordinate (Integer)	Asks the user to enter a coordinate – depending on the supplied coordinate axis (x or y). Returns an integer value corresponding to the specified coordinate axis.
AdvanceTimePeriod Ⓟ	Input: None Output: None	Updates the simulation. 1. For each location: a. If there is a warren in the space, and there are foxes that are alive, and they are near the warren, then they should eat some rabbits. b. If the warren has reached its capacity, then a new warren needs to be created. c. The warren should then advance to the next generation. d. If the warren is now empty, then it should be removed from the landscape. 2. For each location: a. If there is a fox in the space, advance to the next generation. b. Check whether the fox has died. i. If it has, remove it from the landscape, and jump to 3). ii. If it has not died, check whether it should reproduce. c. Reset the amount of food that it has consumed in this period. 3. If new foxes should be born, create and add them to the landscape.

Simulation — Methods (cont.)	Description	
CreateLandscapeAndAnimals Ⓟ	Input: Initial number of warrens (Integer), initial number of foxes (Integer), whether fixed locations should be used or not (Boolean) Output: None	Creates the landscape. 1. If the locations of each warren and fox have been fixed, create them in the fixed locations. 2. Otherwise, create new warrens and foxes randomly. The number is determined by the initial fox and warren count.
CreateNewWarren Ⓟ	Input: None Output: None	Creates a new warren. 1. Find a spot that does not already contain a warren. 2. Create a new Warren instance in that spot.
CreateNewFox Ⓟ	Input: None Output: None	Creates a new fox. 1. Find a spot that does not already contain a fox. 2. Create a new Fox instance in that spot.
FoxesEatRabbitsInWarren Ⓟ	Input: Warren's x-coordinate (Integer), warren's y-coordinate (Integer) Output: None	Function that lets foxes eat rabbits. 1. For each location: a. If there is a fox in the location and they are less than 3.5 units away from a warren, 20% of the rabbits should be eaten. b. OTHERWISE if there is a fox in the location and they are less than 7 units away from a warren, 10% of the rabbits should be eaten. c. OTHERWISE no rabbits should be eaten.
DistanceBetween Ⓕ	Input: Two sets of x- and y-coordinates Output: Distance between the points (Double)	Calculates the distance between points – using Pythagoras' theorem.
DrawLandscape Ⓟ	Input: None Output: None	Draws the landscape shown in the simulation. It checks each location and draws either a W for a warren or an F for a fox.

Warren — Methods	Description	
New Ⓟ	Input: Variability (Integer), number of rabbits in warren (Integer) Output: None	Creates a new Warren instance. 1. Creates spaces for the maximum number of allowed rabbits in a warren. 2. If the number of rabbits in the warren is not provided, it decides on an initial number of rabbits to have in the warren – dependent on the variability. 3. It adds that number of rabbits to the warren.
CalculateRandomValue Ⓕ	Input: Base value (Integer), variability (Integer) Output: Random value (Integer)	Provides a random number centred around the provided base value. If the variability is high, the range of possible values is higher.

Warren — Methods (cont.)	Description	
GetRabbitCount Ⓕ	Input: None Output: Number of rabbits in warren (Integer)	Returns the number of rabbits in the warren that the function is being called from.
NeedToCreateNewWarren Ⓕ	Input: None Output: Whether a new warren needs to be created (Boolean)	1. Checks whether a warren has reached capacity, and hasn't already been split up. 2. If this is true, then a new warren needs to be created.
WarrenHasDiedOut Ⓕ	Input: None Output: Whether a warren is empty or not (Boolean)	This function checks the number of rabbits in the warren. 1. If there are no rabbits it returns True. 2. Otherwise, it returns False.
AdvanceGeneration Ⓟ	Input: Whether you should show detail (Boolean) Output: None	Advances the warren to the next generation. 1. If there are rabbits, kill some of them off from other factors. 2. If there are rabbits they should be aged. 3. If there are rabbits, and the warren is not overfull, and the warren contains males, then rabbits should breed. 4. Otherwise, a message will be printed if detail is shown saying that all of the rabbits are dead in that particular warren.
EatRabbits Ⓕ	Input: Number of rabbits that need to be eaten (Integer) Output: Updated number of rabbits to be eaten (Integer)	Removes a fixed number of rabbits from the warren. 1. Finds a rabbit in the warren at random. 2. Removes it from the warren. 3. Repeats until enough rabbits have been eaten. 4. Compresses the list of rabbits.
KillByOtherFactors Ⓟ	Input: Whether you should show detail (Boolean) Output: None	Kills rabbits at random depending on the percentage chance of a rabbit randomly dying from other causes. 1. Goes through the list of rabbits in the warren. 2. Checks whether they have died from other causes. 3. Removes them from the list of rabbits. 4. Compresses the list of the remaining rabbits.
AgeRabbits Ⓟ	Input: Whether you should show detail (Boolean) Output: None	Makes each rabbit older. 1. Goes through the list of rabbits in the warren, incrementing their age. 2. Determines whether a rabbit has died of old age. a. If they have, increase the death count, remove them from the rabbit list, and compress the list of living rabbits.

Warren — Methods (cont.)	Description	
MateRabbits Ⓟ	Input: Whether you should show detail (Boolean) Output: None	Function that makes new rabbits. 1. Goes through the list of rabbits finding females. 2. If the rabbit is female and there is space for a baby rabbit: a. Finds a male rabbit to breed with b. Combines their reproduction rates c. If it is greater than 1, a new rabbit is born
CompressRabbitList Ⓟ	Input: Number of dead rabbits (Integer) Output: None	Shifts the rabbits so in the list there are no spaces between them.
ContainsMales Ⓕ	Input: None Output: Whether a warren contains males (Boolean)	Checks whether a warren has male rabbits in it. 1. It assumes that there are no males. 2. If it sees a male somewhere in the list, the function will return True.
Inspect Ⓟ	Input: None Output: None	Prints the age of the warren, and the number of rabbits that it contains.
ListRabbits Ⓟ	Input: None Output: None	Prints the status of each rabbit in the rabbits list.

Animal — Methods	Description	
New Ⓟ	Input: Average lifespan (Integer), average probability of dying from other causes (Double), variability (Integer) Output: None	Constructs a new instance of Animal.
CalculateNewAge Ⓟ	Input: None Output: None	Increments the animal's age and determines whether it is still alive.
CheckIfDead Ⓕ	Input: None Output: Boolean	Whether the animal is dead or not.
Inspect Ⓟ	Input: None Output: None	Prints out the animal's current state.
CheckIfKilledByOtherFactor Ⓕ	Input: None Output: Boolean	Determines whether the animal has been killed by another factor.
CalculateRandomValue Ⓕ	Input: Base value (Integer), variability (Integer) Output: Double	Calculates a random value.

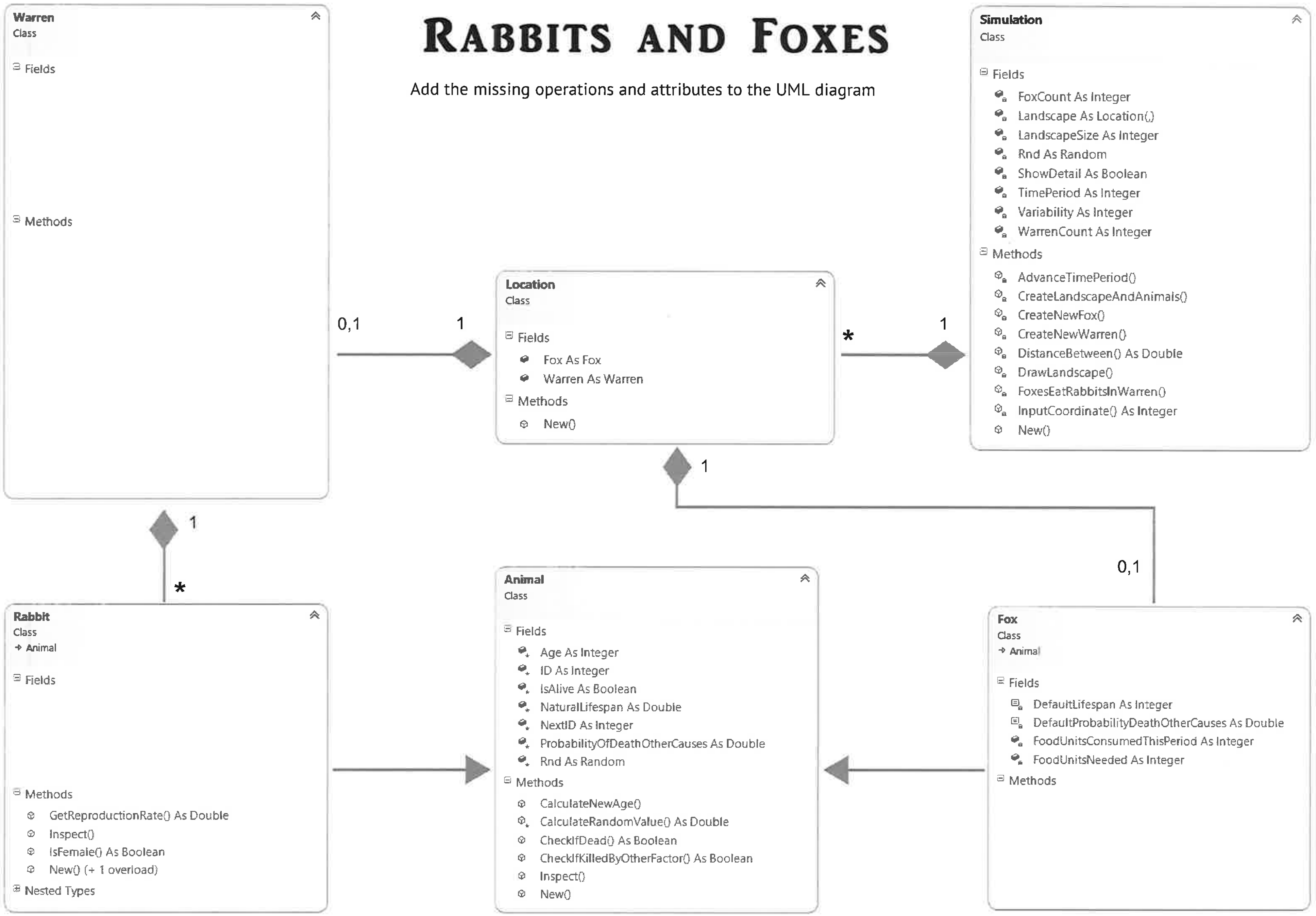
Fox — Methods	Description	
New Ⓟ	Input: Variability (Integer) Output: None	Constructor – creates a new instance of Fox.
AdvanceGeneration Ⓟ	Input: Whether detail should be shown (Boolean) Output: None	Determines whether the fox has died or by how much it ages.
ResetFoodConsumed Ⓟ	Input: None Output: None	Resets this value to 0.
ReproduceThisPeriod Ⓕ	Input: None Output: Boolean	Determines whether the fox should reproduce.
GiveFood Ⓟ	Input: Number of food units (Integer) Output: None	Adds the number of food units passed in to the food consumed.
Inspect Ⓟ	Input: None Output: None	Prints out the fox's current state (overrides method of same name in Animal).

Rabbit — Methods	Description	
New Ⓟ	Input: Variability (Integer), parents reproduction rate (Double) Output: None	Constructor method to create a new instance of Rabbit.
Inspect Ⓟ	Input: None Output: None	Print out the rabbit's current state (overrides method of same name in Animal).
IsFemale Ⓕ	Input: None Output: Boolean	Returns whether the rabbit is male or female.
GetReproductionRate Ⓕ	Input: None Output: Reproduction rate (Double)	Returns the reproduction rate.

In addition to the functions and procedures found in the classes, there is also the main program.

Rabbits and Foxes

Add the missing operations and attributes to the UML diagram



Programming Theory Questions

These questions refer to the Preliminary Material and require you to load the Skeleton Program, but do not require any additional programming.

/50

1. Give an example of instantiation from the skeleton program. [1 mark]

....

2. State the name of an identifier(s) for the following:

   a. An array variable [1 mark]

   ....

   b. A subclass [1 mark]

   ....

   c. A parent class [1 mark]

   ....

   d. A class variable [1 mark]

   ....

   e. An accessor method [1 mark]

   ....

   f. A mutator method [1 mark]

   ....

   g. A variable used to store a whole number [1 mark]

   ....

   h. A Boolean variable [1 mark]

   ....

   i. Four constants that store a float [4 marks]

   ....

   ....

3 a. Two classes that have a composition aggregation relationship. [2 marks]

....

b. Why is Warren to Rabbit not an example of association aggregation? [1 mark]

....

....

4. Are there any examples of polymorphism in the skeleton code? [1 mark]

....

5. State the name of an identifier for a procedure or function that is overridden in a subclass. [1 mark]

....

6. Look at the EatRabbits subroutine in the Warren class in the skeleton program.
Why does the generation of a random rabbit need to be inside a repetition structure? [1 mark]

....

7. Look at the Warren class. Why has a named constant been used instead of a numeric value? [2 marks]

....

....

....

8. State the name of an identifier for an enumerated data type. [1 mark]

....

9. How could the Fox class be changed to make the foxes live longer? [1 mark]

....

10. What is the purpose of the variable AlreadySpread in the Warren class and how is it used? [4 marks]

....

....

....

....

....

11. What is the purpose of the method CompressRabbitList? [2 marks]

....

....

....

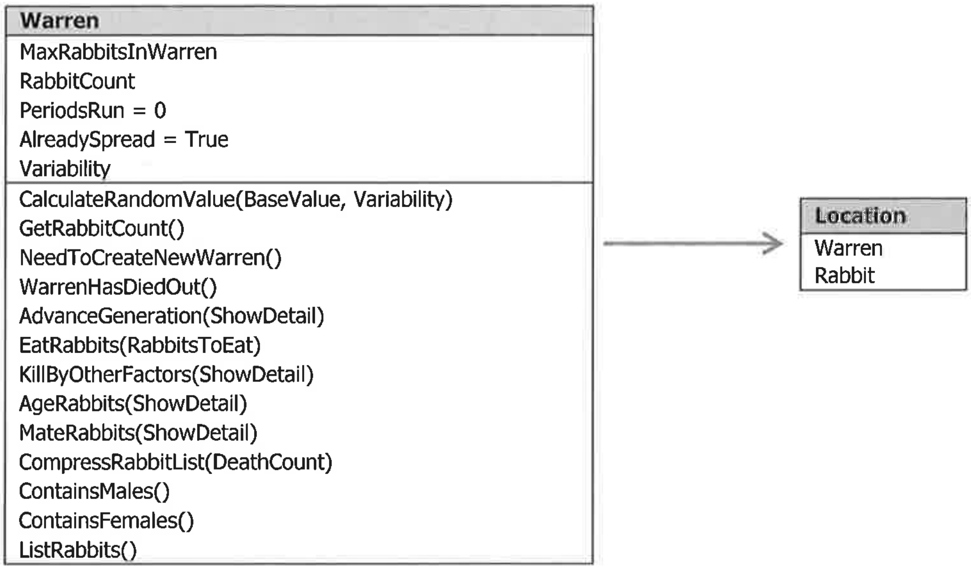
12. Why is it necessary to store the gender of the rabbits? [2 marks]

....

....

....

13. Identify six errors in the section of UML diagram below. [6 marks]



1 ....

2 ....

3 ....

4 ....

5 ....

6 ....

14. Create a UML diagram to show the relationship between rabbits, foxes and animals. All variables and methods must be shown. [11 marks]

15. What conditions are needed for a new warren to be created? [2 marks]

....

....

....

Programming Theory Questions

These questions refer to the Preliminary Material and require you to load the Skeleton Program, but do not require any additional programming.

1. Give an example of instantiation from the skeleton program. [1 mark]

2. State the name of an identifier(s) for the following:

   a. An array variable [1 mark]
   b. A subclass [1 mark]
   c. A parent class [1 mark]
   d. A class variable [1 mark]
   e. An accessor method [1 mark]
   f. A mutator method [1 mark]
   g. A variable used to store a whole number [1 mark]
   h. A Boolean variable [1 mark]
   i. Four constants that store a float [4 marks]

3 a. Two classes that have a composition aggregation relationship. [2 marks]

   b. Why is Warren to Rabbit not an example of association aggregation? [1 mark]

4. Are there any examples of polymorphism in the skeleton code? [1 mark]

5. State the name of an identifier for a procedure or function that is overridden in a subclass. [1 mark]

6. Look at the EatRabbits subroutine in the Warren class in the skeleton program. Why does the generation of a random rabbit need to be inside a repetition structure? [1 mark]

7. Look at the Warren class. Why has a named constant been used instead of a numeric value? [2 marks]

8. State the name of an identifier for an enumerated data type. [1 mark]

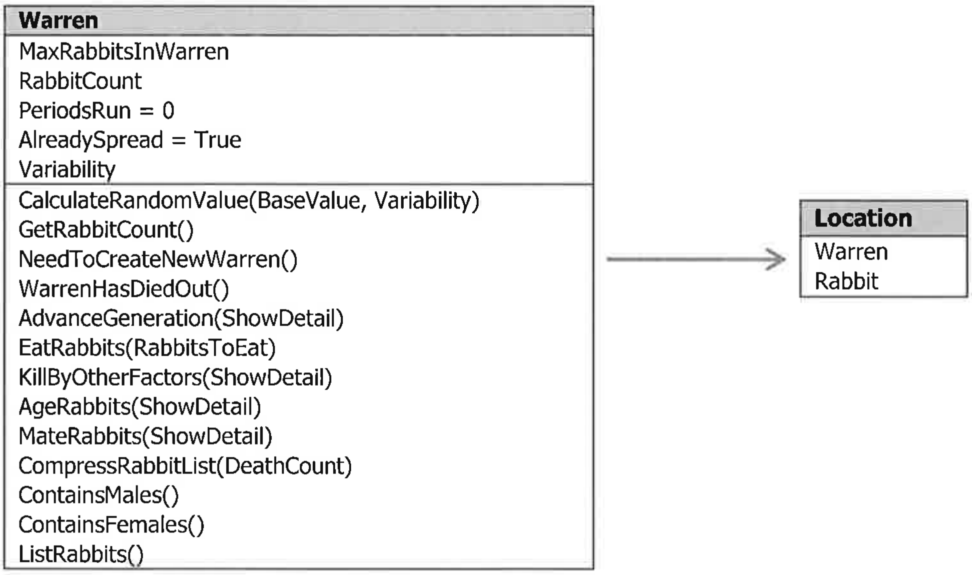
9. How could the Fox class be changed to make the foxes live longer? [1 mark]

10. What is the purpose of the variable AlreadySpread in the Warren class and how is it used? [4 marks]

11. What is the purpose of the method CompressRabbitList? [2 marks]

12. Why is it necessary to store the gender of the rabbits? [2 marks]

13. Identify six errors in the section of UML diagram below. [6 marks]



14. Create a UML diagram to show the relationship between rabbits, foxes and animals. All variables and methods must be shown. [11 marks]

15. What conditions are needed for a new warren to be created? [2 marks]

/50

Programming Exercises

The following require you to open the skeleton program and make modifications. They are written in examination style and illustrate how you should prepare your answers.

Question 1

This task refers to the Main procedure

Alter how the menu displays so that:

- There is a new option '3. Rabbit Paradise'
- The 'Exit' option is now numbered 4

Evidence you need to provide:

- Copy of your amended code
- Screen capture of it executing

5 marks

Question 2

This task refers to the Main procedure

Code option 3 so that when it is selected the simulation is run with the following parameters:

- A landscape size of 20
- 20 warrens
- 0 foxes
- Locations are not fixed
- Variability is 1

Evidence you need to provide:

- Copy of your amended code
- Screen capture of it executing

5 marks

Question 3

This task refers to the Simulation class

Add an option to the game menu:
'0. Advance 10 time periods hiding detail'

Code this option.

Evidence you need to provide:

- Copy of your amended code
- Screen capture of it executing

7 marks

Question 4

This task refers to the Rabbit class

Change *Rabbit's* constructor so that it receives in an extra variable that will allow the ratio of male to female rabbits to be altered. Use the identifier *genderRatio* for the new variable.

Set the default value to 50 so that the constructor can be called without specifying a value for *genderRatio.*

Evidence you need to provide:

- Copy of your amended code

2 marks

Question 5

This task refers to the Fox class

Add *Gender* to the *Fox* class.

Make the ratio of males to females 1 : 2.

Alter the *Inspect* method so that the gender of a fox is reported.

Change *ReproduceThisPeriod* so that only female foxes can reproduce.

Evidence you need to provide:

- Copy of your amended code
- Screen capture of an inspection of the Fox at 2,10

12 marks

Question 6

A new subclass must be created for this task, as well as changes to the createLandscapeAndAnimals procedure in Simulation

Create a subclass of *Warren* called a *GiantWarren.*

- A giant warren has a maximum capacity of 200 and can always spawn a new warren even if it has done so already.
- A giant warren has a default rabbit.
- Add a giant warren to the default game at position (11,4) with a starting population of 115.

Evidence you need to provide:

- Copy of your amended code
- Screen capture of a default simulation executing

11 marks

Question 7

A new subclass must be created for this task, as well as changes to the Location class and createLandscapeAndAnimals, drawLandscape and AdvanceTimePeriod procedures in Simulation

Create a *Den* class that can exist in a location.

- The den will spawn 1 new fox per 3 time periods.
- The den will store how many foxes it has created as a private instance variable.
- The fox will appear at a random position.
- If there is already a fox in this location, it is replaced by the new fox.
- Position the den at (2,3) in a default game.
- The den will be displayed on the map as a D plus the number of foxes it has spawned, e.g. D2.

Evidence you need to provide:

- Copy of your amended code
- Screen capture at time period 3 of a default game running

18 marks

Question 8

This tasks refers to the Fox class

The average age of death of foxes needs to be known.

- Create a class variable called *_TotalDeadFoxes* to store the total foxes who have died.
- Create a class variable called *_TotalFoxAge* to store the sum of the ages of all foxes who have died.
- When a fox dies, the *_TotalDeadFoxes* needs to be incremented and its age added to *_TotalFoxAge*.
- An accessor method in Fox called *getLifeExpect* will return the average age of a fox at death.
- A message stating 'The average life expectancy of a fox stands at X' should be printed under the landscape each time it is displayed.
- If no foxes have yet died, the default lifespan should be returned.

Evidence you need to provide:

- Copy of your amended code
- Screen capture of default simulation at time period 0
- Screen capture of default simulation at time period 4

13 marks

Question 9

This task refers to the Simulation class

Create a menu option in the simulation: '6. Find biggest warren'.

The coordinates of the biggest warren will then be displayed: 'Biggest warren at (X,Y) '.

Create a new procedure called findBiggest to search the warren array in a linear fashion and display the message.

Evidence you need to provide:
- Copy of your amended code
- Screen capture of option 6 running

12 marks

Question 10

This task refers to the Rabbit class

Make rabbit death probability go up by 10% with age.

Evidence you need to provide:
- Copy of your amended code
- Screen capture of a warren inspected (showing individual rabbits) at time period 2

2 marks

Question 11

This task requires changes to Warren and Simulation classes

Create a menu option: '7. Inspect all rabbits'.

It should display a list of all rabbits in all warrens, showing their details.

An accessor method to get the rabbits list out of a warren must be created.

Evidence you need to provide:
- Copy of your amended code
- Screen capture of option 7 running

13 marks

Question 12

This task requires changes to Simulation as well as creation of new classes

Beneath the warrens are secret tunnels connecting them. Not every warren is connected to every other. No warren is connected to more than two other warrens. This data must be stored in a *WarrenGraph.*

WarrenGraph
-nodes[]

+addNode(theNode)
+adjList()

Node
-selfX -selfY -leftBranchX -leftBranchY -rightBranchX -rightBranchY
+getCoord(l/r/s)

Each warren connected to another has the coordinates of itself and its connecting warrens stored in a node. *WarrenGraph* contains a list of all nodes. The procedure *getCoord* returns the x- and y-coordinates of these based on arguments (l)eft, (r)ight and (s)elf.

The *adjList* method displays an adjacency list and should be executed by a new option: '8. Display adjacency list'.

The following data should be used to initially populate the graph.

self	left	right
(1,1)	(2,8)	(9,7)
(2,8)	(13,4)	(1,1)
(9,7)	(1,1)	(13,4)
(13,4)	(9,7)	(2,8)

Evidence you need to provide:

- Copy of your amended code
- Screen capture of option 8 running

22 marks

Question 13

This task requires changes to Simulation and WarrenGraph

Create a new procedure in *WarrenGraph* called *adjMatrix.* It will display the graph as a matrix instead of a list. It will be executed by '9. Display adjacency matrix'. A 1 should be used to indicate a connecting burrow.

Evidence you need to provide:

- Copy of your amended code
- Screen capture of option 9 running

17 marks

Question 14

This task requires changes to WarrenGraph

Amend your solution for task 13 to replace the '1' with the actual distance between the nodes/warrens.

Use Pythagoras' theorem to calculate the distance between the two points.

Distances should be rounded to 1 decimal place.

Evidence you need to provide:

- Copy of your amended code for adjMatrix
- Screen shot of option 9 running

9 marks

Question 15

This task requires changes to Simulation and WarrenGraph

Create a procedure to find whether there is a route between two warrens.

It will be executed by Option 10.

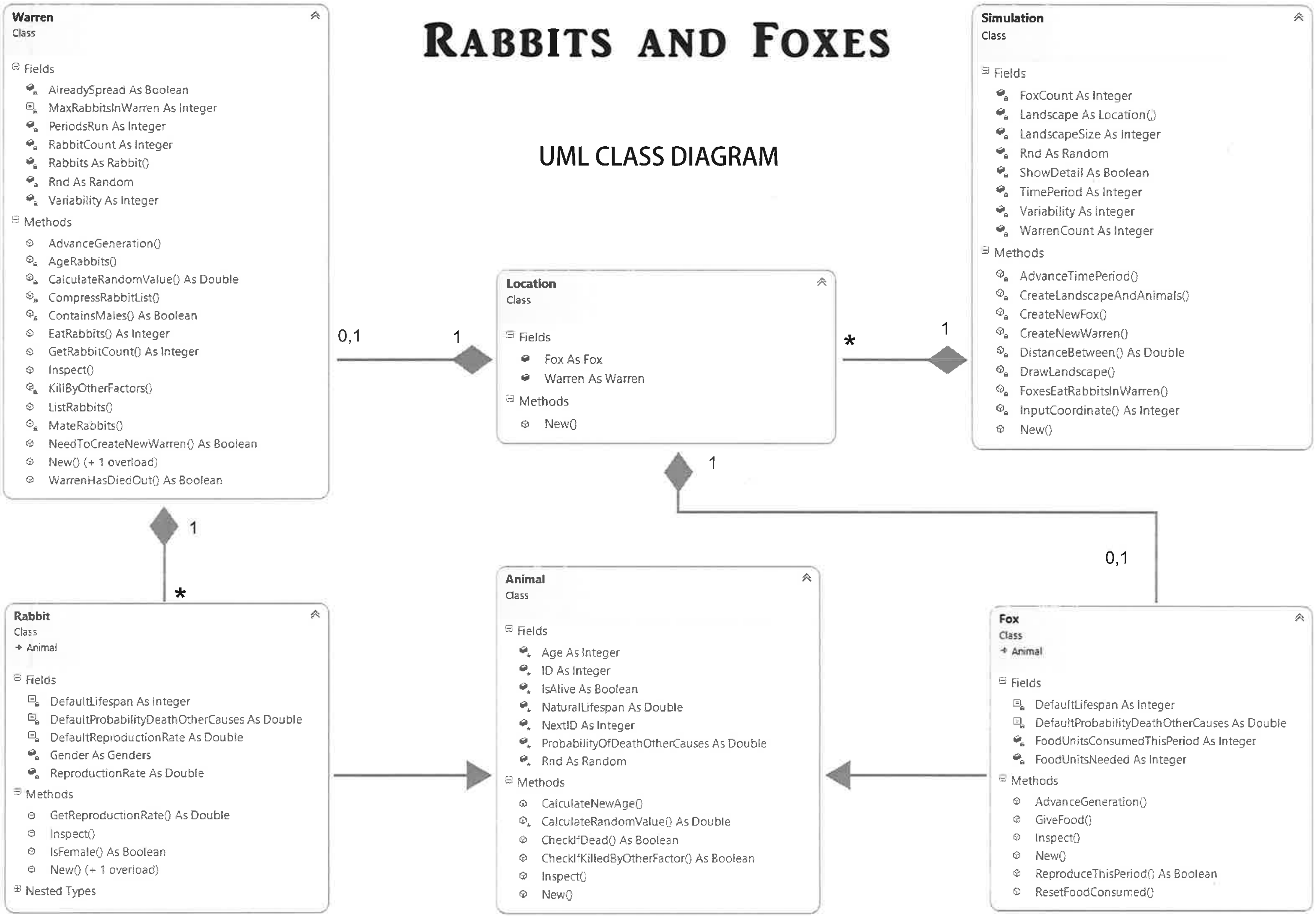
Evidence you need to provide:

- Copy of your code
- Screen capture of option 10 running showing no route between warrens
- Screen capture of option 10 running showing a route between warrens

13 marks

Rabbits and Foxes

UML CLASS DIAGRAM

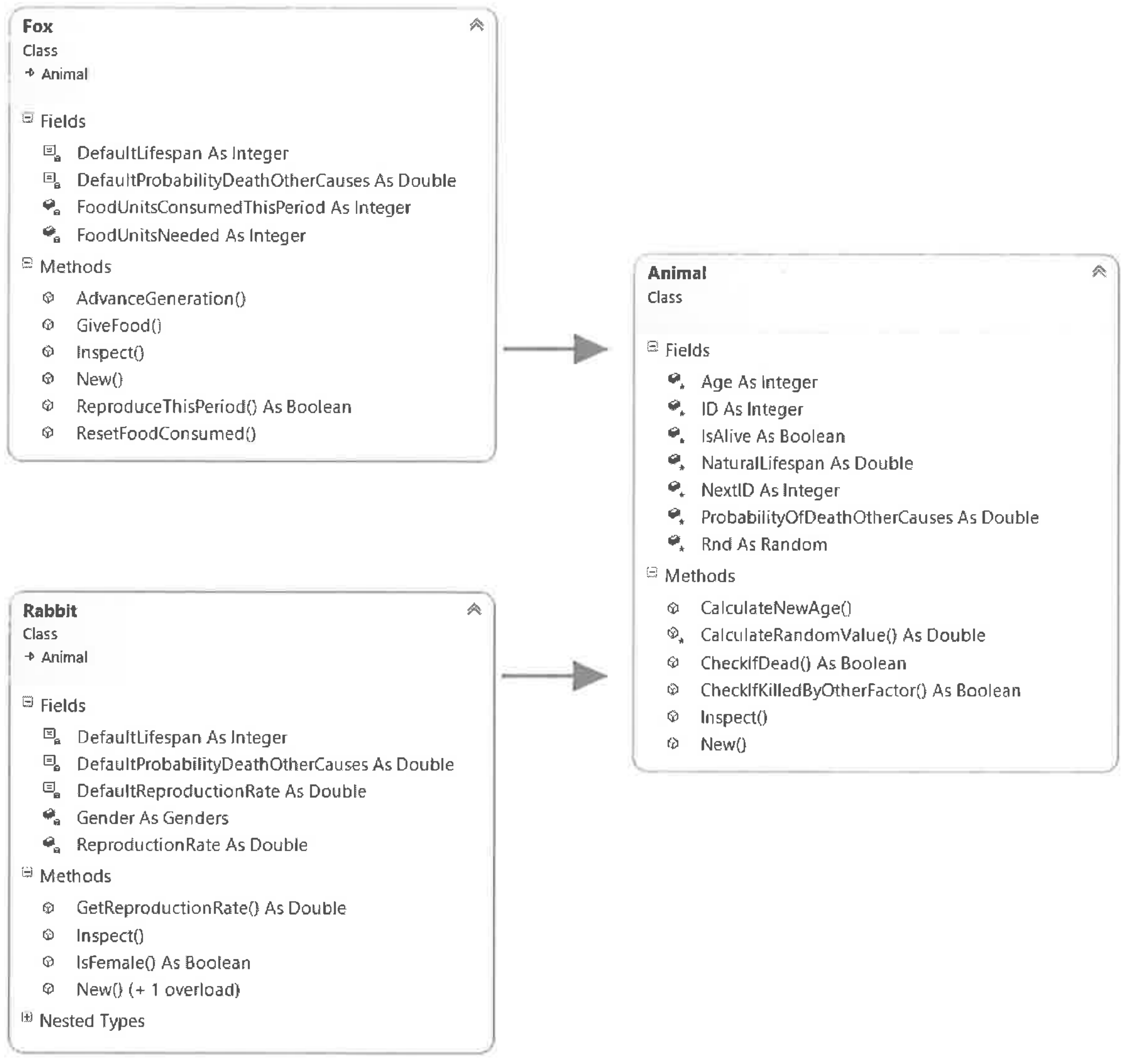


Programming Theory Questions (Suggested Answers)

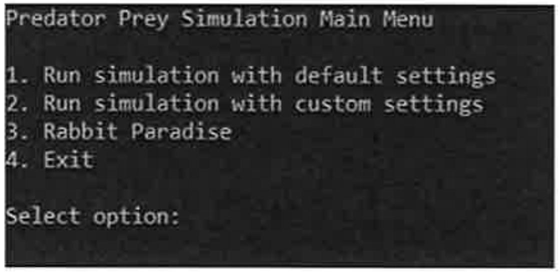
Q	Marking Guidance	Marks

1	Dim Sim As New Simulation(LandscapeSize, InitialWarrenCount, InitialFoxCount, Variability, FixedInitialLocations)
	Landscape(x, y).Warren = New Warren(Variability)
	Landscape(x, y).Fox = New Fox(Variability)
	Rabbits = New Rabbit(MaxRabbitsInWarren)
		1

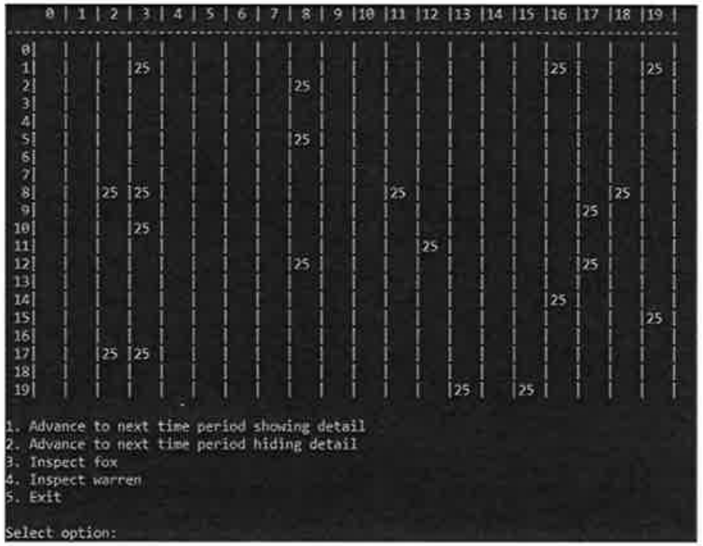
2a	Landscape / Rabbits	1
2b	Fox / Rabbit	1
2c	Animal	1
2d	NextID	1
2e	Any procedures/functions with Get at the start of the identifier	1
2f	Any procedures with Set at the start of the identifier	1
2g	MenuOption / LandscapeSize / InitialWarrenCount / InitialFoxCount / Variability Or any other location	1
2h	FixedInitialLocations OR ShowDetail OR AlreadySpread OR Males OR IsAlive	1
2i	DefaultProbabilityDeathOtherCauses ReproductionProbability DefaultReproductionRate DefaultProbabilityDeathOtherCauses	
3a	Location to Fox or Location to Warren or Warren to Rabbit (any correct pair for both marks)	2
3b	Rabbit objects cannot exist unless they have an associated Warren	1
4	Yes – the constructor for Rabbit	1
5	Inspect	1
6	To keep selecting a different rabbit at random until the required number of rabbits have been eaten	1
7	Makes the program code easier to understand / improves readability Makes it easier to update the program Makes it easier to change the maximum number of rabbits in a warren ANY 2	2
8	Gender	1
9	The DefaultLifeSpan constant needs to be increased from 7	1
10	It stores whether or not the warren has already created a new warren It stops the warren creating more than 1 new warren It is set to False by default It is set to True when a new warren is created	4
11	When rabbits are eaten or die they are removed from random positions in the rabbits list Compressing rabbits list removes the gaps	2
12	Only female rabbits can reproduce This therefore affects the calculation for how many new baby rabbits are born	2
13	Type and direction or arrow wrong Warren does not inherit from Location Location is associated to Warren Location stores warrens and/or foxes Location cannot store rabbits AlreadySpread should be set to False as default The constant MaxRabbitsInWarren has a default value of 99 Warren should contain a list of rabbits The Inspect() procedure is missing There is no function called ContainsFemales() in Warren ANY 6	6

Q	Marking Guidance	Marks
14	 **Fox** Class → Animal ⊟ Fields DefaultLifespan As Integer DefaultProbabilityDeathOtherCauses As Double FoodUnitsConsumedThisPeriod As Integer FoodUnitsNeeded As Integer ⊟ Methods AdvanceGeneration() GiveFood() Inspect() New() ReproduceThisPeriod() As Boolean ResetFoodConsumed() **Animal** Class ⊟ Fields Age As Integer ID As Integer IsAlive As Boolean NaturalLifespan As Double NextID As Integer ProbabilityOfDeathOtherCauses As Double Rnd As Random ⊟ Methods CalculateNewAge() CalculateRandomValue() As Double CheckIfDead() As Boolean CheckIfKilledByOtherFactor() As Boolean Inspect() New() **Rabbit** Class → Animal ⊟ Fields DefaultLifespan As Integer DefaultProbabilityDeathOtherCauses As Double DefaultReproductionRate As Double Gender As Genders ReproductionRate As Double ⊟ Methods GetReproductionRate() As Double Inspect() IsFemale() As Boolean New() (+ 1 overload) ⊞ Nested Types 1 mark for correct class name (×3) 1 mark for correct instance variables (×3) 1 mark for correct methods (×3) 1 mark for correct inheritance arrows (×2)	11
15	The number of rabbits in the warren must have reached the maximum allowed The warren cannot have already created a new warren	2
	TOTAL MARKS	**50**

Programming Exercises (Solutions)

<table>
<tr><th>Q</th><th>Example Solution</th><th>Suggested Marks</th></tr>
<tr><td>1</td><td>
<pre>
Sub Main()
    ...
      ...
      Console.WriteLine("1. Run simulation with default settings")
      Console.WriteLine("2. Run simulation with custom settings")
      Console.WriteLine("3. Rabbit Paradise")
      Console.WriteLine("4. Exit")
      Console.WriteLine()
      ...
        ...
      End If
    Loop While MenuOption <> 4
    Console.ReadKey()
  End Sub
</pre>

<pre>
Predator Prey Simulation Main Menu

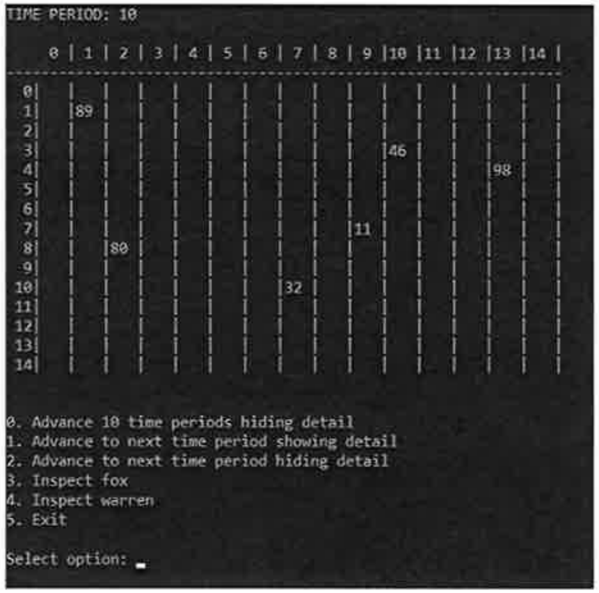
1. Run simulation with default settings
2. Run simulation with custom settings
3. Rabbit Paradise
4. Exit

Select option:
</pre>
</td><td>5 marks
• 1 mark for changing 3 to 4 in while loop condition
• 1 mark for print statement for rabbit paradise
• 1 mark for changing 3 to a 4 in print statement for exit
• 1 mark for options 3 and 4 displayed in correct order
• 1 mark for screen capture (1)</td></tr>
<tr><td>2</td><td>
<pre>
   If MenuOption = 1 Or MenuOption = 2 Or MenuOption = 3 Then
     ...
         ...
          FixedInitialLocations = True
        ElseIf MenuOption = 3 Then
          LandscapeSize = 20
          InitialWarrenCount = 20
          InitialFoxCount = 0
          Variability = 1
          FixedInitialLocations = False
        Else
</pre>

<pre>
     0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |10 |11 |12 |13 |14 |15 |16 |17 |18 |19 |

  0| | | | | | | | | | | | | | | | | | | | |
  1| | | |25 | | | | | | | | | | | | |25 | | |25 |
  2| | | | | | | | |25 | | | | | | | | | | | |
  3| | | | | | | | | | | | | | | | | | | | |
  4| | | | | | | | | | | | | | | | | | | | |
  5| | | | | | | | |25 | | | | | | | | | | | |
  6| | | | | | | | | | | | | | | | | | | | |
  7| | | | | | | | | | | | | | | | | | | | |
  8| | |25 |25 | | | | | | | |25 | | | | | | |25 | |
  9| | | | | | | | | | | | | | | | | |25 | | |
 10| | | |25 | | | | | | | | | | | | | | | | |
 11| | | | | | | | | | | | |25 | | | | | | | |
 12| | | | | | | | |25 | | | | | | | | |25 | | |
 13| | | | | | | | | | | | | | | | | | | | |
 14| | | | | | | | | | | | | | | | |25 | | | |
 15| | | | | | | | | | | | | | | | | | | |25 |
 16| | | | | | | | | | | | | | | | | | | | |
 17| | |25 |25 | | | | | | | | | | | | | | | | |
 18| | | | | | | | | | | | | | | | | | | | |
 19| | | | | | | | | | | | | |25 | |25 | | | | |

1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit

Select option:
</pre>
</td><td>5 marks
• 1 mark for changing IF statement to allow MenuOption to be 3
• 1 mark for adding an ElseIf and correct condition
• 2 marks for assigning variables the correct values (−1 per mistake)
• 1 mark for screen capture (1)</td></tr>
</table>

<table>
<tr><th>Q</th><th>Example Solution</th><th>Suggested Marks</th></tr>
<tr><td>3</td><td>Class Simulation
  ...
    ...
    Do
      Console.WriteLine()
      Console.WriteLine("0. Advance 10 time periods hiding detail")
      Console.WriteLine("1. Advance to next time period showing detail")
      ...
      MenuOption = CInt(Console.ReadLine())
      If MenuOption = 0 Then
        ShowDetail = False
          For x = 1 To 10
            TimePeriod += 1
            AdvanceTimePeriod()
        Next
      End If
      If MenuOption = 1 Then
        ...

TIME PERIOD: 10
     0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |10 |11 |12 |13 |14 |

  0| | | | | | | | | | | | | | | |
  1| |89 | | | | | | | | | | | | | |
  2| | | | | | | | | | | | | | | |
  3| | | | | | | | | | |46 | | | | |
  4| | | | | | | | | | | | | |98 | |
  5| | | | | | | | | | | | | | | |
  6| | | | | | | | | | | | | | | |
  7| | | | | | | | | |11 | | | | | |
  8| | |80 | | | | | | | | | | | | |
  9| | | | | | | | | | | | | | | |
 10| | | | | | | |32 | | | | | | | |
 11| | | | | | | | | | | | | | | |
 12| | | | | | | | | | | | | | | |
 13| | | | | | | | | | | | | | | |
 14| | | | | | | | | | | | | | | |

0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit

Select option: _</td><td>7 marks
• *6 marks for changes shown (1 mark per line – excluding Next and End If)*
• *1 mark for screen capture*</td></tr>
<tr><td>4</td><td>   Public Sub New(ByVal Variability As Integer, **ByVal GenderRatio As Integer**)
      MyBase.New(DefaultLifespan, DefaultProbabilityDeathOtherCauses, Variability)
      ReproductionRate = DefaultReproductionRate * MyBase.CalculateRandomValue(100, Variability) / 100
      If Rnd.Next(0, 100) < **GenderRatio** Then
        Gender = Genders.Male
      ...</td><td>2 marks
• *2 marks for changes shown (1 mark each)*</td></tr>
<tr><td>5</td><td> Class Fox
    Inherits Animal
    Enum Genders
      Male
      Female
    End Enum
    ...
    ...
    Private Gender As Genders
    Public Sub New(ByVal Variability As Integer)</td><td>12 marks
• *4 marks for successfully adding gender to a fox (1 mark per line)*
• *3 marks for stopping male foxes from breeding*
• *4 marks for outputting the gender of a fox in an inspection*</td></tr>
</table>

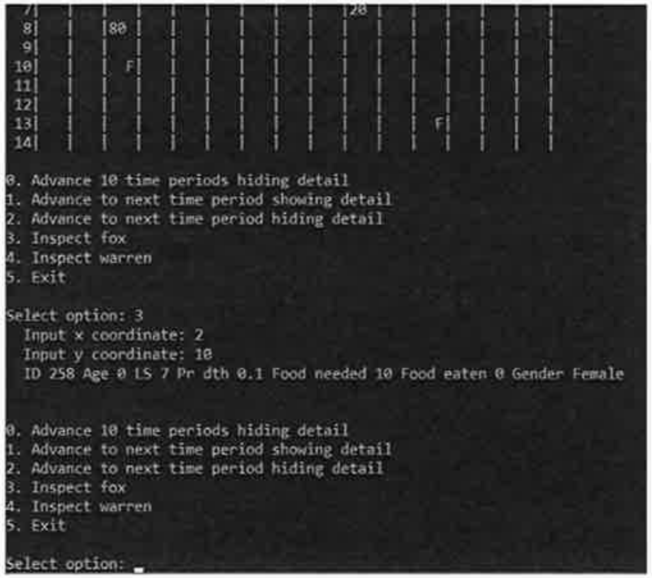
Q	Example Solution	Suggested Marks

5
(cont.)
	`MyBase.New(DefaultLifespan, DefaultProbabilityDeathOtherCauses, Variability)`
	`FoodUnitsNeeded = CInt(10 * MyBase.CalculateRandomValue(100, Variability) / 100)`
	`If Rnd.Next(1, 3) = 1 Then`
	`Gender = Genders.Male`
	`Else`
	`Gender = Genders.Female`
	`End If`
	`End Sub`
	`...`
		• *1 mark for screen capture*

```
  MyBase.New(DefaultLifespan, DefaultProbabilityDeathOtherCauses, Variability)
  FoodUnitsNeeded = CInt(10 * MyBase.CalculateRandomValue(100, Variability) / 100)
  If Rnd.Next(1, 3) = 1 Then
    Gender = Genders.Male
  Else
    Gender = Genders.Female
  End If
End Sub
...
```

```
Public Function ReproduceThisPeriod() As Boolean
  Const ReproductionProbability As Double = 0.25
  If Rnd.Next(0, 100) < ReproductionProbability * 100 And Gender = Genders.Female Then
    Return True
  ...
```

```
Public Overrides Sub Inspect()
  ...
  ...
  Console.Write("Food eaten " & FoodUnitsConsumedThisPeriod & " ")
  If Gender = Genders.Female Then
    Console.WriteLine("Gender Female")
  Else
    Console.WriteLine("Gender Male")
  End If
  Console.WriteLine()
...
```



```
  7|   |   |   |   |   |   |   |   |   |28 |   |   |   |   |   |
  8|   |   |80 |   |   |   |   |   |   |   |   |   |   |   |   |
  9|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 10|   |   |  F|   |   |   |   |   |   |   |   |   |   |   |   |
 11|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 12|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 13|   |   |   |   |   |   |   |   |   |   |   |  F|   |   |   |
 14|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit

Select option: 3
  Input x coordinate: 2
  Input y coordinate: 10
  ID 258 Age 0 LS 7 Pr dth 0.1 Food needed 10 Food eaten 0 Gender Female


0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit

Select option: _
```

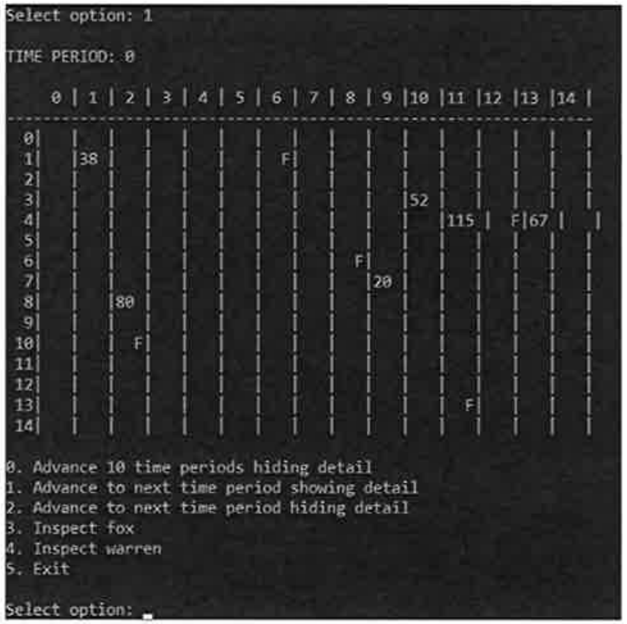
Q	Example Solution	Suggested Marks
6		11 marks

```
Class GiantWarren
    Inherits Warren
    Public Sub New(ByVal Variability As Integer, ByVal RabbitCount As Integer)
      MyBase.New(Variability)
      Rabbits = New Rabbit(200) {}
       RabbitCount = GiantRabbitCount
      For r = 0 To RabbitCount - 1
        Rabbits(r) = New Rabbit(Variability)
      Next
    End Sub
  End Class
```

```
Private Sub CreateLandscapeAndAnimals(ByVal InitialWarrenCount As Integer, ByVal InitialFoxCount As Integer, ByVal
FixedInitialLocations As Boolean)
      ...
        ...
        Landscape(10, 3).Warren = New Warren(Variability, 52)
        Landscape(11, 4).Warren = New GiantWarren(Variability, 115)
        Landscape(13, 4).Warren = New Warren(Variability, 67)
        ...
```

Plus: *Warren instance variables need to be protected and not private:*

```
  Protected Const MaxRabbitsInWarren As Integer = 99
  Protected Rabbits() As Rabbit
  Protected RabbitCount As Integer = 0
  Protected PeriodsRun As Integer = 0
  Protected AlreadySpread As Boolean = False
  Protected Variability As Integer
  Protected Shared Rnd As New Random()
```



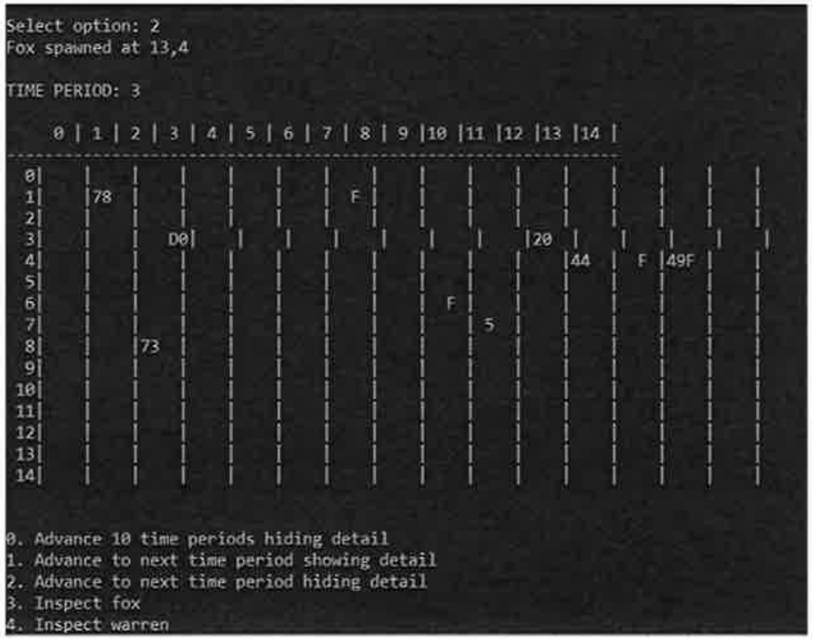
```
Select option: 1

TIME PERIOD: 0

     0  |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |  9  | 10  | 11  | 12  | 13  | 14  |
--------------------------------------------------------------------------------------------
 0|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
 1|     |38   |     |     |     |     |    F|     |     |     |     |     |     |     |     |
 2|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
 3|     |     |     |     |     |     |     |     |     |     |52   |     |     |     |     |
 4|     |     |     |     |     |     |     |     |     |     |     |115  |    F|67   |     |
 5|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
 6|     |     |     |     |     |     |     |     |    F|     |     |     |     |     |     |
 7|     |     |     |     |     |     |     |     |     |20   |     |     |     |     |     |
 8|     |     |80   |     |     |     |     |     |     |     |     |     |     |     |     |
 9|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
10|     |     |    F|     |     |     |     |     |     |     |     |     |     |     |     |
11|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
12|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
13|     |     |     |     |     |     |     |     |     |     |     |    F|     |     |     |
14|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit

Select option: _
```

- *4 marks for GiantWarren class (1 for signature plus 3 for each other highlighted change)*
- *3 marks for changes to CreateLandscapeAndAnimals (1 for creating a GiantWarren, 1 for putting it in the correct position, 1 for changing the warren count)*
- *3 marks for changing Warren instance variables to protected and not private*
- *1 for screen capture showing the GiantWarren*

<table>
<tr><th>Q</th><th>Example Solution</th><th>Suggested Marks</th></tr>
<tr><td>7</td><td>

```
Class Den
    Private FoxesSpawned As Integer

    Public Sub New()
        FoxesSpawned = 0
    End Sub

    Public Function Spawn() As Fox
        Return New Fox(50)
    End Function

    Public Function GetSymbol() As String
        Return "D" + FoxesSpawned.ToString
    End Function
End Class
```

```
Class Location
    Public Fox As Fox
    Public Warren As Warren
    Public Den As Den

    Public Sub New()
        Fox = None
        Warren = None
        Den = None
    End Sub
End Class
```



```
Select option: 2
Fox spawned at 13,4

TIME PERIOD: 3

     0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |10 |11 |12 |13 |14 |
----------------------------------------------------------------
  0|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
  1|    |78  |    |    |    |    |  F |    |    |    |    |    |    |    |    |
  2|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
  3|    |    |   D0|    |    |    |    |    |    |    |20  |    |    |    |    |
  4|    |    |    |    |    |    |    |    |    |    |    |44  |  F |49F |    |
  5|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
  6|    |    |    |    |    |    |    |    |  F |    |    |    |    |    |    |
  7|    |    |    |    |    |    |    |    |    | 5  |    |    |    |    |    |
  8|    |    |73  |    |    |    |    |    |    |    |    |    |    |    |    |
  9|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
 10|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
 11|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
 12|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
 13|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
 14|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
```

</td><td>

18 marks

- *5 marks for Den class (1 for signature, 1 for instance variable, 1 for constructor, 1 for spawn, 1 for getsymbol)*
- *2 marks for change to Location class*
- *3 marks for changes to DrawLandscape*
- *1 mark for adding a Den in CreateLandscape*
- *6 marks for adding the spawn command to AdvanceTimePeriod*
- 1 mark for *screen capture*

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

Q	Example Solution	Suggested Marks
7 (cont.)		

```
Private Sub DrawLandscape()
  ...
      ...
      If Not Landscape(x, y).Fox Is None Then
        Console.Write("F")
      Else
        Console.Write(" ")
      End If
      If Not Landscape(x, y).Den Is None Then
        Console.Write(Landscape(x, y).Den.GetSymbol())
      Else
        Console.Write(" ")
      End If
      Console.Write("|")
    Next
```

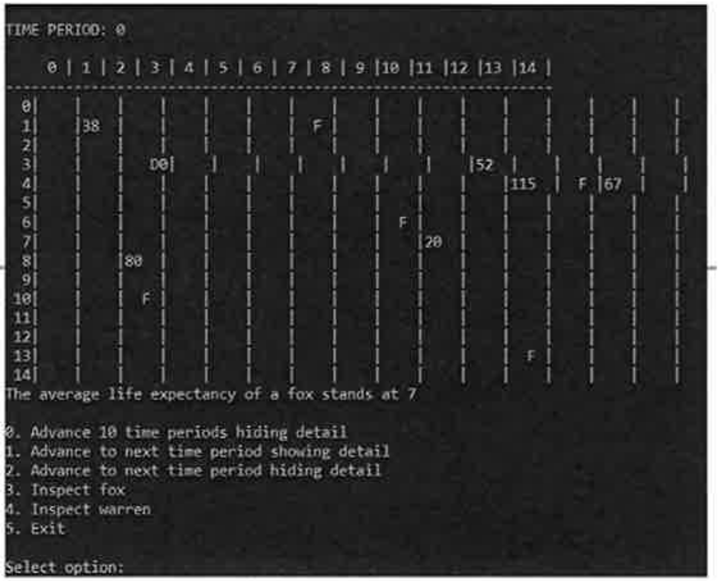
```
Private Sub CreateLandscapeAndAnimals(ByVal InitialWarrenCount As Integer, ByVal InitialFoxCount As Integer, ByVal
FixedInitialLocations As Boolean)
  ...
    ...
    FoxCount = 5
    Landscape(2, 3).Den = New Den()
```

```
Private Sub AdvanceTimePeriod()
  Dim NewFoxCount As Integer = 0
  If TimePeriod Mod 3 = 0 Then
    Dim x, y As Integer
    x = Rnd.Next(1, LandscapeSize - 1)
    y = Rnd.Next(1, LandscapeSize - 1)
    Landscape(x, y).Fox = Landscape(2, 3).Den.Spawn()
    Console.WriteLine("Fox spawned at " + x.ToString + "," + y.ToString)
  End If
```

Q	Example Solution	Suggested Marks
8	Class Fox (see code below)	13 marks

```
Class Fox
   ...
   ...
   Private Gender As Genders
   Private Shared TotalDeadFoxes As Double = 10
   Private Shared TotalAge As Double = 70
```

```
   Public Sub AdvanceGeneration(ByVal ShowDetail As Boolean)
      ...
      If FoodUnitsConsumedThisPeriod = 0 Then
         ...
         ...
      End If
      If Not IsAlive Then
         TotalDeadFoxes +=
         TotalAge = TotalAge + Age
      End If
   End Sub

   Public Function GetLifeExpect() As Double
      Return TotalAge / TotalDeadFoxes
   End Function
```



```
TIME PERIOD: 0

    0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |10 |11 |12 |13 |14 |
--------------------------------------------------------------
 0|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 1|   |38 |   |   |   |   |   | F |   |   |   |   |   |   |   |
 2|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 3|   |   | D0|   |   |   |   |   |   |   |52 |   |   |   |   |
 4|   |   |   |   |   |   |   |   |   |   |   |115|  F|67 |   |
 5|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 6|   |   |   |   |   |   |   |   | F |   |   |   |   |   |   |
 7|   |   |   |   |   |   |   |   |   |20 |   |   |   |   |   |
 8|   |   |80 |   |   |   |   |   |   |   |   |   |   |   |   |
 9|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
10|   |   | F |   |   |   |   |   |   |   |   |   |   |   |   |
11|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
12|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
13|   |   |   |   |   |   |   |   |   |   |   | F |   |   |   |
14|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
The average life expectancy of a fox stands at 7

0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit

Select option:
```

Add to end of DrawLandscape in Simulation:

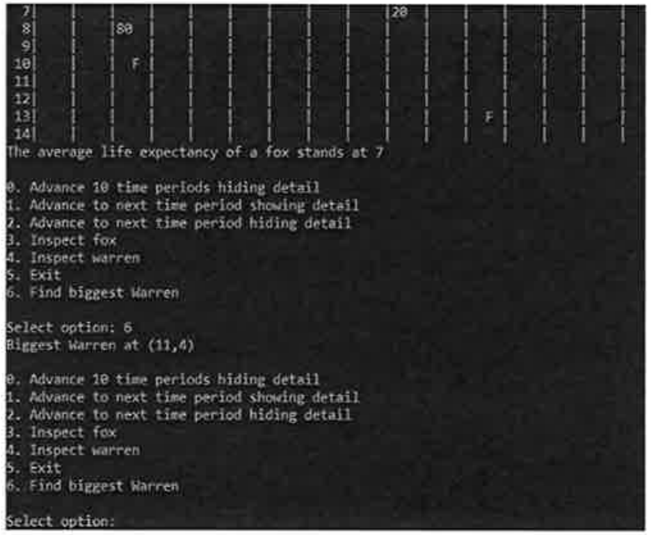
```
Dim lifeExpect As Double
Dim theFox = New Fox(50)
lifeExpect = theFox.GetLifeExpect
Console.WriteLine("The average life expectancy of a fox stands at " + lifeExpect.ToString)
```

13 marks

- *4 marks for changes to DrawLandscape (1 per line)*
- *2 marks for class variables in Fox*
- *3 marks for changes to AdvanceGeneration (1 for If, 1 for incrementing, 1 for adding age to total)*
- *2 marks for creating getLifeExpect (1 for signature, 1 for line of code)*
- *2 marks for screen captures*

<table>
<tr><th>Q</th><th>Example Solution</th><th>Suggested Marks</th></tr>
<tr><td>9</td><td>

```
Public Sub New(ByVal LandscapeSize As Integer, ByVal InitialWarrenCount As Integer, ByVal InitialFoxCount As Integer, ByVal
Variability As Integer, ByVal FixedInitialLocations As Boolean)
  ...
    Console.WriteLine("5. Exit")
    Console.WriteLine("6. Find biggest Warren")
    ...
    ...
    If MenuOption = 6 Then
      findBiggest()
    End If
  ...

Private Sub findBiggest()
  Dim biggestX, biggestY, biggestSize As Integer
  biggestSize = -1
  For x = 0 To LandscapeSize - 1
    For y = 0 To LandscapeSize - 1
      If Not Landscape(x, y).Warren Is None Then
        If Landscape(x, y).Warren.GetRabbitCount > biggestSize Then
          biggestX = x
          biggestY = y
          biggestSize = Landscape(x, y).Warren.GetRabbitCount
        End If
      End If
    Next
  Next
  Console.WriteLine("Biggest Warren at (" + biggestX.ToString + "," + biggestY.ToString + ")")
End Sub
```



```
  7|    |    |    |    |    |    |    |    |    |20  |    |    |    |    |    |
  8|    |    |80  |    |    |    |    |    |    |    |    |    |    |    |    |
  9|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
 10|    |    |  F |    |    |    |    |    |    |    |    |    |    |    |    |
 11|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
 12|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
 13|    |    |    |    |    |    |    |    |    |    |    |  F |    |    |    |
 14|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
The average life expectancy of a fox stands at 7

0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit
6. Find biggest Warren

Select option: 6
Biggest Warren at (11,4)

0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit
6. Find biggest Warren

Select option:
```

</td><td>12 marks

• <em>1 mark for Print statement in menu</em>

• <em>2 marks for IF statement and procedure call when a 6 is entered</em>

• <em>1 mark for findBiggest signature</em>

• <em>1 mark for creating and initialising variables (to a sentinel value) to store data about the current biggest warren</em>

• <em>2 marks for x and y loops</em>

• <em>1 mark for checking that a warren is stored at x,y</em>

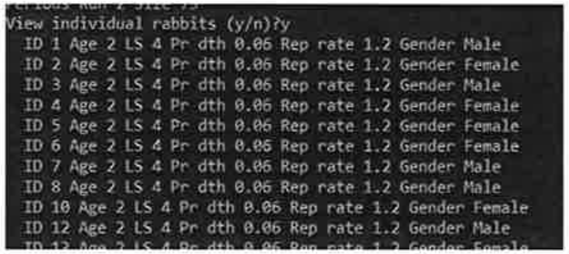
• <em>1 mark for checking if warren is bigger than current biggest</em>

• <em>1 mark for assigning x,y, and size of a new biggest warren</em>

• <em>1 mark for displaying correct message</em>

• <em>1 mark for screen capture</em></td></tr>
<tr><td>10</td><td>

```
 Public Overrides Sub CalculateNewAge()
  MyBase.CalculateNewAge()
  ProbabilityOfDeathOtherCauses = ProbabilityOfDeathOtherCauses * 1.1
End Sub
```



```
View individual rabbits (y/n)?y
  ID 1 Age 2 LS 4 Pr dth 0.06 Rep rate 1.2 Gender Male
  ID 2 Age 2 LS 4 Pr dth 0.06 Rep rate 1.2 Gender Female
  ID 3 Age 2 LS 4 Pr dth 0.06 Rep rate 1.2 Gender Male
  ID 4 Age 2 LS 4 Pr dth 0.06 Rep rate 1.2 Gender Female
  ID 5 Age 2 LS 4 Pr dth 0.06 Rep rate 1.2 Gender Female
  ID 6 Age 2 LS 4 Pr dth 0.06 Rep rate 1.2 Gender Female
  ID 7 Age 2 LS 4 Pr dth 0.06 Rep rate 1.2 Gender Male
  ID 8 Age 2 LS 4 Pr dth 0.06 Rep rate 1.2 Gender Male
  ID 10 Age 2 LS 4 Pr dth 0.06 Rep rate 1.2 Gender Female
  ID 12 Age 2 LS 4 Pr dth 0.06 Rep rate 1.2 Gender Male
  ID 13 Age 2 LS 4 Pr dth 0.06 Rep rate 1.2 Gender Female
```

</td><td>2 marks

• <em>1 mark for incrementing ProbabilityOfDeathOtherCauses by 0.1 when rabbits age</em>

• <em>1 mark for screen capture showing rabbits with different death probabilities</em></td></tr>
</table>

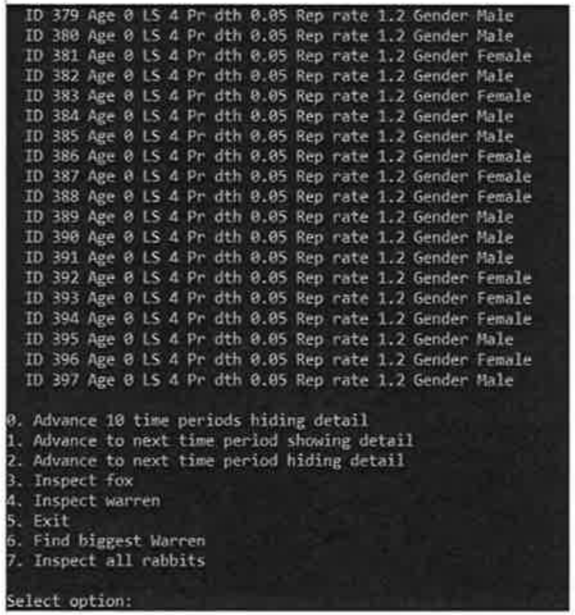
Q	Example Solution	Suggested Marks
11	*Add to class Warren:* (see code below)	13 marks

Add to class Warren:

```
Public Function getRabbits() As Rabbit()
   Return Rabbits
End Function
```

Add to class Simulation:

```
Console.WriteLine("7. Inspect all rabbits")
 Console.WriteLine()
 Console.Write("Select option: ")
 MenuOption = CInt(Console.ReadLine())
If MenuOption = 7 Then
   Dim AllRabbits() As Rabbit
   'get all rabbits
   For x = 0 To LandscapeSize - 1
      For y = 0 To LandscapeSize - 1
         If Not Landscape(x, y).Warren Is None Then
            AllRabbits = Landscape(x, y).Warren.getRabbits()
            'display all rabbits
            For i = 0 To AllRabbits.Length - 1
               Try
                  AllRabbits(i).Inspect()
               Catch ex As Exception
                  'catch null rabbits
               End Try
            Next
         End If
      Next
   Next
End If
```



```
ID 379 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Male
ID 380 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Male
ID 381 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Female
ID 382 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Male
ID 383 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Female
ID 384 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Male
ID 385 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Male
ID 386 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Female
ID 387 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Female
ID 388 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Female
ID 389 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Male
ID 390 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Male
ID 391 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Male
ID 392 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Female
ID 393 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Female
ID 394 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Female
ID 395 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Male
ID 396 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Female
ID 397 Age 0 LS 4 Pr dth 0.05 Rep rate 1.2 Gender Male

0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit
6. Find biggest Warren
7. Inspect all rabbits

Select option:
```

Suggested Marks: 13 marks

- *2 marks for creating getRabbits in Warren (1 for signature, 1 for returning Rabbits)*
- *2 marks for changes to the menu in simulation (1 per correct line)*
- *1 mark for creating an array to hold all the rabbits*
- *2 marks for nested for loops to check every location*
- *1 mark for if statement checking if warren is in that location*
- *1 mark for getting rabbits from a warren*
- *1 mark for looping through all rabbits*
- *1 mark for inspecting rabbits*
- *1 mark for try catch*
- *1 mark for screen capture*

<table>
<tr><th>Q</th><th>Example Solution</th><th>Suggested Marks</th></tr>
<tr><td>12</td><td><pre>
Class WarrenGraph
    Private Nodes As Node()

    Public Sub New()
        Dim n1 As New Node(1, 1, 2, 8, 9, 7)
        Dim n2 As New Node(2, 8, 13, 4, 1, 1)
        Dim n3 As New Node(9, 7, 1, 1, 13, 4)
        Dim n4 As New Node(13, 4, 9, 7, 2, 8)
        Nodes = New Node() {n1, n2, n3, n4}
    End Sub

    Public Sub AdjList()
        Console.WriteLine()
        Console.WriteLine("Self" + vbTab + "Left" + vbTab + "Right")
        For index = 0 To Nodes.Length - 1
            Console.WriteLine(Nodes(index).getCoord("s") + vbTab + Nodes(index).getCoord("l") + vbTab + Nodes(index).getCoord("r"))
        Next
    End Sub
  End Class

  Class Node
    Private selfX As Integer
    Private selfY As Integer
    Private leftX As Integer
    Private leftY As Integer
    Private rightX As Integer
    Private rightY As Integer

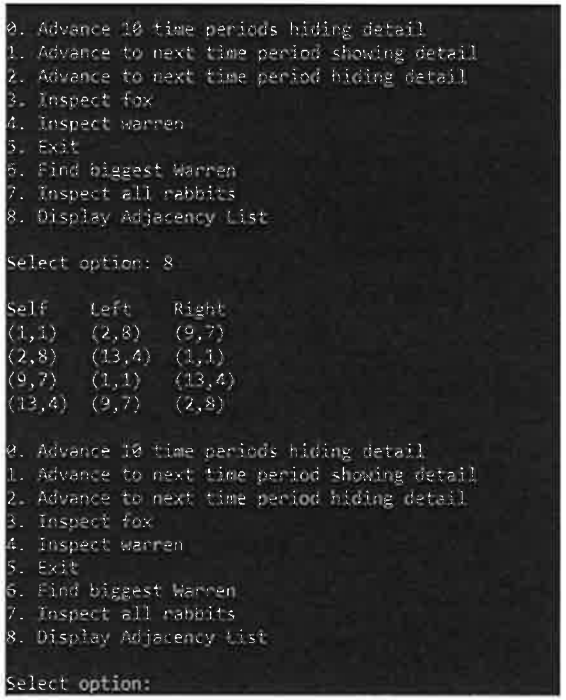
    Public Sub New(sx As Integer, sy As Integer, lx As Integer, ly As Integer, rx As Integer, ry As Integer)
        selfX = sx
        selfY = sy
        leftX = lx
        leftY = ly
        rightX = rx
        rightY = ry
    End Sub
</pre></td><td>22 marks
• <i>5 marks for creating and populating graph (1 for graph, 1 per node)</i>
• <i>3 marks for changes to menu</i>
• <i>6 marks for Node class (1 for signature, 2 for constructor, 3 marks for getCoord (1 for each arg))</i>
• <i>7 marks for WarrenGraph (1 for class signature, 2 for constructor, 2 for addNode procedure, 2 for adjList procedure)</i>
• <i>1 mark for screen capture</i></td></tr>
</table>

Q	Example Solution	Suggested Marks
12 (cont.)		

```
    Public Function getCoord(ByVal branch As String) As String
      If branch.Equals("l") Then
        Return ("(" + leftX.ToString + "," + leftY.ToString + ")")
      ElseIf branch.Equals("r") Then
        Return ("(" + rightX.ToString + "," + rightY.ToString + ")")
      Else
        Return ("(" + selfX.ToString + "," + selfY.ToString + ")")
      End If
    End Function
  End Class
```

Changes to class Simulation:

```
        ...
        Console.WriteLine("8. Display Adjacency List")
        Console.WriteLine()
        Console.Write("Select option: ")
        MenuOption = CInt(Console.ReadLine())
        If MenuOption = 8 Then
          Dim theGraph As New WarrenGraph()
          theGraph.AdjList()
        End If
```



```
0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit
6. Find biggest Warren
7. Inspect all rabbits
8. Display Adjacency List

Select option: 8

Self    Left    Right
(1,1)   (2,8)   (9,7)
(2,8)   (13,4)  (1,1)
(9,7)   (1,1)   (13,4)
(13,4)  (9,7)   (2,8)

0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit
6. Find biggest Warren
7. Inspect all rabbits
8. Display Adjacency List

Select option:
```

Q | **Example Solution** | **Suggested Marks**

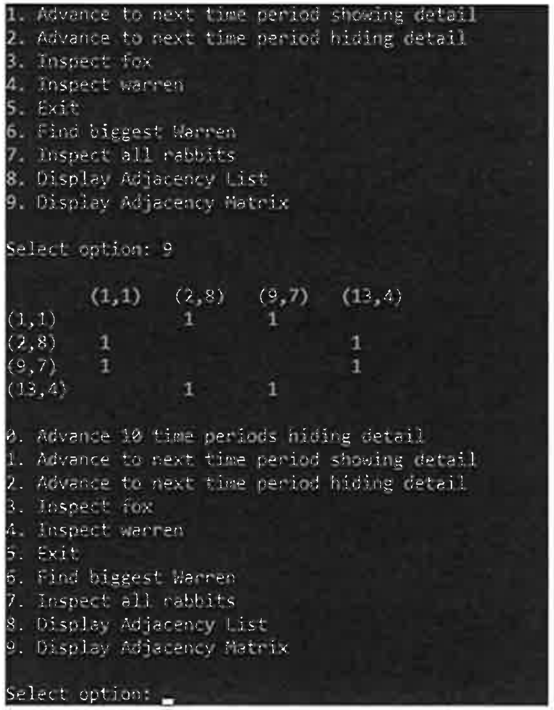
13

Changes to the menu:

```
...
Console.WriteLine("9. Display Adjacency Matrix")
Console.WriteLine()
Console.Write("Select option: ")
MenuOption = CInt(Console.ReadLine())
If MenuOption = 9 Then
    Dim theGraph As New WarrenGraph()
    theGraph.AdjMatrix()
End If
```

adjMatrix procedure in WarrenGraph:

```
Public Sub AdjMatrix()
    Dim theHeadings(Nodes.Length) As String
    Console.WriteLine()
    Console.Write(vbTab)
    For index = 0 To Nodes.Length - 1
        Console.Write(Nodes(index).getCoord("s") + vbTab)
        theHeadings(index) = Nodes(index).getCoord("s")
    Next
    Console.WriteLine()
    For index1 = 0 To Nodes.Length - 1
        Console.Write(Nodes(index1).getCoord("s") + vbTab)
        For index2 = 0 To Nodes.Length - 1
            If (Nodes(index2).getCoord("l") = theHeadings(index1)) Or (Nodes(index2).getCoord("r") = theHeadings(index1)) Then
                Console.Write(" 1 " + vbTab)
            Else
                Console.Write(vbTab)
            End If
        Next
        Console.WriteLine()
    Next
End Sub
```



```
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit
6. Find biggest Warren
7. Inspect all rabbits
8. Display Adjacency List
9. Display Adjacency Matrix

Select option: 9

         (1,1)   (2,8)   (9,7)   (13,4)
(1,1)             1       1
(2,8)     1                       1
(9,7)     1                       1
(13,4)            1       1

0. Advance 10 time periods hiding detail
1. Advance to next time period showing detail
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit
6. Find biggest Warren
7. Inspect all rabbits
8. Display Adjacency List
9. Display Adjacency Matrix

Select option: _
```

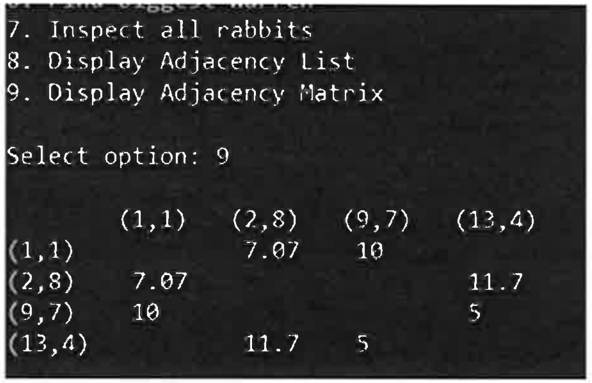
17 marks

- *3 marks for changes to menu*
- *13 marks for creating adjMatrix (1 mark for displaying column headings, 1 mark for displaying row labels correctly, 3 marks for efficient code (loops), 2 marks per correct row displayed).*
  NO CREDIT SHOULD BE GIVEN FOR HARD-CODED SOLUTIONS (the computer must work out the matrix each time the code is run)
- *1 mark for screen capture*

Q	Example Solution	Suggested Marks

14

```
Public Sub AdjMatrix()
    Dim theHeadings(Nodes.Length) As String
    Console.WriteLine()
    Console.Write(vbTab)
    For index = 0 To Nodes.Length - 1
        Console.Write(Nodes(index).getCoord("s") + vbTab)
        theHeadings(index) = Nodes(index).getCoord("s")
    Next
    Console.WriteLine()
    For index1 = 0 To Nodes.Length - 1
        Console.Write(Nodes(index1).getCoord("s") + vbTab)
        For index2 = 0 To Nodes.Length - 1
            If (Nodes(index2).getCoord("l") = theHeadings(index1)) Or (Nodes(index2).getCoord("r") = theHeadings(index1)) Then
                Dim distance As Double
                Dim x1, x2, y1, y2 As Double
                x1 = theHeadings(index1).IndexOf(",")
                y1 = Double.Parse(theHeadings(index1).Substring(x1 + 1, ((theHeadings(index1).Length - (x1 + 2)))))
                x1 = Double.Parse(theHeadings(index1).Substring(1, x1 - 1))
                Dim coord2 As String = Nodes(index2).getCoord("s")
                x2 = coord2.IndexOf(",")
                y2 = Double.Parse(coord2.Substring(x2 + 1, ((coord2.Length - x2 - 2))))
                x2 = Double.Parse(coord2.Substring(1, x2 - 1))
                distance = (Math.Sqrt(Math.Pow(Math.Abs(x2 - x1), 2) + Math.Pow(Math.Abs(y2 - y1), 2)))
                distance = Math.Round(distance, 2)
                Console.Write(" " + distance.ToString + vbTab)
            Else
                Console.Write(vbTab)
            End If
        Next
        Console.WriteLine()
    Next
End Sub
```



```
7. Inspect all rabbits
8. Display Adjacency List
9. Display Adjacency Matrix

Select option: 9

        (1,1)   (2,8)   (9,7)   (13,4)
(1,1)           7.07    10
(2,8)   7.07                    11.7
(9,7)   10                      5
(13,4)          11.7    5
```

9 marks

- *1 mark for getting the x,y coordinates of the starting point*
- *4 marks for IF statement to distinguish between whether node is left or right branch and getting the cords (must be inside IF statement already there)*
- *2 marks for applying Pythagoras correctly (there are several ways to do this, doesn't need to match example; award 1 mark for a good attempt)*
- *1 mark for rounding to 1dp*
- *1 mark for screen capture*

Q	Example Solution	Suggested Marks
15	*Added to class Simulation:* (see code below)	13 marks

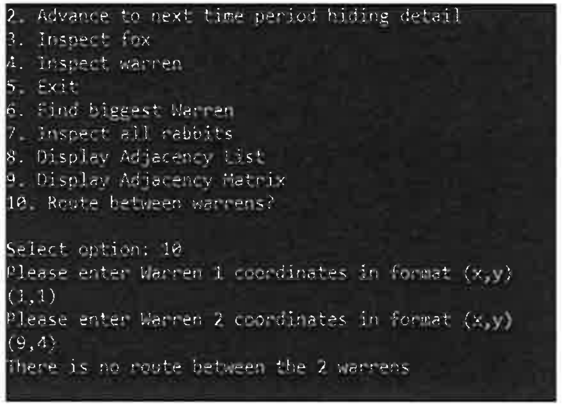
Q 15

Added to class Simulation:

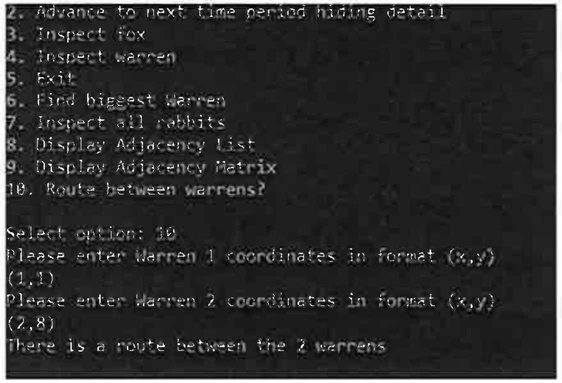
```
        Console.WriteLine("10. Route between warrens?")
        Console.WriteLine()
        Console.Write("Select option: ")
        MenuOption = CInt(Console.ReadLine())
        If MenuOption = 10 Then
          Dim theGraph As New WarrenGraph()
          theGraph.isRoute()
        End If
```

Added to class WarrenGraph:

```
   Public Sub isRoute()
      Dim route As Boolean = False
      Dim coord1, coord2 As String
      Console.WriteLine("Please enter Warren 1 coordinates in format (x,y)")
      coord1 = Console.ReadLine
      Console.WriteLine("Please enter Warren 2 coordinates in format (x,y)")
      coord2 = Console.ReadLine
      For index = 0 To Nodes.Length - 1
        If Nodes(index).getCoord("s") = coord1 Then
          If Nodes(index).getCoord("l") = coord2 Then
            route = True
          ElseIf Nodes(index).getCoord("r") = coord2 Then
            route = True
          End If
        End If
      Next

      If route = True Then
        Console.WriteLine("There is a route between the 2 warrens")
      Else
        Console.WriteLine("There is no route between the 2 warrens")
      End If

    End Sub
```



```
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit
6. Find biggest Warren
7. Inspect all rabbits
8. Display Adjacency List
9. Display Adjacency Matrix
10. Route between warrens?

Select option: 10
Please enter Warren 1 coordinates in format (x,y)
(1,1)
Please enter Warren 2 coordinates in format (x,y)
(9,4)
There is no route between the 2 warrens
```



```
2. Advance to next time period hiding detail
3. Inspect fox
4. Inspect warren
5. Exit
6. Find biggest Warren
7. Inspect all rabbits
8. Display Adjacency List
9. Display Adjacency Matrix
10. Route between warrens?

Select option: 10
Please enter Warren 1 coordinates in format (x,y)
(1,1)
Please enter Warren 2 coordinates in format (x,y)
(2,8)
There is a route between the 2 warrens
```

Suggested Marks

13 marks

- *3 marks for changes to menu*
- *8 marks for isRoute (2 marks for getting each set of coordinates, 1 mark for loop that will check all warrens in graph, 1 mark for IF statement that find the node with correct start coordinates, 1 mark for checking left branch, 1 mark for checking right branch, 2 marks for IF statement with correct output statements)*
- *2 marks for screen captures*

Ideas for modifications	How to implement them

Name	

ZigZag Education supporting

A Level AQA Computer Science Paper 1

Summer 2017: Rabbits and Foxes

Electronic Answer Document (EAD)

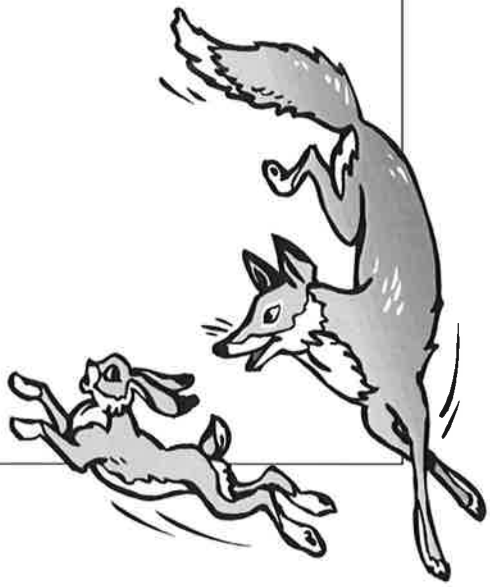
Instructions

- 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:



Programming Theory Questions

Answer all questions.
Remember to save this document regularly.

Q		Answer	*Mark* *(leave blank)*
1			
2	(a)		
	(b)		
	(c)		
	(d)		
	(e)		
	(f)		
	(g)		
	(h)		
	(i)		
3	(a)		
	(b)		
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			

Programming Exercises

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		