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, i.e. 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.
### Question 01

<table>
<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>

**Marks**: 4

**01 mark** per two correct labels (round down).

**I. case**

**Note**: each label must only be used once (if given more than once, reject all occurrences).
### Mark Scheme - AS Computer Science Paper 1 - 7516/1 - June 2017

#### 02 1 All marks AO2 (apply)

<table>
<thead>
<tr>
<th>Qu</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<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>&quot;A2&quot;</td>
<td>0</td>
<td>&quot;A&quot;</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>&quot;2&quot;</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>&quot;1G&quot;</td>
<td>0</td>
<td>&quot;1&quot;</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>&quot;G&quot;</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td></td>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>

**Mark as follows:**

1. Count running over the values 1, 2 with correct sequence of values for HexString ("A2", "1G");
2. The correct sequence of values in Number column (0, 10, 162, 0, 1, 15);
3. The correct sequence of values in HexDigit column ("A", "2", "1", "G");
4. The correct sequence of values in Value column (10, 2, 1, -1);
5. The correct sequence of values in Output column (162, 15);

A. repeating values in first two columns
A. "1G" before "A2"
A. string values without quotes

#### 02 2 All marks for AO2 (analyse)

<table>
<thead>
<tr>
<th>Qu</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>2</td>
</tr>
</tbody>
</table>

1. invalid character produces value -1 from subroutine;
2. -1 should not be used to calculate // deal with -1 separately // using -1 gives a misleading result;
3. final output should be -1 / error message;

**MAX 2**
<table>
<thead>
<tr>
<th>03</th>
<th>1</th>
<th>All marks for AO3 (programming)</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mark as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. Correct prompts &quot;Enter a whole number: &quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Enter another whole number: &quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number1 and Number2 assigned values entered by user;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>R. if inside loop</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Number1 and Number2 assigned to Temp1 and Temp2 respectively;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. WHILE loop with syntax allowed by the programming language and correct condition for termination of the loop;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. Correct syntax and condition for the IF statement inside attempt at loop</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. Correct contents of THEN and ELSE part</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. Correct output &quot;... is GCF of ... and ...&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A. Temp1 instead of Result</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A. output on more than one line</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>R. if inside loop</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A. variations on prompts</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>I. minor differences in case and spelling</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DPT. If different identifiers</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>03</th>
<th>2</th>
<th>Mark is for AO3 (evaluate)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>**** SCREEN CAPTURE ****</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Must match code from 03.1, including prompts on screen capture matching those in code.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Code for 03.1 must be sensible.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Screen capture(s) showing the requested tests</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&gt;&gt;&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>enter a whole number: 12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>enter another whole number: 39</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 is GCF of 12 and 39</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;&lt;&lt;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>03</th>
<th>3</th>
<th>Mark is for AO2 (analyse)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>to preserve the original values for later use // otherwise output won’t make sense;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>Note: must refer to the fact that original values are needed later</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mark is for AO1 (understanding)</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>-------------------------------------------------------------------------------------------------------------------</td>
<td>---</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td><code>Frost // Continuing;</code></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>R.</strong> if any additional code</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>I.</strong> minor differences in case and spelling</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>R.</strong> significant differences in case and spelling</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td><code>GetHowLongToRun;</code></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>R.</strong> if any additional code</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>I.</strong> minor differences in case and spelling</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>R.</strong> significant differences in case and spelling</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td><code>Field;</code></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>R.</strong> <code>FieldRow</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>R.</strong> if any additional code</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>I.</strong> minor differences in case and spelling</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>R.</strong> significant differences in case and spelling</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td><code>Response // FileName // FieldRow;</code></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>R.</strong> if any additional code</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>I.</strong> minor differences in case and spelling</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>R.</strong> significant differences in case and spelling</td>
<td></td>
</tr>
</tbody>
</table>
05 1 All marks for AO2 (analyse)

1. (If the specification for the field size changes) only need to change the values at the beginning of the source code;
2. Makes the simulation/source code more understandable // improves readability of the code;
   A. easier to read
3. Makes clear that the values are the dimensions of the field // Identifiers convey meaning that the values directly don’t;
   A. Can’t change values accidentally;
   MAX 2

05 2 Mark is for AO2 (analyse)

It checks that the coordinates of the proposed seed position are within the field boundaries // not outside the bounds of the field;

NE. Validates seed position // stops generating an error

05 3 All marks for AO2 (analyse)

1. add another selection construct;
2. for rainfall equal to 1 (or 2);
3. when the plant count is exactly divisible by 4, change plant to soil;

A. any method that guarantees killing of 25% of the plants
R. random killing of plants

A. equivalent code

05 4 1 mark for AO1 (knowledge) and 1 mark for AO2 (apply)

Mark as follows:

AO1 (knowledge)

Integer division // Floor division // DIV;

A. Division that always rounds down to the next integer;

NE. division on its own

AO2 (apply)

Row = 10  Column = 17;  A. 10, 17 // 17,10
### AO2 (Analyse)

<table>
<thead>
<tr>
<th>Mark</th>
<th>Description</th>
</tr>
</thead>
</table>
| 06.1 | **InitialiseField**;  
I. minor differences in case and spelling  
R. if any additional code |
| 06.2 | **ReadFile**;  
A. **CreateNewField** if not given in 06.3;  
I. minor differences in case and spelling  
R. if any additional code |
| 06.3 | **CreateNewField**;  
A. **ReadFile** if not given in 06.2  
I. minor differences in case and spelling  
R. if any additional code |

### AO1 (Understanding)

<table>
<thead>
<tr>
<th>Mark</th>
<th>Description</th>
</tr>
</thead>
</table>
| 06.4 | **All marks for AO1 (understanding)**  
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;  
MAX 2 |
## All marks for AO2 (analyse)

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;

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**

## All marks for AO1 (understanding)

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;
08 1 1 mark for AO3 (design) and 5 marks for AO3 (programming)  5

Mark as follows:

AO3 (design) – 1 mark:

1. 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;
   A. recursive method instead of iterative statement

AO3 (programming) – 4 marks:

2. 'Invalid input' is displayed for any one invalid input; 
   R. if always displays error message
3. Function returns value for all valid inputs, and in no other circumstance;
4. test Year is in range -1 to 5; R. if zero excluded
5. test for non-integer input;
   A. test for one type of non-integer input (decimal or string)

I. minor differences in case and spelling

08 2 Mark is for AO3 (evaluate)  1

**** SCREEN CAPTURE ****
Must match code from 08.1, including prompts on screen capture matching those in code.
Code for 08.1 must be sensible.
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

A. different error message (or none) if it matches 08.1

```python
>>> 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

>>> .
```
<table>
<thead>
<tr>
<th>09</th>
<th>1</th>
<th>All marks for AO3 (programming)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. show correct formula for calculating percentage;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. show correct method for rounding result;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A. if it shows 15% (or 10%)</td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>2</td>
<td>Mark is for AO3 (evaluate)</td>
</tr>
</tbody>
</table>

**** SCREEN CAPTURE ****

*Must match code from 09.1, including prompts on screen capture matching those in code.*

*Code for 09.1 must be sensible.*

Screen captures showing the requested test being performed;

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%

- A. 0% if new field created and 09.1 correct
- A. truncated percentage

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: 1

```
......P............................|  0
.....................X.............|  1
...................................|  2
.........P.P.PXP.PP.PP.PP.......|  3
...................................|  4
.........P.P.............P.........|  5
.........P......
...................................|  6
.........P.P.PP.PP.PPX.P.P.........|  7
...........P...........P...........|  8
..X................................|  9
.........P.P.........P.P...........| 10
.........P.X.P.PP.PP...P........| 11
.........P...P.....P.P....X.....| 12
.........P.P...P.P...P.P...........| 13
.........P.P...P...P...P.P.........| 14
...............PP.PP...............| 15
.........P.P.........P.P...........| 16
```
3 marks for AO3 (design) and 6 marks for AO3 (programming)

**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.

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Mark Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>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 Field data into the subroutine. Most of the appropriate design decisions have been taken.</td>
<td>7-9</td>
</tr>
<tr>
<td>2</td>
<td>There is evidence that a line of reasoning has been partially followed. SaveToFile subroutine has been created, but it might only contain code for saving the data without formatting. There is evidence of some appropriate design work.</td>
<td>4-6</td>
</tr>
<tr>
<td>1</td>
<td>An attempt has been made to create SaveToFile 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.</td>
<td>1-3</td>
</tr>
</tbody>
</table>
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 $\text{SaveToFile}$
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 $\mathbf{R}$. 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 $\text{Simulation subroutine}$:
    Either: line above or below "End of Simulation"
    Or: after $\text{FOR}$ loop and after or within $\text{WHILE}$ loop

Refer answers using nested procedures to Team Leaders
<table>
<thead>
<tr>
<th>10</th>
<th>2</th>
<th>Mark is for AO3 (evaluate)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>**** SCREEN CAPTURE ****</th>
</tr>
</thead>
</table>

*Must match code from 10.1, including prompts on screen capture matching those in code.*

*Code for 10.1 must be sensible.*

*All screen captures must be present for mark to be awarded*

Screen captures showing the requested test being performed;

screen capture must show prompt to save and prompt for file name

```
Season:  winter   Year number:  2
        0
        1
        2
        3
        4
        5
        6
        7
      SSSS
        8
      SSSS
        9
      SSSS
       10
      SSSS
       11
      SSSS
       12
      SSSS
       13
      SSSS
       14
      SSSS
       15
      SSSS
       16
      SSSS
       17
      SSSS
       18
      SSSS
       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
```

A. different filename
Mark is for AO3 (evaluate)

**** SCREEN CAPTURE ****
*Must match screen capture from 10.2 (allow for a frost)*

*All screen captures must be present for mark to be awarded*

Screen captures showing the requested test being performed;

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
.................|  7
.................|  8
.................|  9
.................| 10
.................| 11
.................| 12
.................| 13
.................| 14
.................| 15
.................| 16
.................| 17
.................| 18
.................| 19
```

if no frost:

|..................................| 0 |
|..................................| 1 |
|..................................| 2 |
|..................................| 3 |
|..................................| 4 |
|..................................| 5 |
|..................................| 6 |
|..................................| 7 |
|.................PPPP.............| 8 |
|.................P..PP...........| 9 |
|.................P.P.P...........|10 |
|.................PP..P...........|11 |
|.................PPPP.............|12 |
|..................................|13 |
|..................................|14 |
|..................................|15 |
|..................................|16 |
|..................................|17 |
|..................................|18 |
|..................................|19 |
### 3 marks for AO3 (design) and 9 marks for AO3 (programming)

**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.

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Mark Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that is efficient. 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. All of the appropriate design decisions have been taken.</td>
<td>9-12</td>
</tr>
<tr>
<td>2</td>
<td>There is evidence that a line of reasoning has been partially followed. SimulateAutumn has been adapted, but it might only contain code for some of the wind directions and displacements have not been used correctly with SeedLands. There is evidence of some appropriate design work.</td>
<td>5-8</td>
</tr>
<tr>
<td>1</td>
<td>An attempt has been made to alter SimulateAutumn 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.</td>
<td>1-4</td>
</tr>
</tbody>
</table>
AO3 (design) – 3 points

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 SeedLands and deals with more than one wind direction

AO3 (programming) – 9 points

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

DPT. If direction of wind is interpreted as blowing towards instead of coming from

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 0 1 -2 -1 -1 -2 -1 0 0 0 -2 0 0 -2 -2 -2
-1 0 0 0 0 -2 0 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 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
**Mark is for AO3 (evaluate)**

**** SCREEN CAPTURE ****
*Must match code from 11.1, including prompts on screen capture matching those in code.*  
*Code for 11.1 must be sensible.*

Screen captures showing the requested test being performed;

**Two autumns** need to be shown

The screenshots should show the seeds relative to the plant as shown in the following (name of prevailing wind should match the displaced Seed positions relative to the Plant)

<table>
<thead>
<tr>
<th>Prevailing wind: No wind</th>
<th>Season: autumn</th>
<th>Year number: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>...................................</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>.SSS.</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>.S.P.S.</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>.SSS.</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prevailing wind: North</th>
<th>Season: autumn</th>
<th>Year number: 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>...................................</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>.S.P.S.</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>.S.S.</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>.S.S.</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>...................................</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>
Prevailing wind: **South**
Season: autumn  Year number:  1

<table>
<thead>
<tr>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
</tbody>
</table>

Prevailing wind: **East**
Season: autumn  Year number:  1

<table>
<thead>
<tr>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
</tbody>
</table>

Prevailing wind: **West**
Season: autumn  Year number:  1

<table>
<thead>
<tr>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
</tbody>
</table>

Prevailing wind: **NorthWest**
Season: autumn  Year number:  1

<table>
<thead>
<tr>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
</tbody>
</table>

Prevailing wind: **Southwest**
Season: autumn  Year number:  1

<table>
<thead>
<tr>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
</tbody>
</table>
Prevailing wind: **Northeast**  
Season: autumn  Year number: 1

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Prevailing wind: **Southeast**  
Season: autumn  Year number: 1

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

def GetHowLongToRun():
    print "Welcome to the Plant Growing Simulation"
    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

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 = int(round((NumberOfPlants * 100.0) / TotalCells))
    print Percentage, "%"
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(FIELDLENGTH):
            for Column in range(FIELDWIDTH):
                FileHandle.write(Field[Row][Column])
                FileHandle.write('|{0:>3}'.format(Row))
                FileHandle.write('
')
        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()

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
```python
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)
```

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

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, '%')
```
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('
')
        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()
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
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:
                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)
                Field = SeedLands(Field, Row + 1 + RowDisplacement, Column + ColumnDisplacement)
| \( \text{Field} = \text{SeedLands}(\text{Field}, \text{Row} + 1 + \text{RowDisplacement}, \text{Column} + \text{ColumnDisplacement}) \) |
| \( \text{Field} = \text{SeedLands}(\text{Field}, \text{Row} + 1 + \text{RowDisplacement}, \text{Column} + 1 + \text{ColumnDisplacement}) \) |
| return Field |
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

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

...
If Years >= -1 And Years <= 5 Then
Exit While
End If
Catch
End Try
Console.WriteLine("Invalid input")
End While
Return Years
...

09 1 ... 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) & "]")
...

10 1 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()
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.Write("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
    End If
End While
End Sub

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
        ColumnDisplacement = 1
    ElseIf WindDirection = "Northwest" Then
        RowDisplacement = -1
        ColumnDisplacement = 1
    End If
    Return Field(0, 0)
End Function
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)
            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
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.
```

```pascal
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
```

37 of 51
End;
If Not Valid Then
  Writeln('Invalid input');
End;
GetHowLongToRun := Years;
End;

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;

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
            Begin
              Write(FileHandle, Field[Row][Column]);
              Writeln(FileHandle, '|', Row:3);
            End;
        End;
      CloseFile(FileHandle);
    End;
  End;
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;
  End;
  Writeln('End of Simulation');
  SaveToFile(Field);
End;

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[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;
SimulateAutumn := Field;
End;
```csharp
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();
}

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());
        }
```
if (Years >= -1 && Years <= 5) {
    Valid = true;
}
} catch (Exception) {
}
if (!Valid) {
    Console.WriteLine("Invalid input");
}
return Years;

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) + ", ");
}

private static void SaveToFile(char[,] Field) {
    string Response = "", FileName = "";
    Console.Write("Save the current Field state to a text file? (Y/N):");
}
Response = Console.ReadLine();
if (Response == "Y")
{
    Console.Write("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.Write("Press Enter to run simulation for 
another Year, Input X to stop: ");
                Response = Console.ReadLine();
                if (Response == "x" || Response == "X")
                {
                    Continuing = false;
                }
            }
        }
    }
Console.WriteLine("End of Simulation");
    SaveToFile(Field);
}
Console.ReadLine();

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;
    }
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);
        }
    }
}
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);
}

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;
            }
        }
    }
}
catch(Exception e)
{
}
if (!Valid)
{
    Console.println("Invalid input");
}
return Years;

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.println("There is 1 plant growing");
    }
    else
    {
        Console.println("There are " + NumberOfPlants + " plants growing");
    }
    TotalCells = FIELDLENGTH * FIELDWIDTH;
    Percentage =
        ((double)NumberOfPlants/(double)TotalCells)*100.0;
    Console.writeLine(Math.round(Percentage) + "%");
}

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
```java
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)
            {
                Continuing = true;
                Year = 0;
                while (Continuing)
                {
                    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);
}
```
```java
} Console.readLine();
} static void SimulateAutumn(char[][] Field)
{
    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"))
    {
```
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);
    }
  }
}