

AS

Computer Science

Paper 1 (7516/1)

Mark scheme: applicable for all programming languages A, B, C, D and E

7516

June 2017

Version 1.0: Final

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

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

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

AS Computer Science

Paper 1 (7516/1 – applicable to all programming languages A, B C, D and E)

June 2017

The following annotation is used in the mark scheme:

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

Pages # to ## contain the generic mark scheme.

Pages ## to ## contain the program source code specific to the programming languages for questions ##,## and ##;

- pages ## to ## – VB.NET
- pages ## to ## – PYTHON 2
- pages ## to ## – PYTHON 3
- pages ## to ## – PASCAL/Delphi
- pages ## to ## – C#
- pages ## to ## – JAVA

Level of response marking instructions

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

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

Step 1 Determine a level

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

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

Step 2 Determine a mark

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

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

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

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

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

eg

In question **10.1**, the marks available for the AO3 elements are as follows:

AO3 (design) – 3 marks

AO3 (programming) – 6 marks

In question **11.1**, the marks available for the AO3 elements are as follows:

AO3 (design) – 3 marks

AO3 (programming) – 9 marks

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

Qu		Marks															
01	1	All marks AO2 (apply)	4														
		<table border="1"> <thead> <tr> <th>Event</th> <th>Label(s)</th> </tr> </thead> <tbody> <tr> <td>Correct code keyed</td> <td>F</td> </tr> <tr> <td>Door pulled open</td> <td>B</td> </tr> <tr> <td>Door pushed shut</td> <td>A</td> </tr> <tr> <td>New code keyed</td> <td>E</td> </tr> <tr> <td>Press C</td> <td>d, g (I. order)</td> </tr> <tr> <td>Press E</td> <td>h, c (I. order)</td> </tr> </tbody> </table>	Event	Label(s)	Correct code keyed	F	Door pulled open	B	Door pushed shut	A	New code keyed	E	Press C	d, g (I. order)	Press E	h, c (I. order)	
Event	Label(s)																
Correct code keyed	F																
Door pulled open	B																
Door pushed shut	A																
New code keyed	E																
Press C	d, g (I. order)																
Press E	h, c (I. order)																
		<p>1 mark per two correct labels (round down).</p> <p>I. case</p> <p>Note: each label must only be used once (if given more than once, reject all occurrences).</p>															

Qu		Marks																																																						
02	1	5																																																						
All marks AO2 (apply)																																																								
<table border="1"> <thead> <tr> <th>Count</th> <th>HexString</th> <th>Number</th> <th>HexDigit</th> <th>Value</th> <th>Output</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>"A2"</td> <td>0</td> <td>"A"</td> <td>10</td> <td></td> </tr> <tr> <td></td> <td></td> <td>10</td> <td>"2"</td> <td>2</td> <td></td> </tr> <tr> <td></td> <td></td> <td>162</td> <td></td> <td></td> <td>162</td> </tr> <tr> <td>2</td> <td>"1G"</td> <td>0</td> <td>"1"</td> <td>1</td> <td></td> </tr> <tr> <td></td> <td></td> <td>1</td> <td>"G"</td> <td>-1</td> <td></td> </tr> <tr> <td></td> <td></td> <td>15</td> <td></td> <td></td> <td>15</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			Count	HexString	Number	HexDigit	Value	Output	1	"A2"	0	"A"	10				10	"2"	2				162			162	2	"1G"	0	"1"	1				1	"G"	-1				15			15												
Count	HexString	Number	HexDigit	Value	Output																																																			
1	"A2"	0	"A"	10																																																				
		10	"2"	2																																																				
		162			162																																																			
2	"1G"	0	"1"	1																																																				
		1	"G"	-1																																																				
		15			15																																																			
Mark as follows:																																																								
<ol style="list-style-type: none"> Count running over the values 1, 2 with correct sequence of values for HexString ("A2", "1G"); The correct sequence of values in Number column (0, 10, 162, 0, 1, 15); The correct sequence of values in HexDigit column ("A", "2", "1", "G"); The correct sequence of values in Value column (10, 2, 1, -1); The correct sequence of values in Output column (162, 15); 																																																								
<p>A. repeating values in first two columns</p> <p>A. "1G" before "A2"</p> <p>A. string values without quotes</p>																																																								
02	2	2																																																						
All marks for AO2 (analyse)																																																								
<ol style="list-style-type: none"> invalid character produces value -1 from subroutine; -1 should not be used to calculate // deal with -1 seperately // using -1 gives a misleading result; final output should be -1 / error message; 																																																								
MAX 2																																																								

03	1	<p>All marks for AO3 (programming)</p> <p>Mark as follows:</p> <ol style="list-style-type: none"> 1. Correct prompts "Enter a whole number: " "Enter another whole number: " Number1 and Number2 assigned values entered by user; R. if inside loop 2. Number1 and Number2 assigned to Temp1 and Temp2 respectively; 3. WHILE loop with syntax allowed by the programming language and correct condition for termination of the loop; 4. Correct syntax and condition for the IF statement inside attempt at loop 5. Correct contents of THEN and ELSE part 6. Correct output "... is GCF of ... and ..." A. Temp1 instead of Result A. output on more than one line R. if inside loop A. variations on prompts <p>I. minor differences in case and spelling</p> <p>DPT. If different identifiers</p>	6
03	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p>Must match code from 03.1, including prompts on screen capture matching those in code. Code for 03.1 must be sensible.</p> <p>Screen capture(s) showing the requested tests</p> <pre> >>> enter a whole number: 12 enter another whole number: 39 3 is GCF of 12 and 39 >>> </pre>	1
03	3	<p>Mark is for AO2 (analyse)</p> <p>to preserve the original values for later use // otherwise output won't make sense;</p> <p><i>Note: must refer to the fact that original values are needed later</i></p>	1

04	1	<p>Mark is for AO1 (understanding)</p> <p>Frost // Continuing;</p> <p>R. if any additional code I. minor differences in case and spelling R. significant differences in case and spelling</p>	1
04	2	<p>Mark is for AO1 (understanding)</p> <p>GetHowLongToRun;</p> <p>R. if any additional code I. minor differences in case and spelling R. significant differences in case and spelling</p>	1
04	3	<p>Mark is for AO1 (understanding)</p> <p>Field;</p> <p>R. FieldRow R. if any additional code I. minor differences in case and spelling R. significant differences in case and spelling</p>	1
04	4	<p>Mark is for AO1 (understanding)</p> <p>Response // FileName // FieldRow;</p> <p>R. if any additional code I. minor differences in case and spelling R. significant differences in case and spelling</p>	1

05	1	<p>All marks for AO2 (analyse)</p> <ol style="list-style-type: none"> (If the specification for the field size changes) only need to change the values at the beginning of the source code; Makes the simulation/source code more understandable // improves readability of the code; A. easier to read Makes clear that the values are the dimensions of the field // Identifiers convey meaning that the values directly don't; <p>A. Can't change values accidentally; MAX 2</p>	2
05	2	<p>Mark is for AO2 (analyse)</p> <p>It checks that the coordinates of the proposed seed position are within the field boundaries // not outside the bounds of the field;</p> <p>NE. Validates seed position // stops generating an error</p>	1
05	3	<p>All marks for AO2 (analyse)</p> <ol style="list-style-type: none"> add another selection construct; for rainfall equal to 1 (or 2); when the plant count is exactly divisible by 4, change plant to soil; <p>A. any method that guarantees killing of 25% of the plants R. random killing of plants</p> <p>A. equivalent code</p>	3
05	4	<p>1 mark for AO1 (knowledge) and 1 mark for AO2 (apply)</p> <p>Mark as follows:</p> <p>AO1 (knowledge)</p> <p>Integer division // Floor division // DIV;</p> <p>A. Division that always rounds down to the next integer;</p> <p>NE. division on its own</p> <p>AO2 (apply)</p> <p>Row = 10 Column = 17; A. 10, 17 // 17,10</p>	2

06	1	<p>Mark is for AO2 (analyse)</p> <p>InitialiseField;</p> <p>I. minor differences in case and spelling R. if any additional code</p>	1
06	2	<p>Mark is for AO2 (analyse)</p> <p>ReadFile;</p> <p>A. CreateNewField if not given in 06.3; I. minor differences in case and spelling R. if any additional code</p>	1
06	3	<p>Mark is for AO2 (analyse)</p> <p>CreateNewField;</p> <p>A. ReadFile if not given in 06.2 I. minor differences in case and spelling R. if any additional code</p>	1
06	4	<p>All marks for AO1 (understanding)</p> <ol style="list-style-type: none"> 1. Parameters/variables/values/data/arguments are passed; 2. Values are returned from a subroutine; A. reference parameters return updated values in Pascal; 3. Constants are available to all subroutines // A. global variables in Python; <p>MAX 2</p>	2

07	1	<p>All marks for AO2 (analyse)</p> <ol style="list-style-type: none"> 1. Program will load top-most/first rows read // bottom rows are ignored/not used; 2. Program will load left-most/first columns read // rightmost columns are ignored / not used; <p>A. extra data beyond bounds of field are ignored for 2 marks A. read data from top left corner, ignoring extra data for 2 marks A. extra data is ignored for 1 mark</p>	2
07	2	<p>All marks for AO1 (understanding)</p> <ol style="list-style-type: none"> 1. Spring: every seed becomes a plant because there is no frost // every location contains "P"; 2. Summer: no change because there is no drought; 3. Autumn: no seed can land as there is no soil // no seed can land as every location contains "P"; A. no change because there is nowhere for seed to land; 4. Winter: only soil in the field // field will be empty; 	4

08	1	<p>1 mark for AO3 (design) and 5 marks for AO3 (programming)</p> <p>Mark as follows:</p> <p>AO3 (design) – 1 mark:</p> <ol style="list-style-type: none"> Identifying that an iterative statement is required to repeatedly input the data and check that it is valid before returning including a sensible attempt at termination logic; <ol style="list-style-type: none"> recursive method instead of iterative statement <p>AO3 (programming) – 4 marks:</p> <ol style="list-style-type: none"> 'Invalid input' is displayed for any one invalid input; <ol style="list-style-type: none"> if always displays error message Function returns value for all valid inputs, and in no other circumstance; test <code>Year</code> is in range -1 to 5; R. if zero excluded test for non-integer input; <ol style="list-style-type: none"> test for one type of non-integer input (decimal or string) <p>I. minor differences in case and spelling</p>	5
08	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 08.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for 08.1 must be sensible.</i></p> <p>Screen capture(s) showing the requested test being performed and showing the message 'Invalid input' for -2, 6, w, 1.4 but not for 0</p> <p>A. different error message (or none) if it matches 08.1</p> <pre>>>> Welcome to the Plant Growing Simulation 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 mode: -2 Invalid input Enter a number between 0 and 5, or -1 for stepping mode: 6 Invalid input Enter a number between 0 and 5, or -1 for stepping mode: w Invalid input Enter a number between 0 and 5, or -1 for stepping mode: 1.4 Invalid input Enter a number between 0 and 5, or -1 for stepping mode: 0 >>> .</pre>	1

09	1	<p>All marks for AO3 (programming)</p> <ol style="list-style-type: none"> 1. show correct formula for calculating percentage; 2. show correct method for rounding result; <ol style="list-style-type: none"> A. if it shows 15% (or 10%) 	2
09	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE ****</p> <p><i>Must match code from 09.1, including prompts on screen capture matching those in code.</i></p> <p><i>Code for 09.1 must be sensible.</i></p> <p>Screen captures showing the requested test being performed;</p> <p>The first percentage must be 15%. This will be the only percentage. If there has been a frost (see example below) it should be 10%</p> <p>A. 0% if new field created and 09.1 correct A. truncated percentage</p> <pre>Welcome to the Plant Growing Simulation 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 mode: 1 Do you want to load a file with seed positions? (Y/N): Y Enter file name: TestCase.txt There are 103 plants growing 15 % There has been a frost There are 69 plants growing 10 % Season: spring Year number: 1P..... 0X..... 1 2P.PP.PXP.PP.PP..... 3 4P.P.....P..... 5P..... 6P.P.PP.PP.PPX.P.P..... 7P.....P..... 8 ..X..... 9P.P.....P.P..... 10P.X.P..PP.P.P..P..... 11P..P.P.....P.P.....X..... 12P.P..P.P..P.P..... 13P.P..P..P..P.P..... 14PP.PP..... 15P.P.....P.P..... 16</pre>	1

.....XX.....P..... 17
.....P.....P.P..... 18
.....P..... 19

10	1	<p>3 marks for AO3 (design) and 6 marks for AO3 (programming)</p> <p>Note that AO3 (design) marks are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not regardless of whether the solution works.</p> <table border="1"> <thead> <tr> <th style="text-align: center;">Level</th> <th style="text-align: left;">Description</th> <th style="text-align: center;">Mark Range</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">3</td> <td> <p>A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution. Code is written to ensure that all field cells are saved correctly. The formatting of each line has been considered. Appropriate messages are displayed. A formal interface is used to pass the <code>Field</code> data into the subroutine. Most of the appropriate design decisions have been taken.</p> </td> <td style="text-align: center;">7-9</td> </tr> <tr> <td style="text-align: center;">2</td> <td> <p>There is evidence that a line of reasoning has been partially followed. <code>SaveToFile</code> subroutine has been created, but it might only contain code for saving the data without formatting. There is evidence of some appropriate design work.</p> </td> <td style="text-align: center;">4-6</td> </tr> <tr> <td style="text-align: center;">1</td> <td> <p>An attempt has been made to create <code>SaveToFile</code> and some appropriate programming statements have been written. There is insufficient evidence to suggest that a line of reasoning has been followed or that the solution has been designed. The statements written may or may not be syntactically correct and the subroutine will have very little or none of the required functionality. It is unlikely that any of the key design elements of the task have been recognised.</p> </td> <td style="text-align: center;">1-3</td> </tr> </tbody> </table>	Level	Description	Mark Range	3	<p>A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution. Code is written to ensure that all field cells are saved correctly. The formatting of each line has been considered. Appropriate messages are displayed. A formal interface is used to pass the <code>Field</code> data into the subroutine. Most of the appropriate design decisions have been taken.</p>	7-9	2	<p>There is evidence that a line of reasoning has been partially followed. <code>SaveToFile</code> subroutine has been created, but it might only contain code for saving the data without formatting. There is evidence of some appropriate design work.</p>	4-6	1	<p>An attempt has been made to create <code>SaveToFile</code> and some appropriate programming statements have been written. There is insufficient evidence to suggest that a line of reasoning has been followed or that the solution has been designed. The statements written may or may not be syntactically correct and the subroutine will have very little or none of the required functionality. It is unlikely that any of the key design elements of the task have been recognised.</p>	1-3	9
Level	Description	Mark Range													
3	<p>A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution. Code is written to ensure that all field cells are saved correctly. The formatting of each line has been considered. Appropriate messages are displayed. A formal interface is used to pass the <code>Field</code> data into the subroutine. Most of the appropriate design decisions have been taken.</p>	7-9													
2	<p>There is evidence that a line of reasoning has been partially followed. <code>SaveToFile</code> subroutine has been created, but it might only contain code for saving the data without formatting. There is evidence of some appropriate design work.</p>	4-6													
1	<p>An attempt has been made to create <code>SaveToFile</code> and some appropriate programming statements have been written. There is insufficient evidence to suggest that a line of reasoning has been followed or that the solution has been designed. The statements written may or may not be syntactically correct and the subroutine will have very little or none of the required functionality. It is unlikely that any of the key design elements of the task have been recognised.</p>	1-3													

Marking guidance:

AO3 (design) – 3 points

1. Identify the need for a selection statement to act on user response
2. Identify a method to save each array element
3. Identify a method for required formatting to right-align line numbers

AO3 (programming) – 6 points

4. subroutine header with correct parameter
 - A. similar identifier to `SaveToFile`
5. user interaction to allow filename to be entered when save chosen
6. create a text file for writing
7. each array row output on a new line **R.** if new line as line 0
8. "|" and row number added to end of row
 - A. without extra space after "|"
9. subroutine call in suitable place(s) in `Simulation` subroutine:
 - Either: line above or below "End of Simulation"
 - Or: after `FOR` loop **and** after or within `WHILE` loop

Refer answers using nested procedures to Team Leaders

10	2	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE **** <i>Must match code from 10.1, including prompts on screen capture matching those in code. Code for 10.1 must be sensible.</i></p> <p><i>All screen captures must be present for mark to be awarded</i></p> <p>Screen captures showing the requested test being performed; screen capture must show prompt to save and prompt for file name</p> <pre> Season: winter Year number: 2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 End of Simulation Save the current Field state to a text file? (Y/N): Y Enter the chosen filename to save your field data: Test1.txt </pre> <p>A. different filename</p>	1
-----------	----------	---	----------

10	3	<p>Mark is for AO3 (evaluate)</p> <p>**** SCREEN CAPTURE **** <i>Must match screen capture from 10.2 (allow for a frost)</i></p> <p><i>All screen captures must be present for mark to be awarded</i></p> <p>Screen captures showing the requested test being performed;</p> <pre>Welcome to the Plant Growing Simulation 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 mode: 1 Do you want to load a file with seed positions? (Y/N): Y Enter file name: Test1.txt There are 17 plants growing There has been a frost There are 12 plants growing Season: spring Year number: 1 0 1 2 3 4 5 6 7PP.P..... 8P...P..... 9P...P..... 10P...P..... 11P.PP..... 12 13 14 15 16 17 18 19</pre>	1
----	---	---	---

	if no frost:	
	0
	1
	2
	3
	4
	5
	6
	7
 P P P P	8
 P . P P	9
 P . P . P	10
 P P . P	11
 P P P P	12
	13
	14
	15
	16
	17
	18
	19

11	1	3 marks for AO3 (design) and 9 marks for AO3 (programming)	12												
<p>Note that AO3 (design) marks are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not regardless of whether the solution works.</p>															
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Level</th> <th style="text-align: left;">Description</th> <th style="text-align: center;">Mark Range</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">3</td> <td> <p>A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that is efficient.</p> <p>Code is written to ensure that all possible wind directions result in the displacement of the seeds. The Field cells are updated (mostly) correctly. The 'no wind' option has been considered. Appropriate messages are displayed.</p> <p>All of the appropriate design decisions have been taken.</p> </td> <td style="text-align: center;">9-12</td> </tr> <tr> <td style="text-align: center;">2</td> <td> <p>There is evidence that a line of reasoning has been partially followed.</p> <p><code>SimulateAutumn</code> has been adapted, but it might only contain code for some of the wind directions and displacements have not been used correctly with <code>SeedLands</code>.</p> <p>There is evidence of some appropriate design work.</p> </td> <td style="text-align: center;">5-8</td> </tr> <tr> <td style="text-align: center;">1</td> <td> <p>An attempt has been made to alter <code>SimulateAutumn</code> and some appropriate programming statements have been written. There is insufficient evidence to suggest that a line of reasoning has been followed or that the solution has been designed. The statements written may or may not be syntactically correct and the subroutine will have very little or none of the required functionality. It is unlikely that any of the key design elements of the task have been recognised.</p> </td> <td style="text-align: center;">1-4</td> </tr> </tbody> </table>				Level	Description	Mark Range	3	<p>A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that is efficient.</p> <p>Code is written to ensure that all possible wind directions result in the displacement of the seeds. The Field cells are updated (mostly) correctly. The 'no wind' option has been considered. Appropriate messages are displayed.</p> <p>All of the appropriate design decisions have been taken.</p>	9-12	2	<p>There is evidence that a line of reasoning has been partially followed.</p> <p><code>SimulateAutumn</code> has been adapted, but it might only contain code for some of the wind directions and displacements have not been used correctly with <code>SeedLands</code>.</p> <p>There is evidence of some appropriate design work.</p>	5-8	1	<p>An attempt has been made to alter <code>SimulateAutumn</code> and some appropriate programming statements have been written. There is insufficient evidence to suggest that a line of reasoning has been followed or that the solution has been designed. The statements written may or may not be syntactically correct and the subroutine will have very little or none of the required functionality. It is unlikely that any of the key design elements of the task have been recognised.</p>	1-4
Level	Description	Mark Range													
3	<p>A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that is efficient.</p> <p>Code is written to ensure that all possible wind directions result in the displacement of the seeds. The Field cells are updated (mostly) correctly. The 'no wind' option has been considered. Appropriate messages are displayed.</p> <p>All of the appropriate design decisions have been taken.</p>	9-12													
2	<p>There is evidence that a line of reasoning has been partially followed.</p> <p><code>SimulateAutumn</code> has been adapted, but it might only contain code for some of the wind directions and displacements have not been used correctly with <code>SeedLands</code>.</p> <p>There is evidence of some appropriate design work.</p>	5-8													
1	<p>An attempt has been made to alter <code>SimulateAutumn</code> and some appropriate programming statements have been written. There is insufficient evidence to suggest that a line of reasoning has been followed or that the solution has been designed. The statements written may or may not be syntactically correct and the subroutine will have very little or none of the required functionality. It is unlikely that any of the key design elements of the task have been recognised.</p>	1-4													
<p>Marking guidance:</p>															

	<p>AO3 (design) – 3 points</p> <ol style="list-style-type: none"> 1. identifying a method to associate each different random value with a different wind direction, including 'no wind' 2. identifying that a displacement needs to be added to row and/or column 3. identifying a method of solution that does not increase the number of calls to <code>SeedLands</code> and deals with more than one wind direction <p>AO3 (programming) – 9 points</p> <ol style="list-style-type: none"> 4. setting up random number generator correctly generating 9 different values 5. displaying a message about wind in a sensible place in the code 6. correctly displays wind direction associated with the generated random number 7. correctly displaying alternative message when there was no wind 8. adjusting column correctly for east/west wind and leaving row unchanged 9. adjusting row correctly for north/south wind and leaving column unchanged 10. adjusting row and column correctly for one of NW / NE / SW / SE winds 11. adjusting row and column correctly for 2 or 3 of NW / NE / SW / SE winds 12. adjusting row and column correctly for all of NW / NE / SW / SE winds <p>DPT. If direction of wind is interpreted as blowing towards instead of coming from</p>	
--	--	--

None		N		S		E		W		NW		SW		NE		SE	
0	0	1	0	-1	0	0	-1	0	1	1	1	-1	1	1	-1	-1	-1
-1	-1	0	-1	-2	-1	-1	-2	-1	0	0	0	-2	0	0	-2	-2	-2
-1	0	0	0	-2	0	-1	-1	-1	+1	0	+1	-2	+1	0	-1	-2	-1
-1	+1	0	+1	-2	+1	-1	0	-1	+2	0	+2	-2	+2	0	0	-2	0
0	-1	+1	-1	-1	-1	0	-2	0	0	+1	0	-1	0	+1	-2	-1	-2
0	+1	+1	+1	-1	+1	0	0	0	+2	+1	+2	-1	+2	+1	0	-1	0
+1	-1	+2	-1	0	-1	+1	-2	+1	0	+2	0	0	0	+2	-2	0	-2
+1	0	+2	0	0	0	+1	-1	+1	+1	+2	+1	0	+1	+2	-1	0	-1
+1	+1	+2	+1	0	+1	+1	0	+1	+2	+2	+2	0	+2	+2	0	0	0

Prevaling wind: South	
Season: autumn Year number: 1	
.....	6
.....	7
.....SSS.....	8
.....S.S.....	9
.....SPS.....	10
.....	11
.....	12
.....	13
.....	14
Prevaling wind: East	
Season: autumn Year number: 1	
.....	6
.....	7
.....	8
.....SSS.....	9
.....S.P.....	10
.....SSS.....	11
.....	12
.....	13
.....	14
Prevaling wind: West	
Season: autumn Year number: 1	
.....	6
.....	7
.....	8
.....SSS.....	9
.....P.S.....	10
.....SSS.....	11
.....	12
.....	13
.....	14
Prevaling wind: NorthWest	
Season: autumn Year number: 1	
.....	6
.....	7
.....	8
.....	9
.....PSS.....	10
.....S.S.....	11
.....SSS.....	12
.....	13
.....	14
Prevaling wind: Southwest	
Season: autumn Year number: 1	
.....	6
.....	7

	SSS..... 8	
	S.S..... 9	
	PSS..... 10	
	 11	
	 12	
	 13	
	 14	
		Prevailing wind: Northeast	
		Season: autumn Year number: 1	
	 6	
	 7	
	 8	
	 9	
	SSP..... 10	
	S.S..... 11	
	SSS..... 12	
	 13	
	 14	
		Prevailing wind: Southeast	
		Season: autumn Year number: 1	
	 6	
	 7	
	SSS..... 8	
	S.S..... 9	
	SSP..... 10	
	 11	
	 12	
	 13	
	 14	

Python 2

03	1	<pre> Number1 = int(raw_input("Enter a whole number: ")) Number2 = int(raw_input("Enter another whole number: ")) Temp1 = Number1 Temp2 = Number2 while Temp1 != Temp2: if Temp1 > Temp2: Temp1 = Temp1 - Temp2 else: Temp2 = Temp2 - Temp1 Result = Temp1 print Result, " is GCF of ", Number1, " and ", Number2 </pre>	6
08	1	<pre> def GetHowLongToRun(): print "Welcome to the Plant Growing Simulation" print print "You can step through the simulation a year at a time" print "or run the simulation for 0 to 5 years" print "How many years do you want the simulation to run?" Valid = False while not Valid: try: # catch non-integer input Years = int(raw_input("Enter a number between 0 and 5, or -1 for stepping mode: ")) if Years >= -1 and Years <= 5: Valid = True except: pass if not Valid: print "Invalid input" return Years </pre>	5
09	1	<pre> def CountPlants(Field): NumberOfPlants = 0 for Row in range(FIELDLLENGTH): for Column in range(FIELDWIDTH): if Field[Row][Column] == PLANT: NumberOfPlants += 1 if NumberOfPlants == 1: print "There is 1 plant growing" else: print "There are", NumberOfPlants, "plants growing" TotalCells = FIELDWIDTH * FIELDLLENGTH Percentage = int(round((NumberOfPlants * 100.0) / TotalCells)) print Percentage , "%" </pre>	2

10	1	<pre> def SaveToFile(Field): Response = raw_input('Save the current Field state to a text file? (Y/N): ') if Response == 'Y': FileName = raw_input('Enter the chosen filename to save your field data: ') FileHandle = open(FileName, 'w') for Row in range(FIELDLLENGTH): for Column in range(FIELDWIDTH): FileHandle.write(Field[Row][Column]) FileHandle.write(' {0:>3}'.format(Row)) FileHandle.write('\n') FileHandle.close() def Simulation(): YearsToRun = GetHowLongToRun() if YearsToRun != 0: Field = InitialiseField() if YearsToRun >= 1: for Year in range(1, YearsToRun + 1): SimulateOneYear(Field, Year) else: Continuing = True Year = 0 while Continuing: Year += 1 SimulateOneYear(Field, Year) Response = raw_input("Press Enter to run simulation for another Year, Input X to stop: ") if Response == "x" or Response == "X": Continuing = False print "End of Simulation" SaveToFile(Field) raw_input() </pre>	9
11	1	<pre> def SimulateAutumn(Field): Direction = ['None', 'East', 'West', 'North', 'South', 'Southeast', 'Northeast', 'Southwest', 'Northwest'] PrevailingWind = randint(0,8) WindDirection = Direction[PrevailingWind] ColumnDisplacement = 0 RowDisplacement = 0 if WindDirection == 'East': ColumnDisplacement = -1 elif WindDirection == 'West': ColumnDisplacement = 1 elif WindDirection == 'North': </pre>	12

```
    RowDisplacement = 1
elif WindDirection == 'South':
    RowDisplacement = -1
elif WindDirection == 'Southeast':
    RowDisplacement = -1
    ColumnDisplacement = -1
elif WindDirection == 'Northeast':
    RowDisplacement = 1
    ColumnDisplacement = -1
elif WindDirection == 'Southwest':
    RowDisplacement = -1
    ColumnDisplacement = 1
elif WindDirection == 'Northwest':
    RowDisplacement = 1
    ColumnDisplacement = 1
if PrevailingWind == 0:
    print 'There was no wind this season'
else:
    print 'Prevailing wind: ', WindDirection
for Row in range(FIELDDLENGTH):
    for Column in range(FIELDWIDTH):
        if Field[Row][Column] == PLANT:
            Row = Row + RowDisplacement
            Column = Column + ColumnDisplacement
            Field = SeedLands(Field, Row - 1, Column - 1)
            Field = SeedLands(Field, Row - 1, Column)
            Field = SeedLands(Field, Row - 1, Column + 1)
            Field = SeedLands(Field, Row, Column - 1)
            Field = SeedLands(Field, Row, Column + 1)
            Field = SeedLands(Field, Row + 1, Column - 1)
            Field = SeedLands(Field, Row + 1, Column)
            Field = SeedLands(Field, Row + 1, Column + 1)
return Field
```

Python 3

03	1	<pre> Number1 = int(input("Enter a whole number: ")) Number2 = int(input("Enter another whole number: ")) Temp1 = Number1 Temp2 = Number2 while Temp1 != Temp2: if Temp1 > Temp2: Temp1 = Temp1 - Temp2 else: Temp2 = Temp2 - Temp1 Result = Temp1 print(Result, " is GCF of ", Number1, " and ", Number2) </pre>	6
08	1	<pre> def GetHowLongToRun(): print('Welcome to the Plant Growing Simulation') print() print('You can step through the simulation a year at a time') print('or run the simulation for 0 to 5 years') print('How many years do you want the simulation to run?') Valid = False while not Valid: try: # catch non-integer input Years = int(input('Enter a number between 0 and 5, or -1 for stepping mode: ')) if Years >= -1 and Years <= 5: Valid = True except: pass if not Valid: print('Invalid input') return Years </pre>	5
09	1	<pre> def CountPlants(Field): NumberOfPlants = 0 for Row in range(FIELDLENGTH): for Column in range(FIELDWIDTH): if Field[Row][Column] == PLANT: NumberOfPlants += 1 if NumberOfPlants == 1: print('There is 1 plant growing') else: print('There are', NumberOfPlants, 'plants growing') TotalCells = FIELDWIDTH * FIELDLENGTH Percentage = round((NumberOfPlants / TotalCells)* 100) print(Percentage, '%') </pre>	2

10	1	<pre> def SaveToFile(Field): Response = input('Save the current Field state to a text file? (Y/N): ') if Response == 'Y': FileName = input('Enter the chosen filename to save your field data: ') FileHandle = open(FileName, 'w') for Row in range(FIELDLENGTH): for Column in range(FIELDWIDTH): FileHandle.write(Field[Row][Column]) FileHandle.write(' {0:>3}'.format(Row)) FileHandle.write('\n') FileHandle.close() def Simulation(): YearsToRun = GetHowLongToRun() if YearsToRun != 0: Field = InitialiseField() if YearsToRun >= 1: for Year in range(1, YearsToRun + 1): SimulateOneYear(Field, Year) else: Continuing = True Year = 0 while Continuing: Year += 1 SimulateOneYear(Field, Year) Response = input('Press Enter to run simulation for another Year, Input X to stop: ') if Response == 'x' or Response == 'X': Continuing = False print('End of Simulation') SaveToFile(Field) input() </pre>	9

11	1	<pre> def SimulateAutumn(Field): Direction = ['None', 'East', 'West', 'North', 'South', 'Southeast', 'Northeast', 'Southwest', 'Northwest'] PrevailingWind = randint(0,8) WindDirection = Direction[PrevailingWind] ColumnDisplacement = 0 RowDisplacement = 0 if WindDirection == 'East': ColumnDisplacement = -1 elif WindDirection == 'West': ColumnDisplacement = 1 elif WindDirection == 'North': RowDisplacement = 1 elif WindDirection == 'South': RowDisplacement = -1 elif WindDirection == 'Southeast': RowDisplacement = -1 ColumnDisplacement = -1 elif WindDirection == 'Northeast': RowDisplacement = 1 ColumnDisplacement = -1 elif WindDirection == 'Southwest': RowDisplacement = -1 ColumnDisplacement = 1 elif WindDirection == 'Northwest': RowDisplacement = 1 ColumnDisplacement = 1 if PrevailingWind == 0: print('There was no wind this season') else: print('Prevailing wind: ', WindDirection) for Row in range(FIELDLENGTH): for Column in range(FIELDWIDTH): if Field[Row][Column] == PLANT: Row = Row + RowDisplacement Column = Column + ColumnDisplacement Field = SeedLands(Field, Row - 1, Column - 1) Field = SeedLands(Field, Row - 1, Column) Field = SeedLands(Field, Row - 1, Column + 1) Field = SeedLands(Field, Row, Column - 1) Field = SeedLands(Field, Row, Column + 1) Field = SeedLands(Field, Row + 1, Column - 1) Field = SeedLands(Field, Row + 1, Column) Field = SeedLands(Field, Row + 1, Column + 1) return Field </pre>	12
----	---	--	----

Alternative answer

```

def SimulateAutumn(Field):
    Direction = ['None', 'East', 'West', 'North', 'South',
                'Southeast', 'Northeast', 'Southwest', 'Northwest']
    PrevailingWind = randint(0,8)
    WindDirection = Direction[PrevailingWind]
    ColumnDisplacement = 0
    RowDisplacement = 0
    if WindDirection == 'East':
        ColumnDisplacement = -1
    elif WindDirection == 'West':
        ColumnDisplacement = 1
    elif WindDirection == 'North':
        RowDisplacement = 1
    elif WindDirection == 'South':
        RowDisplacement = -1
    elif WindDirection == 'Southeast':
        RowDisplacement = -1
        ColumnDisplacement = -1
    elif WindDirection == 'Northeast':
        RowDisplacement = 1
        ColumnDisplacement = -1
    elif WindDirection == 'Southwest':
        RowDisplacement = -1
        ColumnDisplacement = 1
    elif WindDirection == 'Northwest':
        RowDisplacement = 1
        ColumnDisplacement = 1
    if PrevailingWind == 0:
        print('There was no wind this season')
    else:
        print('Prevailing wind: ', WindDirection)
    for Row in range(FIELDDLENGTH):
        for Column in range(FIELDWIDTH):
            if Field[Row][Column] == PLANT:
                Field = SeedLands(Field, Row - 1 +
RowDisplacement, Column - 1 + ColumnDisplacement)
                Field = SeedLands(Field, Row - 1 +
RowDisplacement, Column + ColumnDisplacement)
                Field = SeedLands(Field, Row - 1 +
RowDisplacement, Column + 1 + ColumnDisplacement)
                Field = SeedLands(Field, Row + RowDisplacement,
Column - 1 + ColumnDisplacement)
                Field = SeedLands(Field, Row + RowDisplacement,
Column + 1 + ColumnDisplacement)
                Field = SeedLands(Field, Row + 1 +
RowDisplacement, Column - 1 + ColumnDisplacement)

```

		<pre>Field = SeedLands(Field, Row + 1 + RowDisplacement, Column + ColumnDisplacement) Field = SeedLands(Field, Row + 1 + RowDisplacement, Column + 1 + ColumnDisplacement) return Field</pre>	
--	--	---	--

VB.NET

03	1	<pre> Sub Main() Dim Number1 As Integer Dim Number2 As Integer Dim Temp1 As Integer Dim Temp2 As Integer Dim Result As Integer Console.Write("Enter a whole number: ") Number1 = Console.ReadLine Console.Write("Enter another whole number: ") Number2 = Console.ReadLine Temp1 = Number1 Temp2 = Number2 While Temp1 <> Temp2 If Temp1 > Temp2 Then Temp1 = Temp1 - Temp2 Else Temp2 = Temp2 - Temp1 End If End While Result = Temp1 Console.WriteLine(Result & " is GCF of " & Number1 & " and " & Number2) Console.ReadLine() End Sub </pre>	6
08	1	<pre> ... Console.WriteLine("How many years do you want the simulation to run?") Dim Valid As Boolean = False While Not Valid Try Console.Write("Enter a number between 0 and 5, or -1 for stepping mode: ") Years = Convert.ToInt32(Console.ReadLine()); If Years >= -1 And Years <= 5 Then Valid = True End If Catch End Try If Not Valid Then Console.WriteLine("Invalid input") End If End While Return Years ... Alternative answer ... Console.WriteLine("How many years do you want the simulation to run?") While True Try Console.Write("Enter a number between 0 and 5, or -1 for stepping: ") Years = Convert.ToInt32(Console.ReadLine()); </pre>	5

		<pre> If Years >= -1 And Years <= 5 Then Exit While End If Catch End Try Console.WriteLine("Invalid input") End While Return Years ... </pre>	
09	1	<pre> ... Console.WriteLine("There are " & NumberOfPlants & " plants growing") End If Dim TotalCells As Integer Dim Percentage As Integer TotalCells = FIELDWIDTH * FIELDLENGTH Percentage = Math.Round((NumberOfPlants / TotalCells) * 100) Console.WriteLine(Percentage & "%") ... Alternative answer ... Console.WriteLine("There are " & NumberOfPlants & " plants growing") End If Console.WriteLine(Math.Round((NumberOfPlants / (FIELDWIDTH * FIELDLENGTH)) * 100) & "%") ... </pre>	2
10	1	<pre> Sub SaveToFile(ByVal Field(,) As Char) Dim Response As String Dim Row As Integer Dim Column As Integer Dim FileName As String Dim FileHandle As IO.StreamWriter Console.Write("Save the current Field state to a text file? (Y/N): ") Response = Console.ReadLine() If Response = "Y" Then Console.Write("Enter the chosen filename to save your field data: ") FileName = Console.ReadLine() FileHandle = New IO.StreamWriter(FileName) For Row = 0 To FIELDLENGTH - 1 For Column = 0 To FIELDWIDTH - 1 FileHandle.write(Field(Row, Column)) Next FileHandle.Write(" " & Str(Row).PadLeft(3) & vbCrLf) Next FileHandle.close() End If End Sub Sub Simulation() </pre>	9

		<pre> Dim YearsToRun As Integer Dim Continuing As Boolean Dim Response As String Dim Year As Integer Dim Field(FIELDWIDTH, FIELDLENGTH) As Char While True YearsToRun = GetHowLongToRun() If YearsToRun <> 0 Then Field = InitialiseField() If YearsToRun >= 1 Then For Year = 1 To YearsToRun SimulateOneYear(Field, Year) Next Else Continuing = True Year = 0 While Continuing Year += 1 SimulateOneYear(Field, Year) Console.WriteLine("Press Enter to run simulation for another Year, Input X to stop: ") Response = Console.ReadLine() If Response = "x" Or Response = "X" Then Continuing = False End If End While End If Console.WriteLine("End of Simulation") SaveToFile(Field) End If Console.ReadLine() End While End Sub </pre>	
11	1	<pre> Function SimulateAutumn(ByVal Field(,) As Char) As Char(,) Dim RowDisplacement As Integer Dim ColumnDisplacement As Integer Dim PrevailingWind As Integer Dim WindDirection As String Dim Direction() As String = {"None", "East", "West", "North", "South", "Southeast", "Northeast", "Southwest", "Northwest"} PrevailingWind = Int(Rnd() * 10) WindDirection = Direction(PrevailingWind) ColumnDisplacement = 0 RowDisplacement = 0 If WindDirection = "East" Then ColumnDisplacement = -1 ElseIf WindDirection = "West" Then ColumnDisplacement = 1 ElseIf WindDirection = "North" Then RowDisplacement = 1 ElseIf WindDirection = "South" Then RowDisplacement = -1 ElseIf WindDirection = "Southeast" Then RowDisplacement = -1 ColumnDisplacement = -1 ElseIf WindDirection = "Northeast" Then RowDisplacement = 1 </pre>	12

```
ColumnDisplacement = -1
ElseIf WindDirection = "Southwest" Then
    RowDisplacement = -1
    ColumnDisplacement = 1
ElseIf WindDirection = "Northwest" Then
    RowDisplacement = 1
    ColumnDisplacement = 1
End If
If PrevailingWind = 0 Then
    Console.WriteLine("There was no wind this season")
Else
    Console.WriteLine("Prevailing wind: " & WindDirection)
End If
For Row = 0 To FIELDLENGTH - 1
    For Column = 0 To FIELDWIDTH - 1
        If Field(Row, Column) = Plant Then
            Field = SeedLands(Field, Row + RowDisplacement - 1, Column +
ColumnDisplacement - 1)
            Field = SeedLands(Field, Row + RowDisplacement - 1, Column +
ColumnDisplacement)
            Field = SeedLands(Field, Row + RowDisplacement - 1, Column +
ColumnDisplacement + 1)
            Field = SeedLands(Field, Row + RowDisplacement, Column +
ColumnDisplacement - 1)
            Field = SeedLands(Field, Row + RowDisplacement, Column +
ColumnDisplacement + 1)
            Field = SeedLands(Field, Row + RowDisplacement + 1, Column +
ColumnDisplacement - 1)
            Field = SeedLands(Field, Row + RowDisplacement + 1, Column +
ColumnDisplacement)
            Field = SeedLands(Field, Row + RowDisplacement + 1, Column +
ColumnDisplacement + 1)
        End If
    Next
Next
Return Field
End Function
```

Pascal

03	1	<pre> program Project2; {\$APPTYPE CONSOLE} uses SysUtils; var Number1, Number2 : Integer; Temp1, Temp2 : Integer; Result : Integer; begin Write('Enter a whole number: '); Readln(Number1); Write('Enter another whole number: '); Readln(Number2); Temp1 := Number1; Temp2 := Number2; while Temp1 <> Temp2 do if Temp1 > Temp2 then Temp1 := Temp1 - Temp2 else Temp2 := Temp2 - Temp1; Result := Temp1; Write(Result, ' is GCF of ', Number1, ' and ', Number2); Readln; end. </pre>	6
08	1	<pre> Function GetHowLongToRun() : Integer; Var Valid : Boolean; Years : Integer; Begin Writeln('Welcome to the Plant Growing Simulation'); Writeln; Writeln('You can step through the simulation a year at a time'); Writeln('or run the simulation for 0 to 5 years'); Writeln('How many years do you want the simulation to run?'); Valid := False; While Not Valid Do Begin Try Write('Enter a number between 0 and 5, or -1 for stepping mode: '); Readln(Years); If (Years >= -1) And (Years <= 5) Then Valid := True; Except </pre>	5

		<pre> End; If Not Valid Then Writeln('Invalid input'); End; GetHowLongToRun := Years; End; </pre>	
09	1	<pre> Procedure CountPlants(Field : TField); Var TotalCells, Percentage : Integer; NumberOfPlants : Integer; Row, Column : Integer; Begin NumberOfPlants := 0; For Row := 0 To FIELDLENGTH - 1 Do For Column := 0 To FIELDWIDTH - 1 Do If Field[Row][Column] = PLANT Then NumberOfPlants := NumberOfPlants + 1; If NumberOfPlants = 1 Then Writeln('There is 1 plant growing') Else Writeln('There are ', NumberOfPlants, ' plants growing'); TotalCells := FIELDWIDTH * FIELDLENGTH; Percentage := Round((NumberOfPlants / TotalCells) * 100); Writeln(Percentage, '%'); End; </pre>	2
10	1	<pre> Procedure SaveToFile(Field : TField); Var Response : Char; Row, Column : Integer; FileName : String; FileHandle : Text; Begin Write('Save the current Field state to a text file? (Y/N): '); Readln(Response); If Response = 'Y' Then Begin Write('Enter the chosen filename to save your field data: '); Readln(FileName); AssignFile(FileHandle, FileName); Rewrite(FileHandle); For Row := 0 To FIELDLENGTH - 1 Do Begin For Column := 0 To FIELDWIDTH - 1 Do Write(FileHandle, Field[Row][Column]); Writeln(FileHandle, ' ', Row:3); End; CloseFile(FileHandle); </pre>	9

		<pre> End; End; Procedure Simulation(); Var YearsToRun, Year : Integer; Field : TField; Continuing : Boolean; Response : String; Begin YearsToRun := GetHowLongToRun(); If YearsToRun <> 0 Then Begin Field := InitialiseField(); If YearsToRun >= 1 Then For Year := 1 To YearsToRun Do SimulateOneYear(Field, Year) Else Begin Continuing := True; Year := 0; While Continuing = True Do Begin Year := Year + 1; SimulateOneYear(Field, Year); Write('Press Enter to run simulation for another Year, Input X to stop: '); Readln(Response); If (Response = 'x') Or (Response = 'X') Then Continuing := False; End; End; Writeln('End of Simulation'); SaveToFile(Field); End; Readln; End; End; </pre>	
11	1	<pre> Function SimulateAutumn(Field : TField) : TField; Var PrevailingWind, ColumnDisplacement, RowDisplacement, Row, Column : Integer; WindDirection : String; Direction : Array [0..8] Of String; Begin Direction[0] := 'None'; Direction[1] := 'East'; Direction[2] := 'West'; Direction[3] := 'North'; Direction[4] := 'South'; Direction[5] := 'Southeast'; Direction[6] := 'Northeast'; </pre>	12

```

Direction[7] := 'Southwest';
Direction[8] := 'Northwest';
PrevailingWind := Random(9);
WindDirection := Direction[PrevailingWind];
ColumnDisplacement := 0;
RowDisplacement := 0;
Case PrevailingWind of
1 : ColumnDisplacement := -1;
2 : ColumnDisplacement := 1;
3 : RowDisplacement := 1;
4 : RowDisplacement := -1;
5 : Begin
    RowDisplacement := -1;
    ColumnDisplacement := -1;
End;
6 : Begin
    RowDisplacement := 1;
    ColumnDisplacement := -1;
End;
7 : Begin
    RowDisplacement := -1;
    ColumnDisplacement := 1;
End;
8 : Begin
    RowDisplacement := 1;
    ColumnDisplacement := 1;
End;
End;
If PrevailingWind = 0 Then
    Writeln('There was no wind this season')
Else
    Writeln('Prevailing wind: ', WindDirection);

For Row := 0 To FIELDLENGTH - 1 Do
    For Column := 0 To FIELDWIDTH - 1 Do
        If Field[Row][Column] = PLANT Then
            Begin
                Field := SeedLands(Field, Row + RowDisplacement - 1,
Column + ColumnDisplacement - 1);
                Field := SeedLands(Field, Row + RowDisplacement - 1,
Column + ColumnDisplacement);
                Field := SeedLands(Field, Row + RowDisplacement - 1,
Column + ColumnDisplacement + 1);
                Field := SeedLands(Field, Row + RowDisplacement,
Column + ColumnDisplacement - 1);
                Field := SeedLands(Field, Row + RowDisplacement,
Column + ColumnDisplacement + 1);
                Field := SeedLands(Field, Row + RowDisplacement + 1,
Column + ColumnDisplacement - 1);
                Field := SeedLands(Field, Row + RowDisplacement + 1,
Column + ColumnDisplacement);
                Field := SeedLands(Field, Row + RowDisplacement + 1,
Column + ColumnDisplacement + 1);
            End;
        End;
    End;
End;

```

		<pre>End; SimulateAutumn := Field; End;</pre>	
--	--	---	--

C#

03	1	<pre> static void Main(string[] args) { int Number1 = 0, Number2 = 0; int Temp1 = 0, Temp2 = 0; int Result = 0; Console.Write("Enter a whole number: "); Number1 = Convert.ToInt32(Console.ReadLine()); Console.Write("Enter another whole number: "); Number2 = Convert.ToInt32(Console.ReadLine()); Temp1 = Number1; Temp2 = Number2; while (Temp1 != Temp2) { if (Temp1 > Temp2) { Temp1 = Temp1 - Temp2; } else { Temp2 = Temp2 - Temp1; } } Result = Temp1; Console.WriteLine(Result + " is GCF of " + Number1 + " and " + Number2); Console.ReadLine(); } </pre>	6
08	1	<pre> static int GetHowLongToRun() { int Years = 0; bool Valid = false; Console.WriteLine("Welcome to the Plant Growing Simulation"); Console.WriteLine(); Console.WriteLine("You can step through the simulation a year at a time"); Console.WriteLine("or run the simulation for 0 to 5 years"); Console.WriteLine("How many years do you want the simulation to run?"); Console.Write("Enter a number between 0 and 5, or -1 for stepping mode: "); while (!Valid) { try { Years = Convert.ToInt32(Console.ReadLine()); </pre>	5

		<pre> if (Years >= -1 && Years <= 5) { Valid = true; } } catch (Exception) { } if (!Valid) { Console.WriteLine("Invalid input"); } } return Years; } </pre>	
09	1	<pre> static void CountPlants(char[,] Field) { int NumberOfPlants = 0; int TotalCells = 0; double Percentage = 0; for (int Row = 0; Row < FIELDLENGTH; Row++) { for (int Column = 0; Column < FIELDWIDTH; Column++) { if (Field[Row, Column] == PLANT) { NumberOfPlants++; } } } if (NumberOfPlants == 1) { Console.WriteLine("There is 1 plant growing"); } else { Console.WriteLine("There are " + NumberOfPlants + " plants growing"); } TotalCells = FIELDLENGTH * FIELDWIDTH; Percentage = (((double)NumberOfPlants / (double)TotalCells) * 100.0); Console.WriteLine(Math.Round(Percentage) + "%"); } </pre>	2
10	1	<pre> private static void SaveToFile(char[,] Field) { string Response = "", FileName = ""; Console.Write("Save the current Field state to a text file? (Y/N):"); } </pre>	9

```
Response = Console.ReadLine();
if (Response == "Y")
{
    Console.WriteLine("Enter the File Name ");
    FileName = Console.ReadLine();
    StreamWriter CurrentFile = new StreamWriter(FileName);
    for (int Row = 0; Row < FIELDLENGTH; Row++)
    {
        for (int Column = 0; Column < FIELDWIDTH; Column++)
        {
            CurrentFile.Write(Field[Row, Column]);
        }
        CurrentFile.WriteLine("| " + String.Format("{0,3}",
Row));
    }
    CurrentFile.Close();
}

static void Simulation()
{
    int YearsToRun;
    char[,] Field = new char[FIELDLENGTH, FIELDWIDTH];
    bool Continuing;
    int Year;
    string Response;
    YearsToRun = GetHowLongToRun();
    if (YearsToRun != 0)
    {
        InitialiseField(ref Field);
        if (YearsToRun >= 1)
        {
            for (Year = 1; Year <= YearsToRun + 1; Year++)
            {
                SimulateOneYear(Field, Year);
            }
        }
        else
        {
            Continuing = true;
            Year = 0;
            while (Continuing)
            {
                Year++;
                SimulateOneYear(Field, Year);
                Console.WriteLine("Press Enter to run simulation for
another Year, Input X to stop: ");
                Response = Console.ReadLine();
                if (Response == "x" || Response == "X")
                {
                    Continuing = false;
                }
            }
        }
    }
}
```

		<pre> } } } Console.WriteLine("End of Simulation"); SaveToFile(Field); } Console.ReadLine(); } </pre>	
11	1	<pre> static void SimulateAutumn(char[,] Field) { string[] Direction = new string[] {"None", "East", "West", "North", "South", "Southeast", "Northeast", "Southwest", "Northwest"}; Random RNDWindDirection = new Random(); int PrevailingWind = RNDWindDirection.Next(0, 9); string WindDirection = Direction[PrevailingWind]; int ColumnDisplacement = 0, RowDisplacement = 0; if (WindDirection == "East") { ColumnDisplacement = -1; } else if(WindDirection == "West") { ColumnDisplacement = 1; } else if (WindDirection == "North") { RowDisplacement = 1; } else if (WindDirection == "South") { RowDisplacement = -1; } else if (WindDirection == "Southeast") { RowDisplacement = -1; ColumnDisplacement = -1; } else if (WindDirection == "Northeast") { RowDisplacement = 1; ColumnDisplacement = -1; } else if (WindDirection == "Southwest") { RowDisplacement = -1; ColumnDisplacement = 1; } else if (WindDirection == "Northwest") { RowDisplacement = 1; ColumnDisplacement = 1; } } </pre>	12

```
}
if (WindDirection == "None")
{
    Console.WriteLine("There was no wind this season");
}
else
{
    Console.WriteLine("Prevailing wind: " + WindDirection );
}

for (int Row = 0; Row < FIELDLENGTH; Row++)
{
    for (int Column = 0; Column < FIELDWIDTH; Column++)
    {
        if (Field[Row, Column] == PLANT)
        {
            SeedLands(Field, Row - 1 + RowDisplacement , Column -
1 + ColumnDisplacement);
            SeedLands(Field, Row - 1 + RowDisplacement, Column +
ColumnDisplacement);
            SeedLands(Field, Row - 1 + RowDisplacement, Column +
1 + ColumnDisplacement);
            SeedLands(Field, Row + RowDisplacement, Column - 1 +
ColumnDisplacement);
            SeedLands(Field, Row + RowDisplacement, Column + 1 +
ColumnDisplacement);
            SeedLands(Field, Row + 1 + RowDisplacement, Column -
1 + ColumnDisplacement);
            SeedLands(Field, Row + 1 + RowDisplacement, Column +
ColumnDisplacement);
            SeedLands(Field, Row + 1 + RowDisplacement, Column +
1 + ColumnDisplacement);
        }
    }
}
}
```

Java

03	1	<pre> public static void main(String[] args) { int Number1 = 0; int Number2 = 0; int Temp1 = 0; int Temp2 = 0; int Result = 0; Number1 = Console.readInteger("Enter a whole number: "); Number2 = Console.readInteger("Enter another whole number: "); Temp1 = Number1; Temp2 = Number2; while (Temp1 != Temp2) { if (Temp1 > Temp2) { Temp1 = Temp1 - Temp2; } else { Temp2 = Temp2 - Temp1; } } Result = Temp1; Console.println(Result + " is GCF of " + Number1 + " and " + Number2); } </pre>	6
08	1	<pre> static int GetHowLongToRun() { int Years = 0; boolean Valid = false; Console.println("Welcome to the Plant Growing Simulation"); Console.println(); Console.println("You can step through the simulation a year at a time"); Console.println("or run the simulation for 0 to 5 years"); Console.println("How many years do you want the simulation to run?"); while(!Valid) { try { Years = Console.readInteger("Enter a number between 0 and 5, or -1 for stepping mode: "); if(Years >= -1 && Years <=5) { Valid = true; } } } } </pre>	5

		<pre> catch(Exception e) { } if (!Valid) { Console.WriteLine("Invalid input"); } } return Years; } </pre>	
09	1	<pre> static void CountPlants(char[][] Field) { int NumberOfPlants = 0; int TotalCells = 0; double Percentage = 0; for (int Row = 0; Row < FIELDLENGTH; Row++) { for (int Column = 0; Column < FIELDWIDTH; Column++) { if (Field[Row][Column] == PLANT) { NumberOfPlants++; } } } if (NumberOfPlants == 1) { Console.WriteLine("There is 1 plant growing"); } else { Console.WriteLine("There are " + NumberOfPlants + " plants growing"); } TotalCells = FIELDLENGTH * FIELDWIDTH; Percentage = ((double)NumberOfPlants/(double>TotalCells)*100.0; Console.WriteLine(Math.round(Percentage) + "%"); } </pre>	2
10	1	<pre> private static void SaveToFile(char[][] Field) { String Response = ""; String FileName = ""; Console.print("Save the current Field state to a text file? (Y/N):"); Response = Console.readLine(); if (Response.equals("Y")) { Console.print("Enter the File Name "); FileName = Console.readLine(); AQAWriteTextFile2017 CurrentFile = new </pre>	9

```
AQAWriteTextFile2017(FileName);
for (int Row = 0; Row < FIELDLENGTH; Row++)
{
    for (int Column = 0; Column < FIELDWIDTH; Column++)
    {
        CurrentFile.write(Field[Row][Column]);
    }
    CurrentFile.writeLine("|" + String.format("%3d", Row));
}
CurrentFile.closeFile();
}
}

private static void Simulation()
{
    int YearsToRun;
    char[][] Field = new char[FIELDLENGTH][FIELDWIDTH];
    Boolean Continuing;
    int Year;
    String Response;
    YearsToRun = GetHowLongToRun();
    if (YearsToRun != 0)
    {
        InitialiseField(Field);
        if (YearsToRun >= 1)
        {
            for (Year = 1; Year <= YearsToRun; Year++)
            {
                SimulateOneYear(Field, Year);
            }
        }
        else
        {
            Continuing = true;
            Year = 0;
            while (Continuing)
            {
                Year++;
                SimulateOneYear(Field, Year);
                Console.print("Press Enter to run simulation for
another Year, Input X to stop: ");
                Response = Console.readLine();
                if (Response.equals("x") || Response.equals("X"))
                {
                    Continuing = false;
                }
            }
        }
        Console.println("End of Simulation");
        SaveToFile(Field);
    }
}
```

		<pre> } Console.readLine(); } </pre>	
11	1	<pre> static void SimulateAutumn(char[][] Field) { String[] Direction = new String[] {"None", "East", "West", "North", "South", "Southeast", "Northeast", "Southwest", "Northwest"}; Random RNDWindDirection = new Random(); int PrevailingWind = RNDWindDirection.nextInt(9); String WindDirection = Direction[PrevailingWind]; int ColumnDisplacement = 0; int RowDisplacement = 0; if(WindDirection.equals("East")) { ColumnDisplacement = -1; } else if(WindDirection.equals("West")) { ColumnDisplacement = 1; } else if(WindDirection.equals("North")) { RowDisplacement = 1; } else if(WindDirection.equals("South")) { RowDisplacement = -1; } else if(WindDirection.equals("Southeast")) { ColumnDisplacement = -1; RowDisplacement = -1; } else if(WindDirection.equals("Northeast")) { ColumnDisplacement = -1; RowDisplacement = 1; } else if(WindDirection.equals("Southwest")) { ColumnDisplacement = 1; RowDisplacement = -1; } else if(WindDirection.equals("Northwest")) { ColumnDisplacement = 1; RowDisplacement = 1; } if(WindDirection.equals("None")) { </pre>	12

```
        Console.println("There was no wind this season");
    }
    else
    {
        Console.println("Prevailing wind: " + WindDirection);
    }
    for (int Row = 0; Row < FIELDLENGTH; Row++)
    {
        for (int Column = 0; Column < FIELDWIDTH; Column++)
        {
            if (Field[Row][Column] == PLANT)
            {
                SeedLands(Field, Row - 1 + RowDisplacement, Column -
1 + ColumnDisplacement);
                SeedLands(Field, Row - 1 + RowDisplacement, Column +
ColumnDisplacement);
                SeedLands(Field, Row - 1 + RowDisplacement, Column +
1 + ColumnDisplacement);
                SeedLands(Field, Row + RowDisplacement, Column - 1 +
ColumnDisplacement);
                SeedLands(Field, Row + RowDisplacement, Column + 1 +
ColumnDisplacement);
                SeedLands(Field, Row + 1 + RowDisplacement, Column -
1 + ColumnDisplacement);
                SeedLands(Field, Row + 1 + RowDisplacement, Column +
ColumnDisplacement);
                SeedLands(Field, Row + 1 + RowDisplacement, Column +
1 + ColumnDisplacement);
            }
        }
    }
}
```