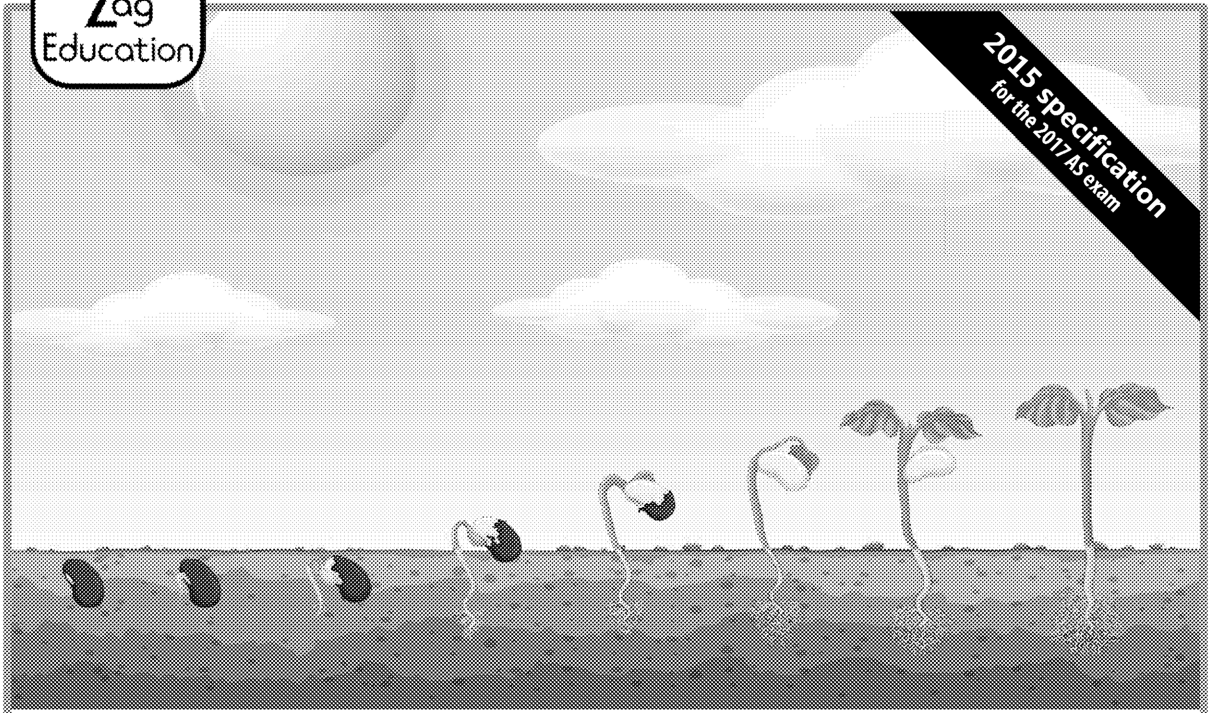
Computer Science AS | AQA | 7516



AQA PAPER 1 EXAM RESOURCE PACK 2017

Plant Growing Simulation

for AS AQA Computer Science

PYTHON 3

zigzageducation.co.uk POD 7356

Contents

Teacher's Introduction

This pack is designed to help you support your students studying AQA AS Computer Science. It is based on the AQA Paper 1 'Plant Growing Simulation' preliminary material (Python 3) – for examination in June 2017.

① **Pre-release Commentary** (for teachers)
A detailed overview of the skeleton program, describing all Python 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 **Structure Chart Activity**
A partially incomplete diagram for students to complete while getting to grips with the skeleton program. Any missing routines and variables 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 'Plant Growing Simulation' code, like Section B in the Paper 1 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 C in the Paper 1 exam. An Electronic Answer Document (EAD) and the modified code can be downloaded using the link below.

Answers and solutions for the structure chart activity, theory questions and programming exercises are provided from page 24 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. Some suggestions are given on page 2.
- Electronic Answer Document (EAD) printout (for reference).

Enter the URL zzed.uk/7356 in your web **browser to download a folder containing the following:**

- **MODIFIED_PY3_CODE.txt** — text file containing the new and/or modified program code as shown in the mark scheme for section ④ (from page 27).
- **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.

Possible Additional Question

1. Allowing the user to specify a season in which the simulation should sta beginning in spring and ending in winter

2. Introducing a second plant type that grows in a different way, e.g.:
    - One that grows best when there is frost
    - One that is drought-resistant
    - One that distributes seeds further away than adjacent squares
    - One that can climb on rocks
    - One that expands into adjacent squares instead of dropping see

3. Allowing the user to specify where the first seed is planted

4. An animal that eats plants, seeds or both on an entire row or column, pe

5. A period of growth in the summer, in which more rainfall means more g adjacent squares without seeds

6. Use of a fertiliser, instead of just soil, that has a change of causing plants seed, as well as in the square of the seed itself

7. Plants that only grow when there are no other plants in adjacent locatio

8. Changing the 'step' mode so that it pauses during every season rather th

9. Allowing the user to input the likelihood of frost and drought at the beg

10. Seeds that will only turn into plants if there was rainfall during the previ

11. Preventing frost or drought from occurring in consecutive years

12. Introducing moss: plants next to a square of moss will never die out in a moss will always be lots

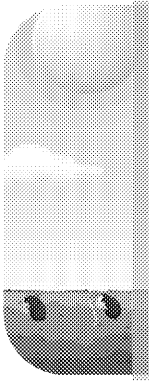
13. Plants that will only produce seeds if they are within two squares of ano blank field, with a single seed, would not be a good starting point for th

14. Plants that grow in height each year, with characters 0, 1, 2, 3, 4, 5, etc. i

15. Including pollinated plants (P) and unpollinated plants (U). Only pollin is only a limited probability, for each plant, of being pollinated.

Plant Growing Simul

Description of the Program



The program is designed to model how a particular plant in a field would propagate over the course of a variable number of years. The field is divided into a grid of squares, and the contents of each square can be any one of the following:

S	Soil
•	A seed
P	A plant
X	A rock

The field is represented in the program using a two-dimensional character array and thus the size of the array, are specified in the program's constants `FIELDL` The value for `FIELDLENGTH` is set to 20 and `FIELDWIDTH` to 35.

The user is first prompted for a number of years that they wish the program to si between -1 and 5. A number between 1 and 5 indicates the number of years tha indicates a desire to step through each year, one at a time, until the user choose

The user is then prompted to choose between an empty fie ld as a starting poin the middle) or a field loaded from a text file. If the user chooses to load the field for the file name. Each file can only hold one field.

Subsequently, there is no user input, except to advance each year in 'step' mode, an behaviour of plants in the fields 'spring', 'summer', 'autumn' and 'winter' for the num

Spring

In spring, all seeds in the field become plants. Subsequently, there If there is a frost, every third plant is killed off and reverts to soil:

•	•	•	•	•
	•	•	•	•
•	•	P	•	•
•	•	•	•	•
•	•	•	•	•

Summer

In summer, there is a 1 in 3 chance of a drought. In the event of a d
simulation (specifically every *other* plant) revert back to soil. If the
happens during summer:

•	•	•	•	•
•	•	•	•	•
•	•	P	•	•
•	•	•	•	•
•	•	•	•	•

Autumn

In autumn, every square adjacent to a plant is given a seed, unless t
rock or a plant, in which case it remains a rock or a plant. If multipl
same place, only one survives:

•	•	•	•	•
•	S	S	S	•
•	S	P	S	•
•	S	S	S	•
•	•	•	•	•

Winter

In winter, all plants die, reverting to soil, but all seeds remain:

•	•	•	•	•
•	S	S	S	•
•	S	•	S	•
•	S	S	S	•
•	•	•	•	•

After each season, the whole field is displayed in the console, along with the na
of the year.

Description of Program Elements

Global Constants

Element	Type	Description
SOIL	A character constant	Stores the character to represent soil: •
SEED	A character constant	Stores the character to represent a seed: **S**
PLANT	A character constant	Stores the character to represent a plant: **P**
ROCKS	A character constant	Stores the character to represent a rock: **X**
FIELDLENGTH	An integer constant	Stores the width of the field, which is also the si is 20.
FIELDWIDTH	An integer constant	Stores the length of the field, which is also the si program is 35.

Local Variables

Element	Type	Description
Column	An integer variable	Declared separately in <u>multiple subroutines</u>, this i
Continuing	A Boolean variable	Indicates whether another year should run in 'ste
Field(,)	A two-dimensional character array	Each element of this array makes up one square o The array is local, declared and initialised in the ReadFile or InitialiseField), but it is pas
FieldRow	A string variable	Used to store each line in turn from read from the
FileName	A string variable	Entered by the user, the name of a file to load. L
Frost	A Boolean variable	Indicates whether or not there will be frost in the
NumberOfPlants	An integer variable	Used to count the number of plants in the field. I
PlantCount	An integer variable	Used to help model frost and drought in Simula
Rainfall	An integer variable	Stores an indication of the amount of rain as an i SimulateSummer.

Element	Type	Description
`Response`	A string variable	Used to store the user's response to the question Simulation.
`Row`	An integer variable	Declared separately in <u>multiple subroutines</u>, this field.
`Year`	An integer variable	The current year, e.g. 1, 2, 3, etc. Local to `Simu`
`Years`	An integer variable	Input by the user, this is the number of years for
`YearsToRun`	An integer variable	Essentially a copy of `Years` (above), returned fr of the local variable within `Simulation`.

Description of Program Routines

The program functions Ⓕ and procedures Ⓟ are described below

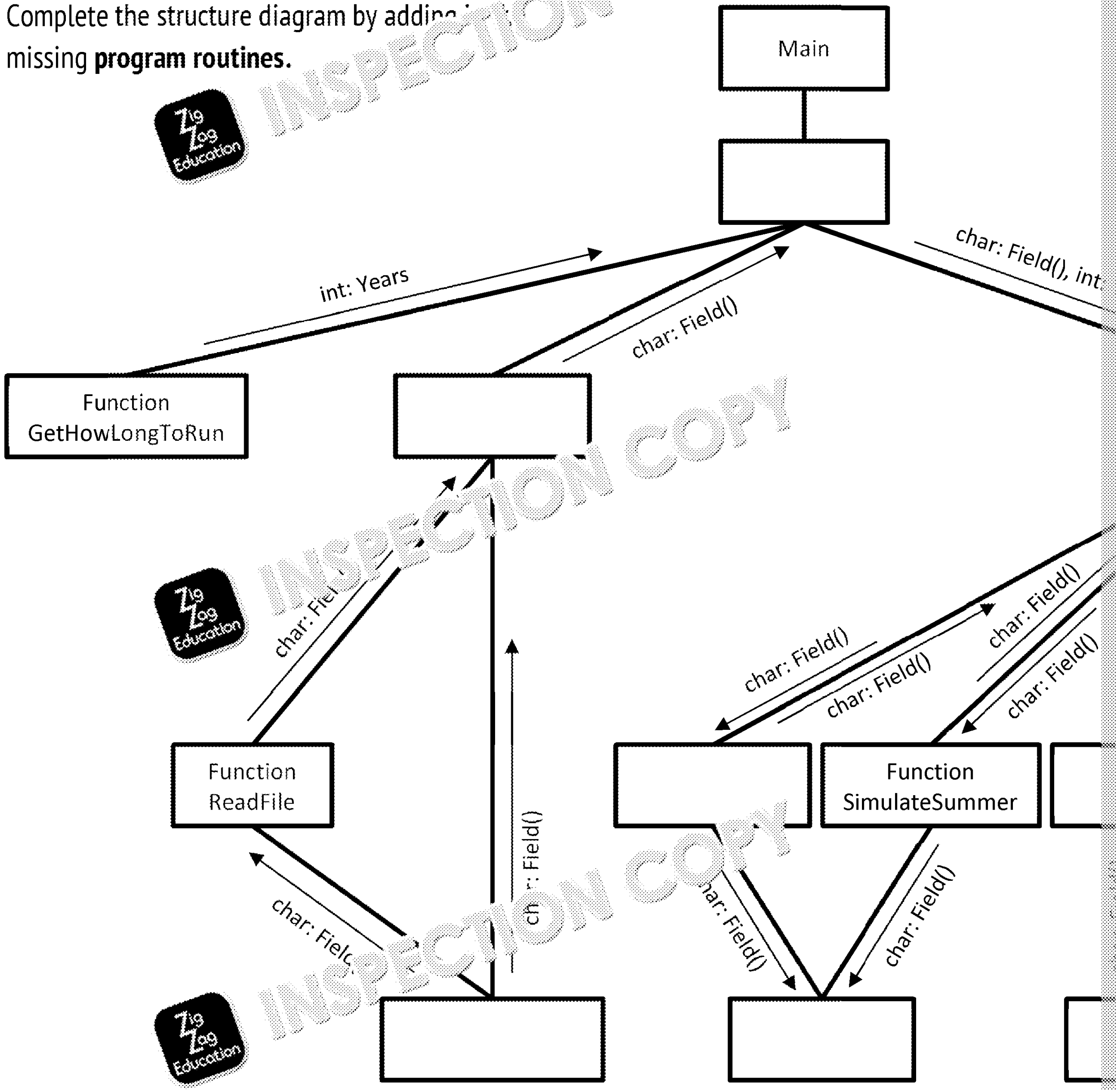
Routine	Description	
`CountPlants` Ⓟ	Receives: `Field` Returns: nothing Called from: `SimulateSpring`, `SimulateSummer`	1. Create variables `Row` and 2. Create variable `NumberOf` 3. Using a nested loop, iterat each plant found 4. Display the number of plan
`CreateNewField` Ⓕ	Receives: nothing Returns: character array Called from: `Readfile`, `InitialiseField`	1. Create variables `Row` and array called `Field` 2. Initialise Field using the `F` 3. Using `Row` and `Column`, it to represent 'soil' 4. Set the array element at th 5. Return the `Field` array to
`Display` Ⓟ	Receives: `Field`, `Season`, `Year` Returns: nothing Called from: `SimulateOneYear`	1. Create variables `Row` and 2. Initialise `Field` using the 3. Display the season and the 4. Using a nested loop, displa

Routine	Description	
GetHowLongToRun Ⓕ	Receives: nothing Returns: integer Called from: Simulation	1. Declare integer variable Y 2. Display user instructions, p 5 to indicate the number o 3. Prompt user for a response
InitialiseField Ⓕ	Receives: nothing Returns: character array Called from: Simulation	1. Initialise Field using the 2. Prompt the user as to whe 3. If 'yes', populate Field w 4. Otherwise, populate Fiel
Main Ⓟ	Receives: nothing Returns: nothing Called from: N/A	1. Initialise random number g consecutive runs extremel 2. Call Simulation
ReadFile Ⓕ	Receives: nothing Returns: character array Called from: InitialiseField	1. Create variables Row and C array called Field 2. Initialise Field using the 3. Prompt the user for a file n 4. For each row read from the 5. Close the file 6. If anything went wrong wi call the CreateNewField the middle 7. Return the Field array, w back to the Initialise
SeedLands Ⓕ	Receives: Field, Row, Column Returns: character array Called from: SimulateAutumn	1. Check that Row and Colu beyond its bounds 2. Check that the element ide 3. If both (1) and (2) are true, 4. Return Field to Simula
SimulateAutumn Ⓕ	Receives: Field Returns: character array Called from: SimulateOneYear	1. Uses local variables Row a 2. For each 'plant' element e elements, i.e. including dia 3. Return Field to Simulate

Routine	Description	
SimulateOneYear Ⓟ	Receives: Field, Year Returns: nothing Called from: Simulation	1. Call SimulateSpring, t 2. Call SimulateSummer, t 3. Call SimulateAutumn, t 4. Call SimulateWinter, t (i.e. simulate each season in turn, c
SimulateSpring Ⓕ	Receives: Field Returns: character array Called from: SimulateOneYear	1. Create Boolean variable F 2. Iterate through the Fielc 3. Randomly determine whet 4. If there is frost, iterate thr and display 'there has bee 5. Call CountPlants 6. Return Field to Simula
SimulateSummer Ⓕ	Receives: Field Returns: character array Called from: SimulateOneYear	1. Create Integer variable Ra 2. By storing a random integ with a 1 in 3 chance of drc 3. If there is drought, iterate 'soil', and display 'there ha 4. Call CountPlants 5. Return Field to Simula
SimulateWinter Ⓕ	Receives: Field Returns: character array Called from: SimulateOneYear	1. Uses local variables Row a 2. Replace any instance of 'p 3. Return Field to Simula
Simulation Ⓟ	Receives: nothing Returns: nothing Called from: Main	1. Declare YearsToRun inte 2. Declare Continuing Boc 'step' mode is not used, th 3. Declare Response string, 4. Declare Field, the two-d InitialiseField, unle 5. If 'step' mode has *not* beer by the user in GetHowLor 6. If 'step' mode *has* been sel indefinitely, until they pre 7. Display 'end of simulation

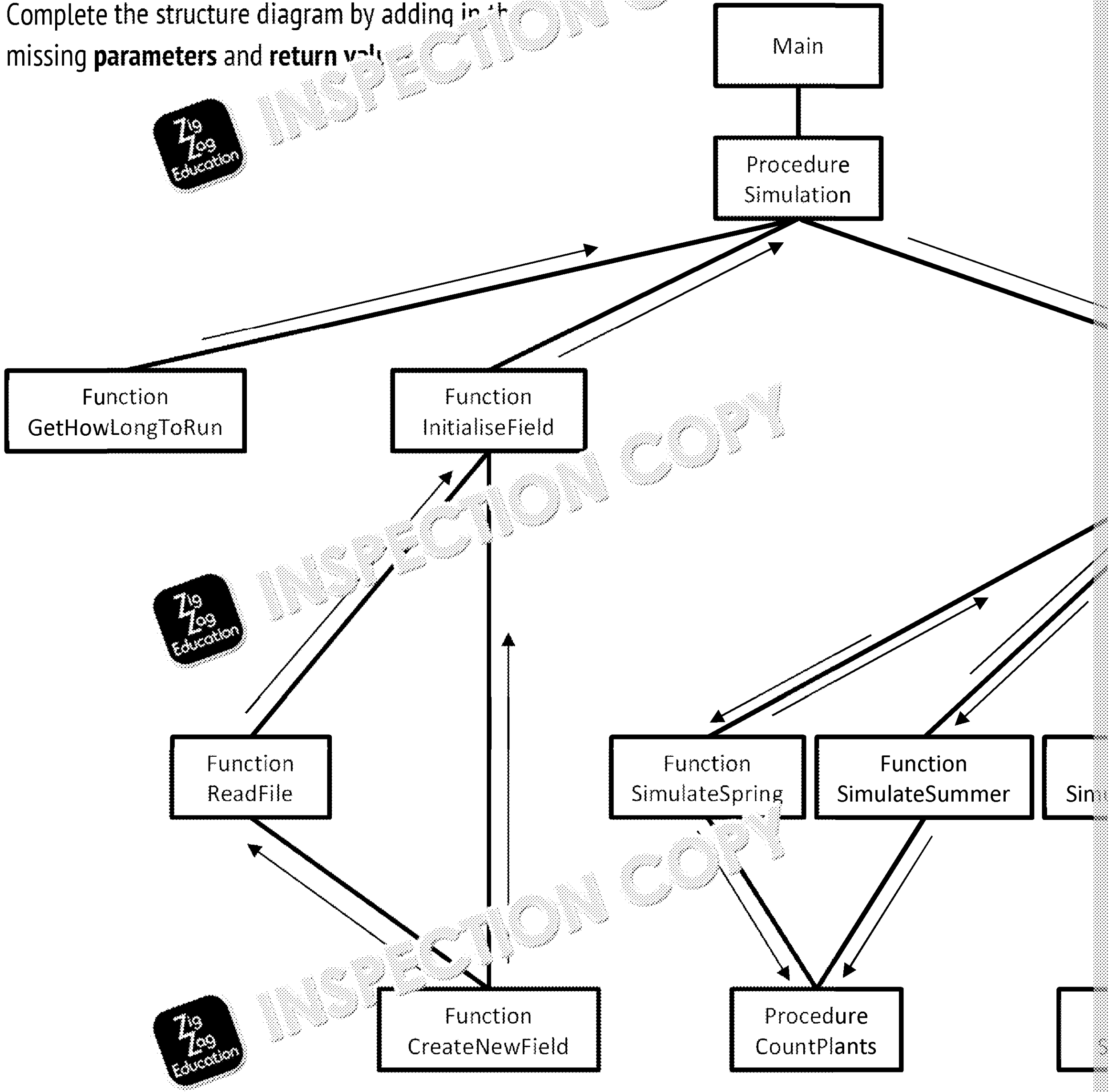
Plant Growing Simulation

Complete the structure diagram by adding in the missing **program routines.**



Plant Growing Sim~~l~~ation

Complete the structure diagram by adding in the missing **parameters** and **return val**



Programming Theory Questio

These questions refer to the Preliminary Material and require you to load the Skeleton Program, but dc

1. State the name of an identifier for:

   (a) A two-dimensional array

   ....

   (b) A function wi meters

   ....

   (c) A constant that can only store a whole number

   ....

   (d) A function that returns an integer

   ....

   (e) A variable that stores a string value

   ....

   (f) A subroutine that calls more than one other subroutine

   ....

   (g) A variable that stores a Boolean

   ....

2. Write three lines of code from the skeleton program that each call different

   ....

   ....

   ....

3. Look at the function `InitialiseField`. Describe the purpose of the

   ....

   ....

   ....

   ....

4. The skeleton program utilises the variable `Field`.

   (a) State the data structure held by `Field`.

   ....

   (b) Explain how data is stored and used in this data structure.

   ....

   ....

   ....

   ....

   ....

   ....

5. State the purpose of the following instruction in the `ReadFile` function?

```
FileHandle = open(FileName, 'r')
```

....

....

6. Describe what would happen, during execution of the `ReadFile` functi
file na[illegible] a [illegible] that does not exist.

....

....

....

....

7. Describe the purpose and operation of the nested loop in the `Display` su

....

....

....

....

....

....

8. Explain the operation of the SeedLands function, including any parame

9. Simulation is a procedure, whereas SeedLands is a function.
Describe the difference between a procedure and a function.

10. Describe the purpose of the following code in the CreateNewField fu

```
Row = FIELDLENGTH // 2
Column = FIELDWIDTH // 2
Field[Row][Column] = SEED
```

11. Look at the function `SimulateSpring`. Describe the purpose and use of t

....

....

....

....

12. The skeleton program begins with a number of constants.
Describe the benefits of the program being written in this way.

....

....

....

....

13. The subroutine `Simulation` uses a While loop, and the function `Simul`

Describe the difference between a While loop and a For loop.

You do not need to address nesting in your answer.

....

....

....

....

14. The procedures `Display` and `CountPlants` both use local variables call
approach would have been to create a single global variable called `Column`.

Describe the advantages of using local variables and the advantages of using a

....

....

....

....

....

....

....

....

15. Describe what is meant by 'string concatenation', and write down an instructi the skeleton program, that uses string concatenation.

16. Explain the purpose of the following instruction in the SimulateSummer f

```
Field[Row][Column] = SOIL
```

17. Describe the purpose of the following instruction in the SimulateSumme

```
RainFall = randint(0, 2)
```

Programming Theory Questio

These questions refer to the Preliminary Material and require you to load the Skeleton Program, but do n

1. State the name of an identifier for:
   (a) A two-dimensional array
   (b) A function with no parameters
   (c) A constant that can only store a whole number
   (d) A function that returns an integer
   (e) A variable that stores a string value
   (f) A subroutine that calls more than one other subroutine
   (g) A variable that stores a Boolean value

2. Write three lines of code from the skeleton program that each call different

3. Look at the function `InitialiseField`. Describe the purpose of the

4. The skeleton program utilises the variable `Field`.
   (a) State the data structure held by `Field`.
   (b) Explain how data is stored and used in this data structure.

5. State the purpose of the following instruction in the `ReadFile` function?

```
FileHandle = open(FileName, 'r')
```

6. Describe what would happen if, during execution of the `ReadFile` funct file name for a file that does not exist.

7. Describe the purpose and operation of the nested loop in the `Display` su

8. Explain the operation of the `SeedLands` function, including any parame

9. `Simulation` is a procedure, whereas `SeedLands` is a function.
   Describe the difference between a procedure and a function.

10. Describe the purpose of the following code in the `CreateNewField` fu

```
Row = FIELDLENGTH // 2
Column = FIELDWIDTH // 2
Field[Row][Column] = SEED
```

11. Look at the function `SimulateSpring`. Describe the purpose and use of

12. The skeleton program begins with a number of constants.
    Describe two benefits of the program being written in this way.

13. The subroutine `Simulation` uses a While loop, and the function `Simula`
    Describe the difference between a While loop and a For loop. *Your answer doe*

14. The procedures `Display` and `CountPlants` both use local variables cal
    An alternative approach would have been to create a single global variable ca
    Describe the advantages of using local variables and the advantages of using a

15. Describe what is meant by 'string concatenation', and write down an instructio the skeleton program that uses string concatenation.

16. Explain the purpose of the following instruction in the `SimulateSummer`

```
Field[Row][Column] = SOIL
```

17. Describe the purpose of the following instruction in the `SimulateSumme`

```
Rainfall = randint(0, 2)
```

Programming Exercises

The following require you to open the skeleton program and make mod

Question 1

This question refers to `SimulateSpring`.

Currently, the source code runs in such a way that a seed *always* turns into a pla `SimulateSpring` so that each seed has only a 40% chance of becoming a p plants remain in the field and may become plants in later years.

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for `SimulateSpring`
- One SCREEN CAPTURE, from the spring of year 1, having loaded the data

Question 2

This question refers to `ReadFile`.

At present, the user must include the suffix .txt in order to load a file. Modify the automatically added, meaning the user is not required to include the '.txt' suffix include the .txt suffix, nothing will be automatically added to the user's input.

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for `ReadFile`
- One SCREEN CAPTURE, showing entry of the data's correct name file wit output for spring of a one-year simulation
- One SCREEN CAPTURE, showing entry of the data's correct name file wit the output for spring of a one-year simulation

Question 3

This question refers to `Simulation`.

The first field the program outputs is always spring of the first year. Modify the pr layout is output, the starting state of the field is output, i.e. the state of the field be

There should be a call to `Display` from `Simulation` which will pass, as pa of a year, and the string 'start' in place of a season. You should make no change

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for `Simulation`
- One SCREEN CAPTURE, showing the full output of the program for which menu, and the file was loaded

Question 4

This question refers to `SimulateSummer`.

Presently, when there is a severe drought, the program simply outputs "There ha Modify the code so that it also outputs the number of plants that have died, in th

"There has been a severe drought: X plants have died".

Evidence you need to provide:

- Your amended sections of the SOURCE CODE PROGRAM for `Simulate`

Question 5

This question refers to `CreateNewField`.

Currently, a new field will consist of only a single seed, in the middle of the field, program so that, for a new field, five rocks are generated and placed at random. rock where the seed, or another rock, already exists, a new random position sho

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for `CreateNewField`
- One SCREEN CAPTURE, showing a user input requesting one year, along first year

Question 6

This question refers to `GetHowLongToRun`.

The user is currently prompted for a number between -1 and 5, although the pro other integers, and crashes when the user attempts to enter any nun-numerical that will cause the program to loop until an integer between -1 and 5 has been e An error message of "Please enter a whole number between -1 and 5" should be

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for `GetHowLongToRun`
- One SCREEN CAPTURE, showing an attempt to add the character 'x' in re "Enter a number between 0 and 5, or -1 for stepping mode:".
- One SCREEN CAPTURE, showing an attempt to add the integer 6 in respo "Enter a number between 0 and 5, or -1 for stepping mode:".
- One SCREEN CAPTURE, showing an attempt to add the integer '1' in resp "Enter a number between 0 and 5, or -1 for stepping mode:".

Question 7

This question refers to `SimulateSpring` and to the constants section.

A new type of plant is to be introduced to the simulation, which will be a weed, n `WEED`, which stores a capital 'W'. Every spring, each point containing soil has a 1 weed. Weeds, unlike plants, do not die in winter.

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for the constants section
- Your amended SOURCE CODE PROGRAM for `SimulateSpring`
- One SCREEN CAPTURE, showing the spring output in year 1 of a two-yea not the data file)
- One SCREEN CAPTURE, showing the spring output in year 2 of a two-yea not the data file)

Question 8

This question refers to `CreateNewField`.

When a new field is created, a single seed is placed at the centre of that field. Mo prompted for a number of seeds that will be randomly dispersed throughout the seed in the middle of the field should be removed.

If a seed is randomly placed in a position where a seed already exists, the second spring of the first year the program should output the number of seeds that hav following fo

"x seed(s) lost."

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for `CreateNewField`
- One SCREEN CAPTURE, showing a user input requesting 20 seeds, along first year (new field, one-year simulation)

Question 9

This question refers to `InitialiseField`.

As the program currently runs, it requires the user to enter a capital 'Y' in order t entered, a new field is created. The following changes should be made to how t

- Entering either an upper case 'Y' or a lower case 'y' should result in the use by `ReadFile` being called
- Entering an upper case 'N' or a lower case 'n' should result in a new field, by C
- Entering any other value should result in an appropriate error message, wit the original question again

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for `InitialiseField`
- One SCREEN CAPTURE, showing the result of entering a lower case 'y' in "Do you want to load a file with seed positions?"
- One SCREEN CAPTURE, showing the result of entering a lower case 'n' in "Do you want to load a file with seed positions?"
- One SCREEN CAPTURE, showing the result of entering a lower case 'x' in "Do you want to load a file with seed positions?"

Question 10

This question refers to `SimulateAutumn`.

In autumn, seeds will blow in from the north, south, east or west. During `Simul` integer should be generated that will be 0, 1, 2 or 3. That number will cause see way, indicated as follows:

<table>
<tr><td><b>Random number</b></td><td>0</td><td>1</td><td>2</td></tr>
<tr><td><b>Wind direction</b></td><td>From North</td><td>From East</td><td>From S</td></tr>
<tr><td><b>Seed dispersal</b></td>
<td><table><tr><td></td><td></td><td></td></tr><tr><td></td><td>P</td><td></td></tr><tr><td>S</td><td>S</td><td>S</td></tr></table></td>
<td><table><tr><td>S</td><td></td><td></td></tr><tr><td>S</td><td>P</td><td></td></tr><tr><td>S</td><td></td><td></td></tr></table></td>
<td><table><tr><td>S</td><td>S</td></tr><tr><td></td><td>P</td></tr><tr><td></td><td></td></tr></table></td></tr>
</table>

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for `SimulateAutumn`
- One SCREEN CAPTURE, from the autumn of year 1, run from a new field, autumn

Question 11

This question refers to `Simulation`, as well as a new procedure, `SaveFiel`

When a simulation has ended, after the output "End of Simulation", the user sho they wish to save the field. If they do, the `Simulation` procedure should cal `SaveField`, which will require a two-dimensional cha[illegible]r array, `Field`, to `SaveField` will perform three tasks:

- Prompt the user for a file n[illegible]which the program will add a '.txt' suffi
- Attempt to save th[illegible]is file, with each field row on a separate row
- If the[illegible]s s[illegible]ul, display the message "File saved", otherwise, displ

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for `Simulation`
- The SOURCE CODE for a new procedure, in full, called `SaveField`
- One SCREEN CAPTURE that shows the following, after the simulation has
  - The user indicating that they would like to save the field
  - The user entering 'test' as the file name
  - The program's subsequent output
- One SCREEN CAPTURE that shows the output for spring, having loaded ' for one year

Question 12

This question refers to `SimulateS`[illegible] new function, `GrowPlants`, a

Currently, if a drought [illegible] of the plants within the simulation are killed, summer in [illegible]en [illegible] there is no drought. Modify the program to enable the

- If ther[illegible] drought, the program should function as it currently functions
- If there is no drought, the program should call a new function, `GrowPlan`
- `GrowPlants` requires the following parameters:
  - A two-dimentional character array, `Field`
  - An integer, `Row`, indicating the row the of the current plant
  - An integer, `Column`, indicating the column of the current plant
- `GrowPlants` should return a two-dimensional character array
- For each directly adjacent point to the plant on the field (i.e. each of north, should be a 25% chance of an offshoot, which will be represented as a capi use a constant, `OFFSHOOT`
- Offshoots can only sprout in locations curr[illegible]taining only soil
- Offshoots cannot sprout outsid[illegible] for a plant on the edge of the fie
- At the start of `Simu`[illegible]`er`, any offshoots from last year become drought [illegible]

Evidence yo[illegible]d to provide:

- Your amended SOURCE CODE PROGRAM for `SimulateSummer`
- Your amended SOURCE CODE PROGRAM for the constants section
- The SOURCE CODE for a new procedure, in full, called `GrowPlants`

Question 13

This question refers to multiple sections of the skeleton code.

Currently, any seeds in the ground at the beginning of spring become plants du modify the program to model a seed that takes longer to germinate. The consta and will be replaced by two new constants, NEWSEED (using a lower case 's') a upper case 'S').

During spring, new seeds will become mature seeds, and will remain mature see seeds will become plants, which will then grow and develop as normal. The sim seed in the [illegible] of the field.

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for the constants section
- Your amended SOURCE CODE PROGRAM for `CreateNewField`
- You amended SOURCE CODE PROGRAM for `SimulateSpring`
- Your amended SOURCE CODE PROGRAM for `SeedLands`
- One SCREEN CAPTURE, showing the spring output in year 1 of a two-yea
- One SCREEN CAPTURE, showing the spring output in year 2 of a two-yea

Question 14

This question refers to `CreateNewField` and the constants section.

Currently, the size of the field is fixed to a width of 35 and a length of 25. Modify specify the length and width of the field when creating a new field. Validation s minimum of 10 and a maximum of 50 for both the width and the length. Any at enter a value beyond this range should cause a suitable error message to be dis

The program should default to a width of 35 and a length of 20 when a file is loa

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for the constants section
- Your amended SOURCE CODE PROGRAM for `CreateNewField`
- One SCREEN CAPTURE showing the program's response to an attempt t dimensions a value of '9', having requested a one-year simulation
- One SCREEN CAPTURE showing the output requesting a length of 10 anc of a one-year simulation

Question 15

This question refers to `SimulateAutumn`, and to a new function `Simulat`

The simulation is to be changed to include the possibility that a disease can strik it has had a chance to spread its seeds. There is a 1 in 25 chance of a plant contr the plant itself and any other plants or seeds within two squares on the field. It c rocks are in no way affected themselves

The effects of the disease are shown below, with the disease striking the middle

Before disease	A

P	S	•	•	P
S	S	•	X	•
•	•	P	•	•
•	P	X	•	P
P	•	P	•	•

•	•
•	•
•	•
•	•
•	•

A plant that has been eliminated by disease will not spread any seeds.

Create a new function called `SimulateDisease`, to have the following para

- A two-dimensional character array, `Field`
- An integer, `Row`, indicating the row of the diseased plant
- An integer, `Column`, indicating the column of the diseased plant

`SimulateDisease` should return a two-dimensional character array to `Si`

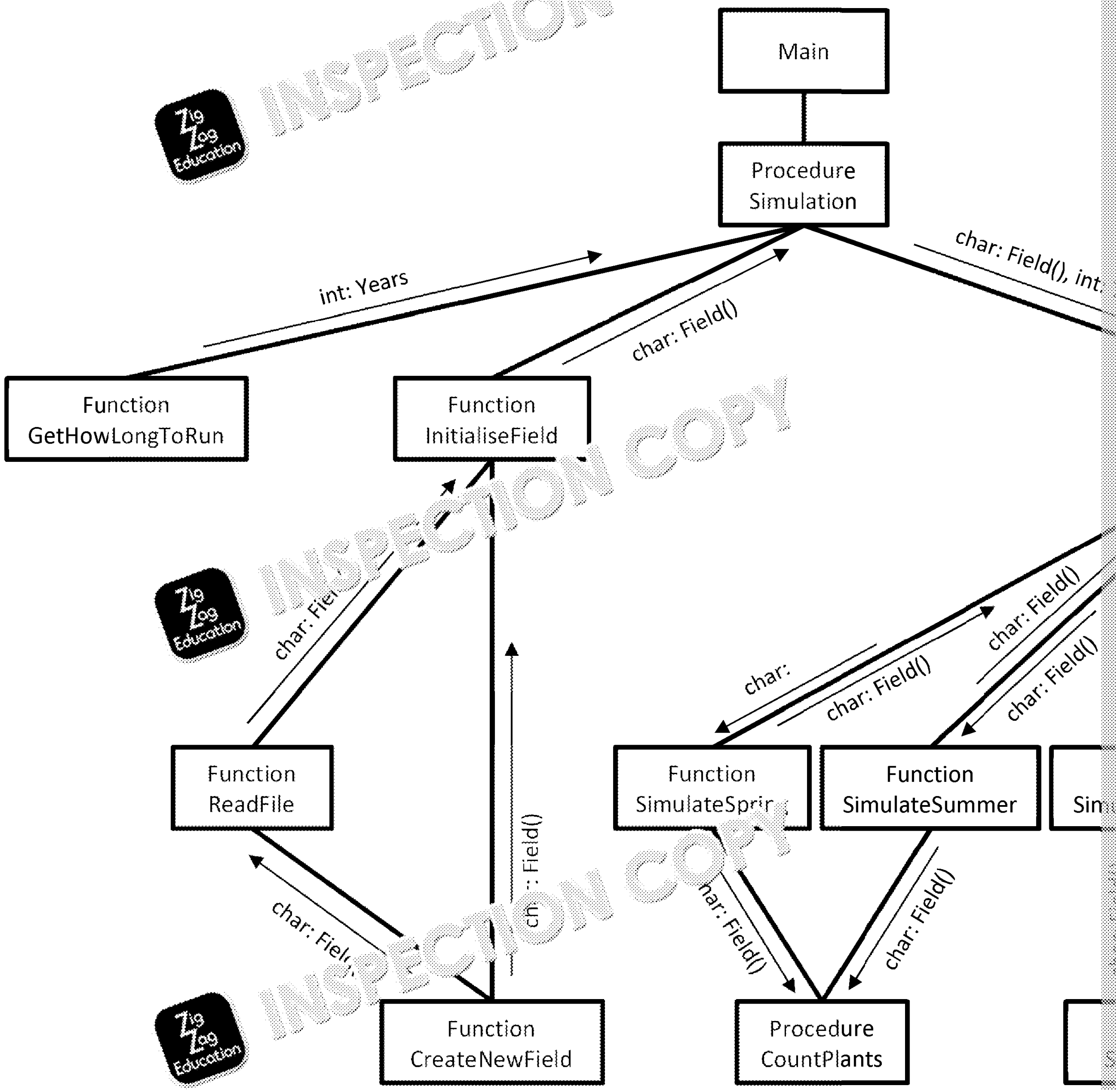
The program should not attempt to spread the disease outside the bounds of th in the leftmost column, the disease will not spread to the left.

If disease does strike, the program should output the text "Disease has struck!" o disease.

Evidence you need to provide:

- Your amended SOURCE CODE PROGRAM for `SimulateAutumn`
- The SOURCE CODE for a new function in full, called `SimulateDisea`

Plant Growing Simulation



Programming Theory Questions (Answ

Q	Answer / Guidance
1a	`Field`
1b	`GetHowLongToRun` / `InitialiseField` / `ReadFile` / `CreateNewFi`
1c	`FIELDLENGTH` / `FIELDWIDTH`
1d	`GetHowLongToRun`
1e	`FileName` / `FieldRow` / `Response`
1f	`Simulation` / `InitialiseField` / `SimulateOneYear`
1g	`Frost` / `Continuing`
2	Any three of the following (1 mark each): • Any instruction beginning `print(`… • Any instruction containing `input(`… • `FileHandle = open(FileName, 'r')` • `FileHandle.close()`
3	To store user input / to accept user response / similar phrasing (1 mark) If 'Y', call `ReadFile` / if not 'Y' call `CreateNewField` / its contents det to then call (1 mark) *Both marks can be picked up in a single sentence, e.g. "to allow user input to subroutine is called".*
4a	Two-dimensional character array / two-dimensional char array (1 mark) *Must include both number of dimensions and data type for the mark.*
4b	Any three of the following (1 mark each): • Dimensioned using `FIELDLENGTH` and `FIELDWIDTH` constants • 20 x 35 array / 35 x 20 array (*ignore confusion of rows and column are present*) • Two values needed to access an element/row, column needed to • Each element can store one character
5	Opens a file whose name is stored in `FileName` / creates a reader to acc (1 mark)
6	The try block would be aborted / the catch block would be executed (1 m Field would be populated by the `CreateNewField` routine / `CreateNe` called (1 mark)
7	Any three of the following (1 mark each): • Iterate/loop/cycle through each element/character in the array/fi • Write/output/display the contents of each element • One 'For' handles each row, the other handles each element/cha • Between rows, insert a line break / new line / label each row wit
8	Requires `Field`, `Row` and `Column` as parameters / accepts an array and t arguments (1 mark) Returns `Field` / returns a character array (1 mark) Checks that the location is within the field / checks that row is between 0 between 0 and 35 / checks that row is between 0 and `FIELDLENGTH` and and `FIELDWIDTH` (1 mark) Checks that the location contains soil / contains '.' (1 mark) Write/puts/plants/etc. a seed/. in the location (1 mark)

Q	Answer / Guidance
9	A procedure performs a set of actions/tasks/instructions and returns no va A function returns a value (1 mark)
10	Perform integer division on `Row` and `Column` (1 mark) Identifying the middle of the field/array (1 mark) Changing the contents of the corresponding element (1 mark) To contain `SEED/S` (1 mark)
11	It is set to a random number/integer of either 0 or 1 (1 mark) If it is 1, loop through the array/field (1 mark)
12	Any of the following (1 mark each): • Constants won't be accidentally changed • By being at the start of the code, the code is easier to read/unders *indication that it is easier for the computer to execute*) • No need to remember values / constant names are more memora code is more readable with constant names than values
13	A for loop repeats a set/predetermined number of times (1 mark) A while loop loops until a condition is met (1 mark)
14	Any four of the following (1 mark each): • Local variables are removed from memory when the routine ends • Local variables in different routines can share a name/identifier • Local variables are protected from / invisible to other programme • Global variables only need to be declared once • Global variables reduce complications regarding parameters / ret • Global variables are declared all in one place, aiding readability / declared at the top, aiding readability

15	Concatenation involves joining multiple strings together (into one) (1 ma
	Examples
	`print('Season: ', Season, '  Year number: ', Year)`
	`print('There are', NumberOfPlants, 'plants growing')`

16	Set/change the array element of `field` / the array (1 mark) Element identified by `Row` and `Column` (1 mark) (Element set to SOIL), placing a dot/. in that element (1 mark)
17	Generate a random integer between 0 and 2 / generate a random number (1 mark) Store that number/integer in `Rainfall` (1 mark)

Question 1

1 mark New code added to `SimulateSpring` within the existing IF statement in

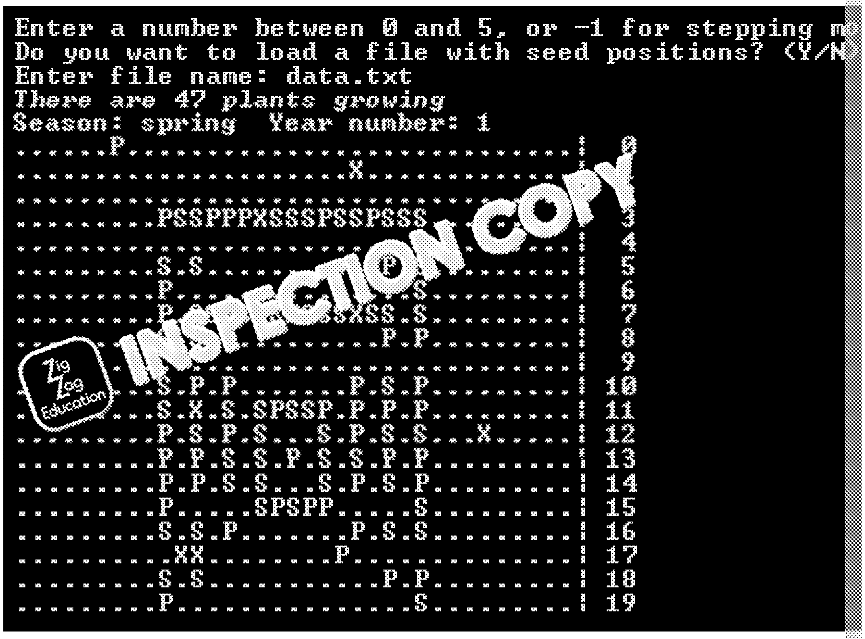
1 mark Random number generated that would allow for a 50% probability

1 mark IF statement that would result in a 50% probability

```
for Row in range(FIELDLENGTH):
  for Column in range(FIELDWIDTH):
    if Field[Row][Column] == SEED:
      if randint(0, 4) < 2:
        Field[Row][Column] = PLANT
CountPlants(Field)
```

1 mark Output containing a combination of plants and seeds.

Due to random variation, the output is highly unlikely to be identical to th or no seeds sprouting as plants are statistically insignificant



```
Enter a number between 0 and 5, or -1 for stepping m
Do you want to load a file with seed positions? (Y/N
Enter file name: data.txt
There are 47 plants growing
Season: spring  Year number: 1
......P............................|  0
.....................X.............|  1
...................................|  2
.........PSSPPPXSSSPSSPSSS.........|  3
...................................|  4
.........S.S...........P...........|  5
.........P...............S.........|  6
.........P.........SSXSS.S.........|  7
.......................P.P.........|  8
...................................|  9
.........S.P.P.......P.S.P.........| 10
.........S.X.S.SPSSP.P.P.P.........| 11
.........P.S.P.S...S.P.S.S..X......| 12
.........P.P.S.S.P.S.S.P.P.........| 13
.........P.P.S.S...S.P.S.P.........| 14
.........P.....SPSPP.....S.........| 15
.........S.S.P.......P.S.S.........| 16
..........XX........P..............| 17
.........S.S...........P.P.........| 18
.........P...............S.........| 19
```

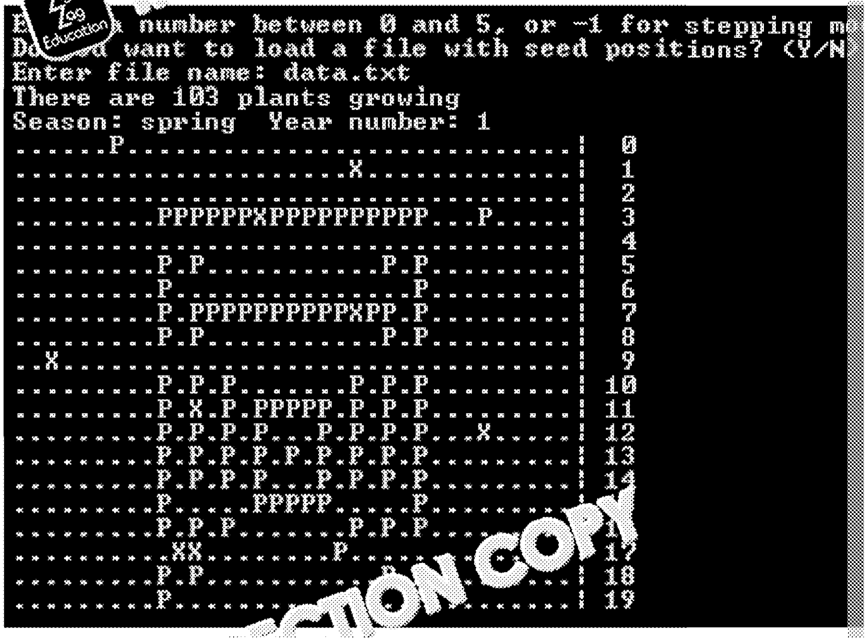
Question 2

1 mark IF statement added to `ReadFile` to determine a missing .txt suffix

1 mark Suffix only added if it is missing

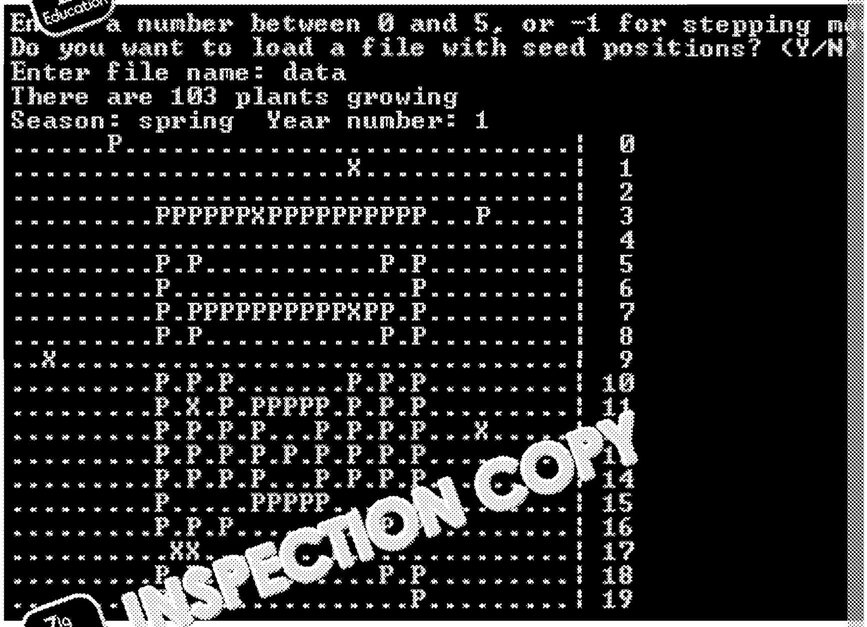
```
FileName = input('Enter file name: ')
  if ".txt" not in FileName:
    FileName = FileName + ".txt"
```

1 mark Screen output showing the file requested with its suffix included – must s
field (file name used in this case is 'data'):



```
Enter a number between 0 and 5, or -1 for stepping m
Do you want to load a file with seed positions? (Y/N
Enter file name: data.txt
There are 103 plants growing
Season: spring  Year number: 1
......P............................|  0
.....................X.............|  1
...................................|  2
.........PPPPPPXPPPPPPPPPP...P.....|  3
...................................|  4
.........P.P...........P.P.........|  5
.........P...............P.........|  6
.........P.PPPPPPPPPPXPP.P.........|  7
.........P.P...........P.P.........|  8
..X................................|  9
.........P.P.P.......P.P.P.........| 10
.........P.X.P.PPPPP.P.P.P.........| 11
.........P.P.P.P...P.P.P.P...X.....| 12
.........P.P.P.P.P.P.P.P.P.........| 13
.........P.P.P.P...P.P.P.P.........| 14
.........P.....PPPPP.....P.........| 15
.........P.P.P.......P.P.P.........| 16
..........XX.........P.............| 17
.........P.P...........P.P.........| 18
.........P...............P.........| 19
```

1 mark Screen output showing the file requested without its suffix – the rest of th



```
Enter a number between 0 and 5, or -1 for stepping m
Do you want to load a file with seed positions? (Y/N
Enter file name: data
There are 103 plants growing
Season: spring  Year number: 1
......P............................|  0
.....................X.............|  1
...................................|  2
.........PPPPPPXPPPPPPPPPP...P.....|  3
...................................|  4
.........P.P...........P.P.........|  5
.........P...............P.........|  6
.........P.PPPPPPPPPPXPP.P.........|  7
.........P.P...........P.P.........|  8
..X................................|  9
.........P.P.P.......P.P.P.........| 10
.........P.X.P.PPPPP.P.P.P.........| 11
.........P.P.P.P...P.P.P.P...X.....| 12
.........P.P.P.P.P.P.P.P.P.........| 13
.........P.P.P.P...P.P.P.P.........| 14
.........P.....PPPPP.....P.........| 15
.........P.P.P.......P.P.P.........| 16
..........XX.........P.............| 17
.........P.P...........P.P.........| 18
.........P...............P.........| 19
```

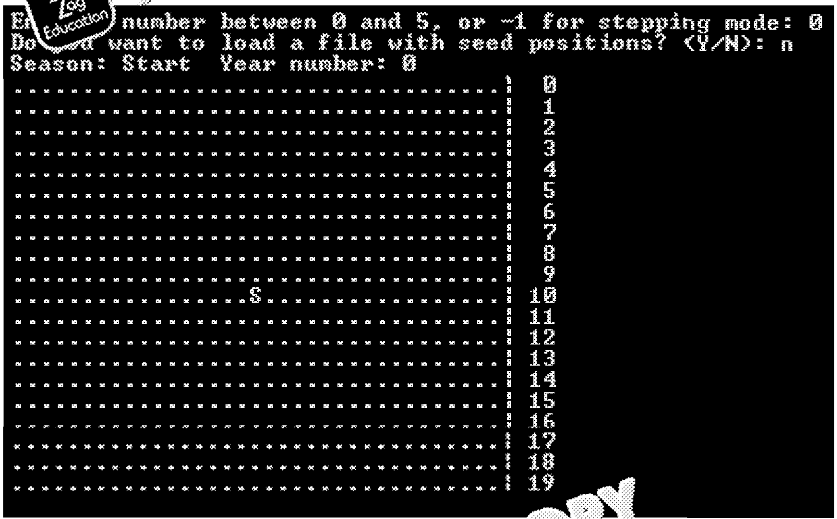
Question 3

1 mark Instruction `Field = InitialiseField()` now appears before the IF s not after it (mark cannot be awarded if a path through this part of the prog `InitialiseField` being called twice)

1 mark Call to `Display` with parameters identical to those below:

```
Display(Field, "Start", 0)
if YearsToRun != 0:
Display(Field, "Start", 0)
```

1 mark N[illegible] for zero years – program should output the field with



```
E[illegible] number between 0 and 5, or -1 for stepping mode: 0
Do [illegible] want to load a file with seed positions? (Y/N): n
Season: Start  Year number: 0
...................................|  0
...................................|  1
...................................|  2
...................................|  3
...................................|  4
...................................|  5
...................................|  6
...................................|  7
...................................|  8
...................................|  9
.................S.................| 10
...................................| 11
...................................| 12
...................................| 13
...................................| 14
...................................| 15
...................................| 16
...................................| 17
...................................| 18
...................................| 19
```

Question 4

1 mark An appropriately named variable to keep track of dead plants in `SimulateS`

```
if RainFall == 0:
  PlantCount = 0
  DeadPlants = 0
```

1 mark Variable initialised to zero before the nested loop:

```
DeadPlants = 0
for Row in range(FIELDLENGTH):
```

1 mark Variable incremented at the same point as each 'plant' becoming 'soil':

```
if PlantCount % 2 == 0:
  Field[Row][Column] = SOIL
  DeadPlants += 1
```

1 mark Console output updated correctly:

```
print('There has been a severe drought:', DeadPlants,
```

Question 5

1 mark Loop that will always run at least five times in `CreateNewField`

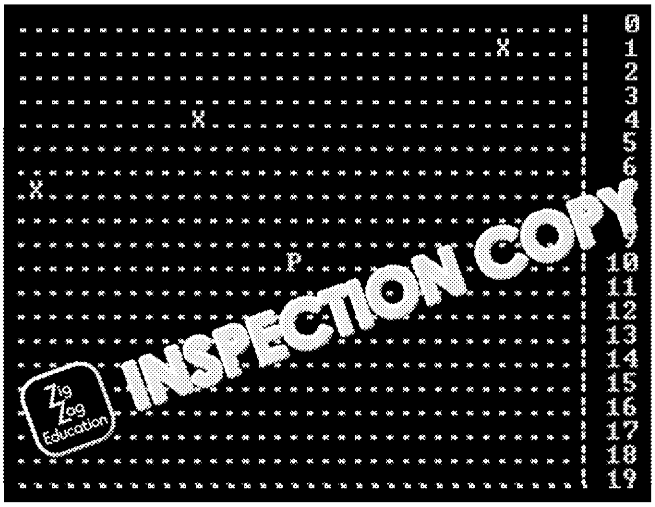
2 marks Random generation of row using `FIELDLENGTH` constant and random gen `FIELDWIDTH` constant (if literals 20 and 35 are used only 1 mark can be a

1 mark IF or condition-controlled loop to ensure one rock is not positioned on top

1 mark If exactly five rocks will always be placed and the seed will never be overw

```
rockCount = 0
while rockCount < 5:
  rowForRock = randint(0,FIELDLENGTH - 1)
  columnForRock = randint(0, FIELDWIDTH - 1)
  if Field[rowForRock][columnForRock] != ROCKS:
    Field[rowForRock][columnForRock] = ROCKS
    rockCount += 1
```

1 mark Spring of the first year with the plant and five rocks:



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

Question 6

1 mark A loop that will continue until valid input has been received within GetHo

1 mark Rejecting non-numeric input

1 mark Error if input is not numeric

1 mark Checking for range between −1 and 5

1 mark Error if input is not within that range

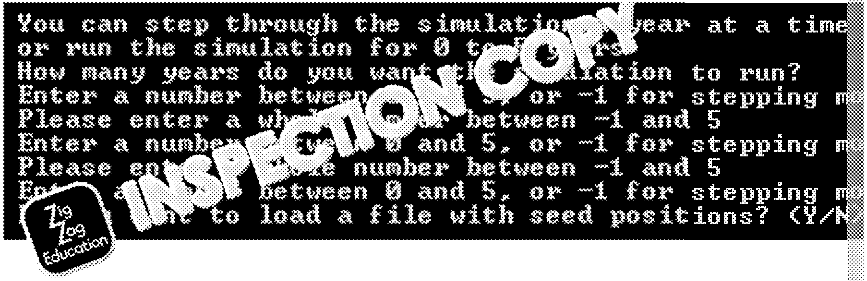
1 mark Return statement only return a valid input

```
while True:
  Years = input('Enter a number between 0 and 5, or 
  try:
    if int(Years) in range(-1, 6):
      return int(Years)
    else:
      print('Please enter a whole number between -1 a
  except:
    print('Please enter a whole number between -1 and
```

1 mark Input of 'x' with error message

1 mark Input of '6' with error message

1 mark Input of '1' followed by prompt to load file



```
You can step through the simulation a year at a time
or run the simulation for 0 to 5 years
How many years do you want the simulation to run?
Enter a number between 0 and 5, or -1 for stepping m
Please enter a whole number between -1 and 5
Enter a number between 0 and 5, or -1 for stepping m
Please enter a whole number between -1 and 5
Enter a number between 0 and 5, or -1 for stepping m
Do you want to load a file with seed positions? (Y/N
```

Question 7

1 mark WEED constant added outside of any subroutines:

```
WEED = 'W'
```

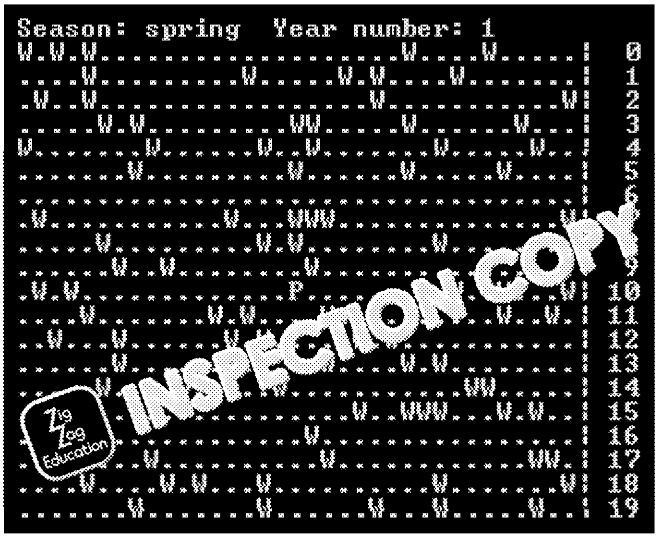
1 mark Random number generated to represent a 10% probability within Simula

1 mark 10% probability of WEED constant being used (no mark if literal 'W')

1 mark No plant or rock can ever become a seed (e.g. with ElseIf structure):

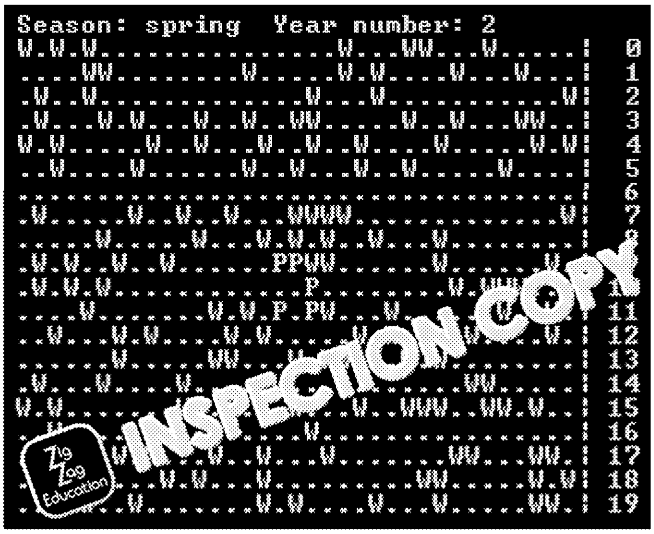
```
for Row in range(FIELDLENGTH):
  for Column in range(FIELDWIDTH):
    if Field[Row][Column] == SEED:
      Field[Row][Column] = PLANT
    elif Field[Row][Column] == SOIL:
      if randint(0, 9) == 0:
        Field[Row][Column] = WEED
```

1 mark Spring of year 1, with weeds and the central plant visible:



```
Season: spring  Year number: 1
W.W.W...............W....W.....|  0
....W.........W.....W.W....W.......|  1
.W..W.................W...........W|  2
.....W.W.........WW.....W......W...|  3
W.......W......W..W.......W.....W..|  4
.......W.........W......W.....W....|  5
...................................|  6
.W...........W...WWW..............W|  7
.....W.........W.W........W........|  8
......W..W........W................|  9
.W.W.............P................W| 10
....W.......W.W...............W..W.| 11
..W...W......W.....................| 12
......W.................W.W........| 13
.....W..........W...........WW.....| 14
.....................W..WWW...W.W..| 15
..................W................| 16
........W..........W............WW.| 17
....W....W.W...W..........W.......W| 18
.......W.......W......W..W......W..| 19
```

1 mark Spring of year 2, with more weeds visible, including all original weeds:



```
Season: spring  Year number: 2
W.W.W...............W...WW...W.....|  0
....WW........W.....W.W....W...W...|  1
.W..W.............W...W...........W|  2
.W...W.W...W..W..WW.....W..W...WW..|  3
W.W.....W..W...W..W..W....W.....W.W|  4
..W....W......W..W...W..W.....W....|  5
...................................|  6
.W.....W..W..W...WWWW.............W|  7
.....W.....W...W.W.W..W...W........|  8
.W.W..W..W......PPWW......W......W.|  9
.W.W.W............P........W.WWW...| 10
....W.......W.W.P.PW...W......W..W.| 11
..W...W.W....W.W.....W......W....W.| 12
......W.....WW...W......W.W........| 13
.W...W....W.....W...........WW.....| 14
W.W..................W..WWW..WW.W..| 15
..W...............W................| 16
......W.W...W..W...W.......WW...WW.| 17
....W....W.W...W.........WW.....W.W| 18
.......W.......W.W....W...W.....WW.| 19
```

Question 8

1 mark Variable to store the number of seeds in `CreateNewField`

1 mark Variable set to user input

1 mark Variable to store lost seeds

1 mark Lost seeds variable initialised to zero

```
seedCount = int(input('Enter number of starting seed
lostSeeds = 0
```

1 mark Loop repetition controlled by user input

2 marks x and y coordinates set randomly using `FIELDLENGTH` and `FIELDWIDTH` (
20 and 35 are used)

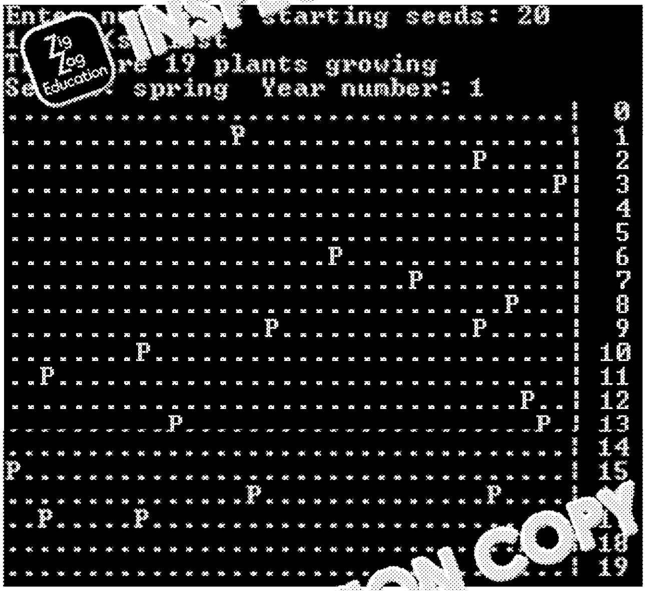
1 mark IF statement to check existence of seed in target element

1 mark Variable correctly incremented if seed already present

1 mark Correct output, outside loop:

```
for x in range(1, seedCount):
  seedRow = randint(0, FIELDLENGTH - 1)
  seedColumn = randint(0, FIELDWIDTH - 1)
  if Field[seedRow][seedColumn] == SEED:
    lostSeeds += 1
  Field[seedRow][seedColumn] = SEED
print(lostSeeds, 'seed(s) lost')
```

1 mark User requests 20 seeds; number of seeds lost is displayed, which, when ad
should equal 20:



```
Enter number of starting seeds: 20
1 seed(s) lost
There are 19 plants growing
Season: spring  Year number: 1
...................................|  0
..............P....................|  1
.............................P.....|  2
..................................P|  3
...................................|  4
...................................|  5
....................P..............|  6
.........................P.........|  7
...............................P...|  8
................P............P.....|  9
........P..........................| 10
..P................................| 11
................................P..| 12
..........P......................P.| 13
...................................| 14
P..................................| 15
...............P..............P....| 16
..P.....P..........................| 17
...................................| 18
...................................| 19
```

Question 9

1 mark ReadFile called when 'Y' or 'y' is entered in InitialiseField

1 mark CreateNewField called when 'N' or 'n' is entered

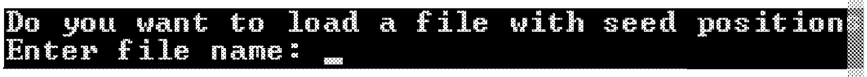
1 mark Error displayed when anything else is entered

1 mark Loop continues until a valid response is entered

1 mark Only a valid response causes a return value and all valid responses cause a

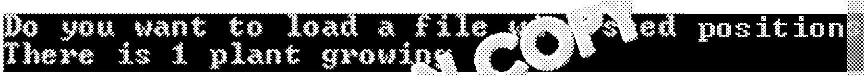
```
while True:
  Response = input('Do you want to load a file with s
  if Response == 'Y' or Response == 'y':
    Field = ReadFile()
    return Field
  elif Response == 'N' or Response == 'n':
    Field = CreateNewField()
    return Field
  else:
    print('Please enter \'Y\' or \'N\'')
```

1 mark Entering lower case 'y' results in 'Enter file name:'



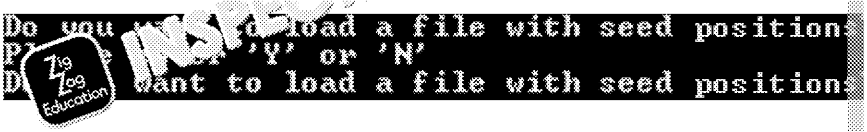
```
Do you want to load a file with seed position
Enter file name: _
```

1 mark Entering lower case 'n' begins the simulation



```
Do you want to load a file with seed position
There is 1 plant growing
```

1 mark Entering 'x' results in error message being displayed and the question bein



```
Do you want to load a file with seed position
Please enter 'Y' or 'N'
Do you want to load a file with seed position
```

Question 10

1 mark Declaration of variable, in `SimulateAutumn`, to store wind – must be be

1 mark Wind variable initialised to random value selected from a range of four **eq** before For loop

```
wind = randint(0, 3)
for Row in range(FIELDLENGTH):
```

1 mark Four-clause IF structure or equivalent, which must be inside the existing IF

1 mark Correct three seeds for wind = 0

1 mark Correct three seeds for wind = 1

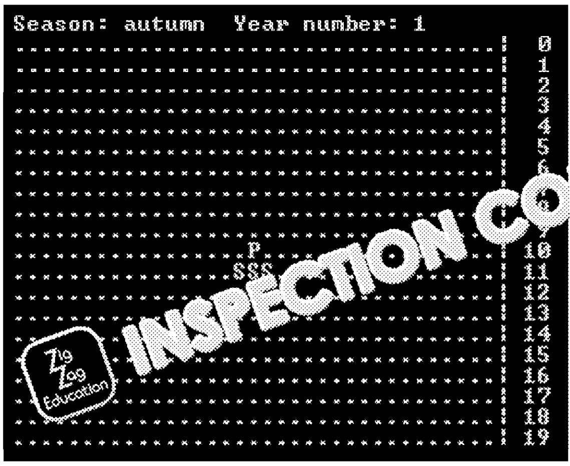
1 mark Correct three seeds for wind = 2

1 mark Correct three seeds for wind = 3

1 mark Function always returns the correct value, i.e. all four values of wind produ and the Return statement is still correctly positioned

```
  for Column in range(FIELDWIDTH):
    if Field[Row][Column] == PLANT:
      if wind == 0:
        Field = SeedLands(Field, Row + 1, Column - 1)
        Field = SeedLands(Field, Row + 1, Column)
        Field = SeedLands(Field, Row + 1, Column + 1)
      elif wind == 1:
        Field = SeedLands(Field, Row - 1, Column - 1)
        Field = SeedLands(Field, Row, Column - 1)
        Field = SeedLands(Field, Row + 1, Column - 1)
      elif wind == 2:
        Field = SeedLands(Field, Row - 1, Column - 1)
        Field = SeedLands(Field, Row - 1, Column)
        Field = SeedLands(Field, Row - 1, Column + 1)
      else:
        Field = SeedLands(Field, Row - 1, Column + 1)
        Field = SeedLands(Field, Row, Column + 1)
        Field = SeedLands(Field, Row + 1, Column + 1)
return Field
```

1 mark Screen output covering autumn of year 1, showing plant with three adjace above, below, to the left or to the right of the plant:



```
Season: autumn  Year number: 1
...................................|  0
...................................|  1
...................................|  2
...................................|  3
...................................|  4
...................................|  5
...................................|  6
...................................|  7
...................................|  8
...................................|  9
.................P.................| 10
................SSS................| 11
...................................| 12
...................................| 13
...................................| 14
...................................| 15
...................................| 16
...................................| 17
...................................| 18
...................................| 19
```

Question 11

1 mark Prompt to save file after "End of Simulation" in `Simulation`

1 mark IF statement to respond to user entering 'Y', 'y', 'Yes', etc. or choosing 'yes' f

1 mark Call to `SaveField`, with `Field` as a parameter:

```
print('End of Simulation')

  save = input('Save file? (Y/N) ')
  if save == 'Y':
    SaveField(Field)
```

1 mark Correct declaration of `SaveField` procedure, with 2D character array requ

1 mark Declaration of file name variable

1 mark File name variable set to user input

1 mark '.txt' suffix concatenated:

```
def SaveField(Field):
  fileName = input('Please enter file name: ')
  fileName = fileName + '.txt'
```

1 mark Object to write to file, constructed using file name variable

1 mark Try-except block with "File error" output in except section

1 mark Loop to iterate through each row

1 mark Nested loop to iterate through column

1 mark Character added to file within nested loop

1 mark Code to add row number at the end of each row

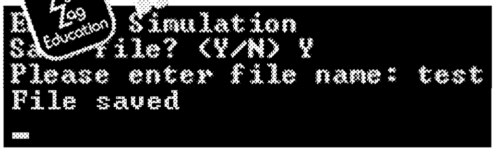
1 mark Carriage return (e.g. \n) in outer loop

1 mark File closed

1 mark "File saved" output:

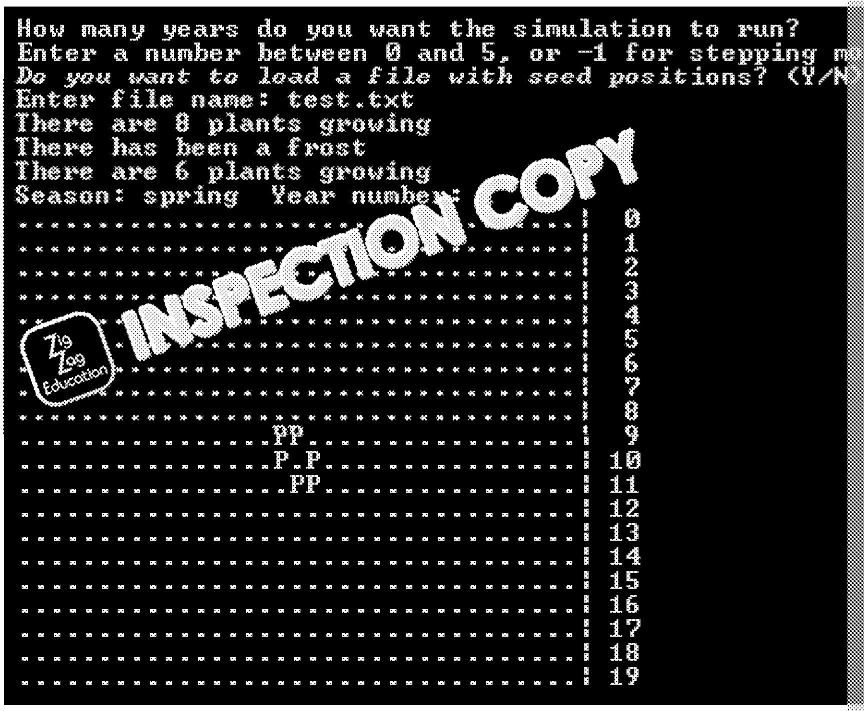
```
try:
  FileHandler = open(fileName, 'w')
  for Row in range(FIELDLENGTH):
    for Column in range(FIELDWIDTH):
      FileHandler.write(Field[Row][Column])
    FileHandler.write('|{0:>3}'.format(Row))
    FileHandler.write('\n')
  FileHandler.close()
  print('File saved')
except:
  print('File error')
```

1 mark Screen output showing file saved as 'test' with "File saved" output:



```
End of Simulation
Save file? (Y/N) Y
Please enter file name: test
File saved
```

1 mark Screen output showing loading of test.txt, with up to eight plants groupec spring of year 1 (NB frost may have occurred):



```
How many years do you want the simulation to run?
Enter a number between 0 and 5, or -1 for stepping m
Do you want to load a file with seed positions? (Y/N
Enter file name: test.txt
There are 8 plants growing
There has been a frost
There are 6 plants growing
Season: spring  Year number:
...................................|  0
...................................|  1
...................................|  2
...................................|  3
...................................|  4
...................................|  5
...................................|  6
...................................|  7
...................................|  8
...............PP..................|  9
...............P.P.................| 10
................PP.................| 11
...................................| 12
...................................| 13
...................................| 14
...................................| 15
...................................| 16
...................................| 17
...................................| 18
...................................| 19
```

Question 12

2 marks Nested loop in `SimulateSummer` to iterate through rows and columns us `FIELDWIDTH` constants. Must be inside ELSE that indicates that there is no point). Max 1 mark if literals 20 and 35 are used.

1 mark If statement to determine whether current element is a plant

1 mark Correct call to `GrowPlants`, with `Field`, `Row` and `Column` passed as pa

```
  print('There has been a severe drought')
  CountPlants(Field)
else:
  for Row in range(FIELDLENGTH):
    for Column in range(FIELDWIDTH):
      if Field[Row][Column] == PLANT:
        GrowPlants(Field, Row, Column)
return Field
```

1 mark Creation of `OFFSHOOT` constant in appropriate place:

```
SOIL = '.'
SEED = 'S'
PLANT = 'P'
ROCKS = 'X'
OFFSHOOT
```

1 mark Correct declaration of `GrowPlants` function, with 2D character array and 2D array as the data type of the return value

3 marks For a single offshoot – north, south, east or west (no diagonals), as follows:

- 1 mark for IF to check that the offshoot would be on the field
- 1 mark for IF to check that target element contains soil
- 1 mark for inserting offshoot constant in correct direction

3 marks 1 mark for each additional correct offshoot; the above 3 marks should be a complete, correct offshoot direction

1 mark Returning the 2D array:

```
def GrowPlants(Field, Row, Column):

  if Row - 1 >= 0 and randint(0, 3) == 0:
    if Field[Row - 1][Column] == SOIL:
      Field[Row - 1][Column] = OFFSHOOT

  if Row + 1 <= FIELDLENGTH - 1 and randint(0, 3) ==
    if Field[Row + 1][Column] == SOIL:
      Field[Row + 1][Column] = OFFSHOOT

  if Column - 1 >= 0 and randint(0, 3) == 0:
    if Field[Row][Column - 1] == SOIL:
      Field[Row][Column - 1] = OFFSHOOT

  if Column + 1 <= FIELDWIDTH - 1 and randint(0, 3)
    if Field[Row][Column + 1] == SOIL:
      Field[Row][Column + 1] = OFFSHOOT

  Return Field
```

1 marks Nested loop at the beginning of `SimulateSummer`

1 marks IF statement to identify offshoots

1 marks Offshoots become plants

```
for Row in range(FIELDLENGTH):
  for Column in range(FIELDWIDTH):
    if Field[Row][Column] == OFFSHOOT:
      Field[Row][Column] = PLANT
```

Question 13

1 mark Constants section updated to include MATURESEED and NEWSEED, with SE

```
SOIL = '.'
MATURESEED = 'S'
NEWSEED = 's'
PLANT = 'P'
ROCKS = 'X'
```

1 mark CreateNewField updated to add NEWSEED instead of SEED

```
Row = FIELDLENGTH // 2
Column = FIELDWIDTH // 2
Field[Row][Column] = NEWSEED
return Field
```

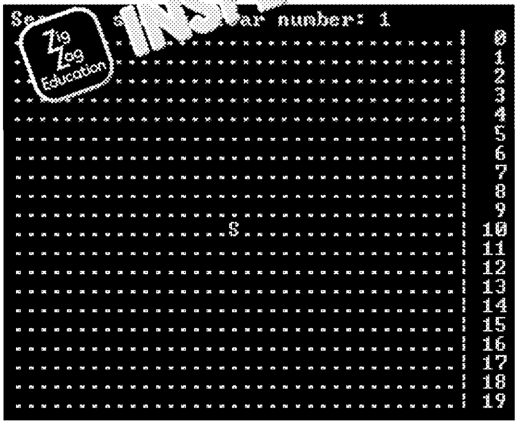
2 marks SimulateSpring updated to turn NEWSEED to MATURESEED and MATU
loop. No marks for this point if NEWSEED becomes MATURESEED and the

```
for Row in range(FIELDLENGTH):
  for Column in range(FIELDWIDTH):
    if Field[Row][Column] == MATURESEED:
      Field[Row][Column] = PLANT
    elif Field[Row][Column] == NEWSEED:
      Field[Row][Column] = MATURESEED
```

1 mark SeedLands drops NEWSEED instead of SEED:

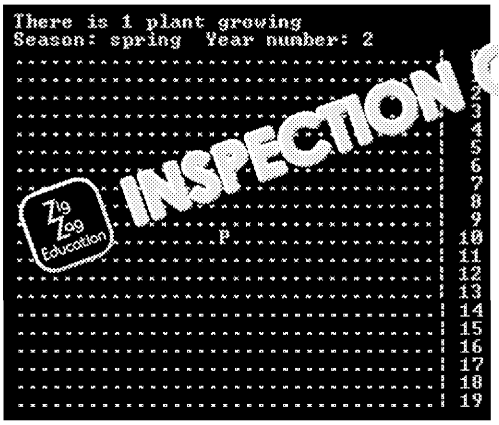
```
if Field[Row][Column] == SOIL:
  Field[Row][Column] = NEWSEED
```

1 mark Spring output from year 1, with a capital 'S' in the middle:



```
Season: spring  Year number: 1
...................................|  0
...................................|  1
...................................|  2
...................................|  3
...................................|  4
...................................|  5
...................................|  6
...................................|  7
...................................|  8
...................................|  9
.................S.................| 10
...................................| 11
...................................| 12
...................................| 13
...................................| 14
...................................| 15
...................................| 16
...................................| 17
...................................| 18
...................................| 19
```

1 mark Spring output from year 2, with 'P' in the middle:



```
There is 1 plant growing
Season: spring  Year number: 2
...................................|  0
...................................|  1
...................................|  2
...................................|  3
...................................|  4
...................................|  5
...................................|  6
...................................|  7
...................................|  8
...................................|  9
.................P.................| 10
...................................| 11
...................................| 12
...................................| 13
...................................| 14
...................................| 15
...................................| 16
...................................| 17
...................................| 18
...................................| 19
```

Question 14

1 mark FIELDLENGTH and FIELDWIDTH now global variables:

```
def CreateNewField():

  global FIELDLENGTH
  global FIELDWIDTH
```

1 mark Prompting user for the length of field in CreateNewField

1 mark Storage of input in FIELDLENGTH

1 mark Check for an input value outside the range 10–50

1 mark Display error message if outside range

1 mark Loop to continue until valid input

1 mark Prompting user for width of field

1 mark Storage of input in FIELDWIDTH

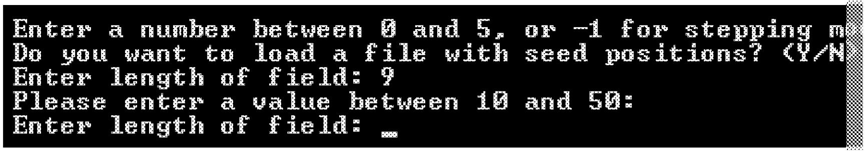
1 mark Check for an input value outside the range 10–50

1 mark Display error message if outside range

1 mark Loop to continue until valid input:

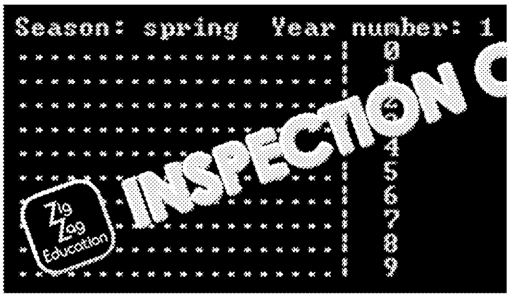
```
while True:
  FIELDLENGTH = int(input('Enter length of field: '))
  if FIELDLENGTH < 10 or FIELDLENGTH > 50:
    print('Please enter a value between 10 and 50: ')
  else:
    break

while True:
  FIELDWIDTH = int(input('Enter width of field: '))
  if FIELDWIDTH < 10 or FIELDWIDTH > 50:
    print('Please enter a value between 10 and 50: ')
  else:
    break
```

1 mark Error displayed if either dimension is entered as ‘9’:



```
Enter a number between 0 and 5, or -1 for stepping m
Do you want to load a file with seed positions? (Y/N
Enter length of field: 9
Please enter a value between 10 and 50:
Enter length of field: _
```

1 mark Input of 10 (length) by 20 (width) dimensions, with field of correct size dis



```
Season: spring  Year number: 1
....................|  0
....................|  1
....................|  2
....................|  3
....................|  4
....................|  5
....................|  6
....................|  7
....................|  8
....................|  9
```

Question 15

1 mark Only plants are subject to disease

1 mark `SimulateAutumn` code modified so that the presence of a plant triggers within an IF statement (or equivalent)

1 mark Existing `SeedLands` function calls will always happen if there is no disea

```
if Field[Row][Column] == PLANT:
  if randint(0, 3) == [illegible]:
    Field = SimulateDisease(Field, Row, Column)
  else:
    Field = SeedLands(Field, Row - 1, Column - 1)
    Field = SeedLands(Field, Row - 1, Column)
```

1 mark `SimulateDisease` function correctly declared, with 2D character array a with the return value's data type being a 2D character array

1 mark Elements affected span two in each direction, horizontally

1 mark Elements affected span two in each direction, vertically

1 mark Code to handle/avoid attempt to access element beyond bounds (horizont

1 mark Code to handle/avoid attempt to access element beyond bounds (vertical)

NB. Both preceding marks above can be awarded for a single technique that capture

1 mark Code to ensure that only plants and seeds are affected

1 mark Affected elements become soil

1 mark Output "Disease has struck!"

1 mark Return of 2D array that always returns correct state of `Field`:

```
def SimulateDisease(Field, Row, Column):

  for rowCount in range(Row -2, Row + 3):
    for columnCount in range(Column - 2, Column + 3):
      try:
        if Field[rowCount][columnCount] == SEED or
        Field[rowCount][columnCount] == PLANT:
          Field[rowCount][columnCount] = SOIL
      except:
        pass
  print('Disease has struck')
  return Field
```

Ideas for modifications	How to

Name	

ZigZag Education supporting

AS AQA Computer Science Paper 1

Summer 2017: `Plant Growing Simu`

Electronic Answer Document (EAD)

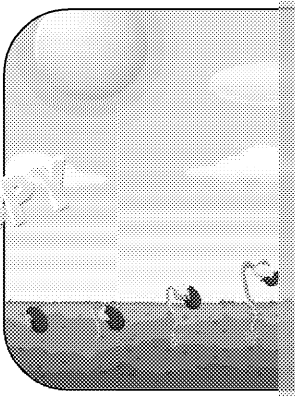
Instructions

- 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 Question

Answer all questions.
Remember to save this document regularly.

Q		Answer
1	(a)	
	(b)	
	(c)	
	(d)	
	(e)	
	(f)	
	(g)	
2		
3		
4	(a)	
	(b)	
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		

Programming Exercises

Answer all questions.
Remember to save this document regularly.

Q	Answer
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	