AQA

AS
COMPUTER SCIENCE
7516/1

Paper 1

Mark scheme

June 2020

Version: 1.0 Final

206A7516/1/MS

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 Examiner.

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

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.

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 **14.1**, the marks available for the AO3 elements are as follows:

AO3 (design) – 3 marks
AO3 (programming) – 6 marks
Where a candidate's answer only reflects one element of the AO, the maximum mark they can receive will be restricted accordingly.

Section A

Qu		Marks	
01		**2 marks for AO1 (knowledge)** Problem definition; Requirements specification // list of objectives; Feedback about requirements specification from end user; Data model / ER diagram; Analysis data dictionary; Interviews; Questionnaires; Observations; Examination of documents; Research existing solutions; Acceptable limitations / constraints; **Max 2**	**2**

<table>
<tr><td>02</td><td>1</td><td>3 marks for AO2 (apply)

<table>
<tr><th>X</th><th>Result</th><th>Output</th></tr>
<tr><td>0</td><td>0</td><td>–</td></tr>
<tr><td>4</td><td>4</td><td></td></tr>
<tr><td>6</td><td>10</td><td></td></tr>
<tr><td>3</td><td>13</td><td></td></tr>
<tr><td>2</td><td>15</td><td></td></tr>
<tr><td>-1</td><td>14</td><td>14</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

1 mark for correct <code>X</code> column (4, 6, 3, 2, -1);
1 mark for correct <code>Result</code> column (4, 10, 13, 15);
1 mark for final <code>Result</code> value (14) and <code>Output</code> column (14);

Max 2 if any errors</td><td>3</td></tr>
<tr><td>02</td><td>2</td><td>2 marks for AO2 (analyse)

The result is wrong // The sentinel value should not have been used in the calculation;
Subtract the last value // input first value before the <code>WHILE</code> loop and swap the instructions within the <code>WHILE</code> loop // add 1 to result after loop is finished;

A. not add the value if it is the sentinel value</td><td>2</td></tr>
</table>

03	**1**	**9 marks for AO3 (programming)** **Mark as follows:** 1) Correct variable declarations for `X`, `Product`, `Factor`, **Note to examiners** If a language allows variables to be used without explicit declaration (eg Python) then this mark should be awarded if the correct variables exist in the program code and the first value they are assigned is of the correct data type. 2) Correct prompt "`Enter an integer greater than 1: `" and `X` assigned integer value entered by user; 3) Correct initialisation of `Product` and `Factor` before `WHILE` loop; 4) `WHILE` loop with syntax allowed by the programming language and correct condition for termination of the loop; 5) Correct incrementation of `Factor` and correct assignment to `Product` within `WHILE` loop; 6) `IF` statement with correct condition and `ELSE` part after the `WHILE` loop; 7) Correct re-initialisation of `Product` within `THEN` part; 8) `FOR` loop with syntax allowed by the programming language over correct range within `THEN` part; 9) Correct assignment to `Product` and output of `N` within `FOR` loop; **I.** minor differences in case and spelling **DPT.** use of incorrect variable name **Max 8** if code does not function correctly	**9**

03	**2**	**Mark is for AO3 (evaluate)**
		
		****** SCREEN CAPTURE ******
		Must match code from **03.1**, including prompts on screen capture matching those in code.
		Code for **03.1** must be sensible.
		
		Screen capture showing:
		'720' being entered and 1 2 3 4 5 6 displayed
		(Accept on same or separate lines)
		'600' being entered and the message 'No result' displayed
		
		`Enter an integer greater than 1: 720`
		`1`
		`2`
		`3`
		`4`
		`5`
		`6`
		`>>>`
		`Enter an integer greater than 1: 600`
		`No result`
			1

		`>>>`	
03	**3**	**Mark is for AO2 (analyse)** `X` is equal to the product of a sequence of (consecutive) whole numbers starting at 1 // `X` is a factorial number (greater than 1) // `X` is the factorial of a positive integer (greater than 1);	**1**

Section B

Qu		**Marks**	
04	**1**	**Mark is for AO1 (understand)** `FileFound` / `FileTypeOK` / `ProgramEnd`; **R.** if any additional code **R.** if spelt incorrectly **I.** case & spacing	**1**
04	**2**	**Mark is for AO1 (understand)** `ConvertChar` // `GetMenuOption` (Python only); **R.** if any additional code **R.** if spelt incorrectly **I.** case & spacing	**1**
04	**3**	**Mark is for AO1 (understand)** `EditImage`; **R.** if any additional code **R.** if spelt incorrectly **I.** case & spacing	**1**
05	**1**	**Mark is for AO1 (understand)** `Header` / `FileHeader`; **R.** if any additional code **R.** if spelt incorrectly **I.** case & spacing	**1**

05	**2**	**Mark is for AO1 (understand)** `Grid / Fields`; **R.** if any additional code **R.** if spelt incorrectly **I.** case & spacing	**1**
06	**1**	**Mark is for AO1 (understand)** It is easier to read / understand / more descriptive (the identifier) than ""; It will never change // a change in value would invalidate the name EMPTY_STRING, **R.** the value never changes <u>in this program</u>; If there was a requirement to change the representation of the empty string it would only need to be changed in one place; The value cannot be accidentally changed elsewhere in the code (**Do NOT accept** for Python); **A.** since the name is uppercase this tells the programmer not to alter it. (Python only) **Max 1**	**1**
06	**2**	**Mark is for AO1 (understand)** (These are the maximum dimensions and) may be changed easily if the program requirements change // a change would only need to be made in one place; It is easier to read/understand than the actual values; The value cannot be accidentally changed elsewhere in the code (**Do NOT accept** for Python); **A.** since the name is uppercase this tells the programmer not to alter it. (Python only) **R.** if answer is the same as **06.1** **Max 1**	**1**
07		**2 marks for AO2 (analyse)** `NextPixel` is not (a string that can be converted to) an integer; File no longer available; **R.** file could not be opened as already open Pixel data stored in wrong format, **R.** file does not have .txt extension; Not enough lines in the file **N.E.** Empty file**;** Not enough characters in a line; **A.** Not enough data in file for given size of image; if neither of the 2 points above have been given <u>Header</u> height larger than `MAX_HEIGHT`, **A.** larger than 100; <u>Header</u> width larger than `MAX_WIDTH`, **A.** larger than 100; **Max 2**	**2**

08	**1**	**Mark is for AO1 (understanding)** To represent the structure of the program // which subroutine is called from which subroutine; To aid decomposition of a problem; To aid with stepwise refinement; **Max 1**	**1**
08	**2**	**Mark is for AO1 (understanding)** A subroutine/procedure/function/method; **A.** module	**1**
08	**3**	**Mark is for AO2 (analyse)** `ConvertChar`; **R.** if any additional code **R.** if spelt incorrectly **I.** case & spacing	**1**
09	**1**	**Mark is for AO2 (analyse)** The default values of the file header; Empty string, `MAX_WIDTH`, `MAX_HEIGHT`, empty string; **A.** "" / '' / `EMPTY_STRING` for empty string **A.** 100 for `MAX_WIDTH` and/or `MAX_HEIGHT` **Max 1**	**1**
09	**2**	**3 marks for AO2 (analyse)** The string is split into separate parts; Delimited by a comma; Each part is assigned to a field of the header record; **A.** File type is used to determine which subroutine is called; **Max 3**	**3**

10		**2 marks for AO2 (analyse)**
		
		Each range of greyscale values is assigned a different ASCII character;
		A. Omission of range if an example range is given
		
		The lighter greyscales have less dense/smaller coverage
		// the darker greyscales have denser/larger coverage
		// the characters used give an effect appropriate for the greyscale value;
			2

11	**1**	**2 marks for AO2 (analyse)** When the image is displayed/loaded it will not be the intended image // the pixels will be misaligned; As the line breaks will now be in the wrong places // image width is shorter but length is longer; **A.** there will be no error message / exception; **Max 2**	**2**
11	**2**	**2 marks for AO2 (analyse)** The image is now a larger dimension; There will not be enough data in the file; The bottom rows of the image will consist of dots; The code will cause an exception // `"Error: Image data error"` will be displayed; **Max 2**	**2**
11	**3**	**2 marks for AO2 (analyse)** The lower part of the picture will not be displayed/loaded // only the upper part of the picture well be displayed/loaded; Only the top 10 rows are displayed/loaded // the bottom 49 rows are not displayed/loaded // only the cat's head is displayed/loaded // the cat's body has been removed; **Max 2**	**2**
12		**3 marks for AO2 (analyse)** 1) Each line/row of output across would consist of column pixels // Each column of output would consist of row pixels; 2) The bottom left corner of the image would now be top right // top right corner of the image would now be bottom left; 3) This would in effect be a flip/inversion; 4) across the diagonal; 5) … top left to bottom right // leading (diagonal); **Max 2** if overall effect is not clear. **Max 3**	**3**

Section C

Qu		Marks	
13	**1**	**1 mark for AO3 (design) and 5 marks for AO3 (programming)** **Mark as follows:** **AO3 (design) – 1 mark:** 1) Declare a new grid to receive mirror image; **AO3 (programming) – 5 marks:** 2) Create subroutine header with required parameters, **I.** extra parameters; 3) Column reference adjusted for mirror image; 4) Nested loops with correct ranges; 5) Add menu option in `DisplayMenu`; 6) Add call to `MirrorImage` in suitable place with parameters that match subroutine definition in code, **A**. call to `MirrorImage` in suitable place with `grid` and `header` parameters if subroutine definition not provided; **Max 5** if code does not function correctly.	**6**

13	**2**	**Mark is for AO3 (evaluate)**
		
		****** SCREEN CAPTURE ******
		Must match code from **13.1**, including prompts on screen capture matching those in code.
		Code for **13.1** must be sensible.
		
		Screen capture showing:
		
		`Main Menu`
		`=========`
		`L - Load graphics file`
		`D - Display image`
		`E - Edit image`
		`S - Save image`
		`M - Mirror image`
		`X - Exit program`
		
		`Enter your choice: M`
		
		`Cat`
		`===`
		
		`,/((,`
		`*/*,(// ,,,.  .#(/(`
		`./(((//////.,//(/#,*(`
		`#///////////////(*(/`
		`,,//////////////////`
		`#///////##//////////(`
		`*,.*////(#////((#///(`
		`,.  #//////////////////#`
		`./  .,*//#/(//(,,#/(///*`
		`/,*  .((*///#///(#///%(///#/#`
		`//  (///*,,,/////////////////(`
		`(/#  /,,,/////,,/////////##.`
		`(/(  ///,.,.,*////////////////(`
		`,.,*  (,/////////*,//////////////(`
		`.///.  ,.,,..,//////////////////////#`
		`,,,(  #///*****//////////////////////(`
		`(//*,  (/////////////,..,,(///////////`
		`(,,*(  #///////*..,,,,,,,,,#(.,,,,(//(`
		`,//*,,  (//////////////////(,.,,,(//#`
		`(,.,/#.  .#////////*,,,,,,,,.##////(//`
		`////,.*//*,,////////////////**/#.,,,(#/`
		`,#.,////,,/////////*,,,,,*/(/#////#,*`
		`#////////////#////,,,*#//#`
		`/#////////#//((((/////#/(`
			1

14	**1**	**3 marks for AO3 (design) and 6 marks for AO3 (programming)**	**9**

Level	Description	Mark Range
3	A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution. All of the appropriate design decisions have been taken. The hidden message may not have been built entirely correctly.	7–9
2	There is evidence that a line of reasoning has been partially followed. There is evidence of some appropriate design work. The subroutine `LoadGreyScaleImage` has been amended with a call to `FindSecretChar` in an appropriate place.	4–6
1	An attempt has been made to write the subroutine `FindSecretChar`. Some appropriate programming statements have been written. There is little 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 subroutines will have very little or none of the extra required functionality. It is unlikely that any of the key design elements of the task have been recognised.	1–3

Marking guidance:

Evidence of AO3 design – 3 marks:

Evidence of design to look for in response:

1) check whether value of pixel is in the correct range
2) convert a range of integers to a range of letters
3) call `FindSecretChar` with `PixelValue` and `Key` as parameters.

Note: AO3 (design) points 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 and regardless of whether the solution works.

Evidence of AO3 programming – 6 marks:

Evidence of programming to look for in response:

4) correct subroutine header and parameters for `FindSecretChar`, **I.** return type
5) generate underscore if no decrypted character found // generate space if `PixelValue - Key` is zero
6) always returns the correct character
7) extract the key from the file header
8) concatenate hidden message and returned character within `FOR` loop
9) output the hidden message after `FOR` loop.

Max 8 if code does not function correctly.

14	**2**	**Mark is for AO3 (evaluate)**
		
		****** SCREEN CAPTURE ******
		Must match code from **14.1**, including prompts on screen capture matching those in code.
		Code for **14.1** must be sensible.
		
		Screen capture showing:
		
		`Enter your choice: L`
		`Enter filename to load: greyscale`
		`H__E___L___P ME`
		
		`TestImage2`
		`==========`
		`##&#.`
		`&:#+&`
		` ####`
		
		A. hyphen instead of underscore character
			1

15	**1**	**3 marks for AO3 (design) and 9 marks for AO3 (programming)**	**12**

Level	Description	Mark Range
3	A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution. All of the appropriate design decisions have been taken. The last value pair may not have been saved.	9–12
2	There is evidence that a line of reasoning has been partially followed. There is evidence of some appropriate design work. Consecutive pixels are counted and most value pairs saved to a new file, delimiter may be missing.	5–8
1	An attempt has been made to write the subroutine `CompressFile`. Some appropriate programming statements have been written. There is little 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 subroutines will have very little or none of the extra required functionality. It is unlikely that any of the key design elements of the task have been recognised. Some appropriate programming statements from the `LoadFile` subroutine may have been used for reading a file.	1–4

Marking guidance:

Evidence of AO3 design – 3 points:

Evidence of design to look for in response:

1) Attempts to create new file with modified file name
2) Structure that compares current character with previous character
3) Under some circumstances counts consecutive symbols correctly

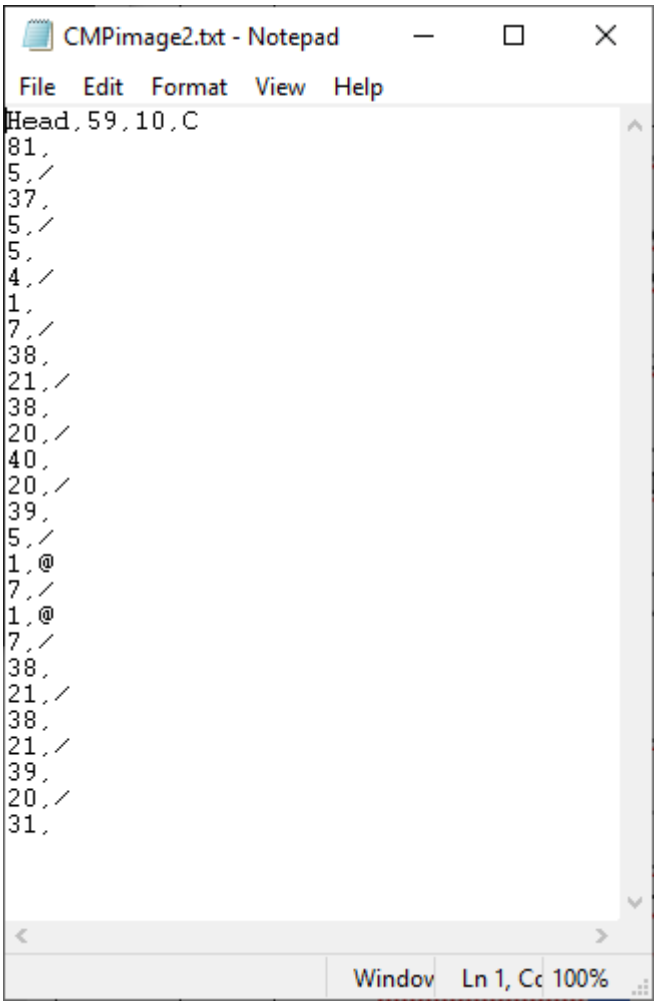
Note: AO3 (design) points are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.

Evidence of AO3 programming – 9 points:

Evidence of programming to look for in response:

4) Ask user for file name
5) Open existing file for reading and new file for writing
6) Edit file header correctly
7) Initialise symbol count for first run of symbols
8) Check each pixel in the file
9) Save first symbol count and symbol to file
10) Save each symbol count and symbol to file (except first and last pairs)

		11) Save last symbol count and symbol to file 12) Reset symbol count for next run of symbols **Max 11** if code does not function correctly.	

15	**2**	**Mark is for AO3 (evaluate)**
		
		****** SCREEN CAPTURE ******
		Must match code from **15.1**, including prompts on screen capture matching those in code.
		Code for **15.1** must be sensible.
		
		Screen capture showing:
		
		`Enter your choice: C`
		`Which graphics file do you want to compress? image2`
		
		 
		
			1

VB.Net

03	**1**	`Dim X As Integer '`
		`Dim Product As Integer '`
		`Dim Factor As Integer '1`
		`Console.Write("Enter an integer greater than 1: ") '`
		`X = Console.ReadLine '2`
		`Product = 1 '`
		`Factor = 0 '3`
		`While Product < X '4`
		`  Factor += 1 '`
		`  Product *= Factor '5`
		`End While`
		`If X = Product Then '6`
		`  Product = 1 '7`
		`  For N = 1 To Factor '8`
		`  Product = Product * N '`
		`  Console.WriteLine(N) '9`
		`  Next`
		`Else`
		`  Console.WriteLine("No result")`
		`End If`
			9

13	**1**	**`Sub MirrorImage(ByVal Grid(,) As String, ByVal Header As FileHeader) '2`**
		`  Dim NewGrid(Header.Height - 1, Header.Width - 1) As String '1`
		`  For Row = 0 To Header.Height - 1 '`
		`  For Column = 0 To Header.Width - 1 '4`
		`  NewGrid(Row, Column) = Grid(Row, Header.Width - 1 - Column) '3`
		`  Next`
		`  Next`
		`  DisplayImage(NewGrid, Header)`
		`End Sub`
		
		`Sub DisplayMenu()`
		`  Console.WriteLine()`
		`  Console.WriteLine("Main Menu")`
		`  Console.WriteLine("=========")`
		`  Console.WriteLine("L - Load graphics file")`
		`  Console.WriteLine("D - Display image")`
		`  Console.WriteLine("E - Edit image")`
		`  Console.WriteLine("S - Save image")`
		`  Console.WriteLine("X - Exit program")`
		`  Console.WriteLine("M - Mirror image") '5`
		`End Sub`
		
		`Sub Graphics()`
		`  Dim MenuOption As Char`
		`  Dim Grid(MAX_WIDTH - 1, MAX_HEIGHT - 1) As String`
		`  ClearGrid(Grid)`
		`  Dim Header As FileHeader`
		`  SetHeader(Header)`
		`  Dim ProgramEnd As Boolean = False`
		`  While Not ProgramEnd`
		`  DisplayMenu()`
		`  MenuOption = GetMenuOption()`
			6

```
    If MenuOption = "L" Then
      LoadFile(Grid, Header)
    ElseIf MenuOption = "D" Then
      DisplayImage(Grid, Header)
    ElseIf MenuOption = "E" Then
      Grid = EditImage(Grid, Header)
    ElseIf MenuOption = "S" Then
      SaveImage(Grid, Header)
    ElseIf MenuOption = "X" Then
      ProgramEnd = True
    ElseIf MenuOption = "M" Then
      MirrorImage(Grid, Header)  '6
    Else
      Console.WriteLine("You did not choose a valid menu option.
Try again")
    End If
  End While
  Console.WriteLine("You have chosen to exit the program")
  Console.Write("Do you want to save the image as a graphics
file? (Y/N) ")
  Dim Answer As String = Console.ReadLine()
  If Answer = "Y" Or Answer = "y" Then
    SaveFile(Grid, Header)
  End If
End Sub
```

Alternative answer for `MirrorImage` subroutine

```
Sub MirrorImage(ByVal Grid(,) As String, ByVal Header As
FileHeader) '2
  For Row = 0 To Header.Height - 1       '
    For Column = 0 To Header.Width - 1  '4
      Console.Write(Grid(Row, Header.Width - 1 - Column))  ' 3
    Next
    Console.WriteLine()
  Next
End Sub
```

14 1

```
Function FindSecretChar(ByVal PixelValue As Integer, ByVal Key
As Integer) As Char  '4
  Dim NewValue As Integer = PixelValue - Key
  If NewValue = 0 Then
    Return " "  '6
  ElseIf NewValue >= 1 And NewValue <= 26 Then  '1
    Return Chr(NewValue + Asc("A") - 1)  '2
  Else
    Return "_"  '5
  End If
End Function

Sub LoadGreyScaleImage(ByVal FileIn As StreamReader, ByVal
Grid(,) As String, ByVal Header As FileHeader)
```

9

		`  Try`
		`    Dim NextPixel As String`
		`    Dim PixelValue As Integer`
		`    Dim HiddenMessage As String = ""`
		`    Dim Key As Integer = CInt(Header.Title.Substring(Header.Title.Length - 1, 1))  '7`
		`    For Row = 0 To Header.Height - 1`
		`      For Column = 0 To Header.Width - 1`
		`        NextPixel = FileIn.ReadLine()`
		`        PixelValue = CInt(NextPixel)`
		`        HiddenMessage += FindSecretChar(PixelValue, Key)  '3, 8`
		`        Grid(Row, Column) = ConvertChar(PixelValue)`
		`      Next`
		`    Next`
		`    Console.WriteLine(HiddenMessage)  '9`
		`  Catch`
		`    DisplayError("Image data error")`
		`  End Try`
		`End Sub`	

15	**1**	`Sub CompressFile()`
		`  Console.Write("Enter the filename containing the data to compress: ")`
		`  Dim FileName As String = Console.ReadLine  '4`
		`  Dim FileIn As StreamReader = New StreamReader(FileName)  '5`
		`  Dim HeaderLine As String = FileIn.ReadLine()`
		`  Dim Fields() As String = HeaderLine.Split(",")`
		`  Dim Header As FileHeader`
		`  Header.Title = Fields(0)`
		`  Header.Width = CInt(Fields(1))`
		`  Header.Height = CInt(Fields(2))`
		`  Header.FileType = "C"  '6`
		`  Dim FileData As String = FileIn.ReadLine()`
		`  FileIn.Close()`
		`  Dim FileOut As StreamWriter = New StreamWriter("CMP" & FileName)  '1, 5`
		`  FileOut.WriteLine(Header.Title + "," + CStr(Header.Width) + "," + CStr(Header.Height) + "," + Header.FileType)  '2`
		`  Dim Count As Integer = 1  '7`
		`  Dim LastChar As Char = FileData(0)`
		`  For Pos = 1 To FileData.Length - 1  '8`
		`    If FileData(Pos) = LastChar Then  '2`
		`      Count += 1  '3`
		`    Else`
		`      FileOut.WriteLine(CStr(Count) + "," + LastChar)  '9, 10`
		`      Count = 1  '12`
		`      LastChar = FileData(Pos)`
		`    End If`
		`  Next`
		`  FileOut.WriteLine(CStr(Count) + "," + LastChar)  '11`
		`  FileOut.Close()`
		`End Sub`
			12

Python 3

03	**1**	`# X, Product, Factor are integers # 1`
		`print("Enter an integer greater than 1: ") #`
		`X = int(input())  # 2`
		`Product = 1 #`
		`Factor = 0  # 3`
		`while Product < X: # 4`
		`  Factor += 1  #`
		`  Product = Product * Factor # 5`
		`if X == Product: # 6`
		`  Product = 1 # 7`
		`  for N in range(1, Factor + 1): # 8`
		`  Product = Product * N #`
		`  print(N)  # 9`
		`else:`
		`  print("No result")`
			9

13	**1**	**`def MirrorImage(Grid, Header): # 2`**
		`  NewGrid = [[EMPTY_STRING for Column in range(MAX_WIDTH)] for Row in range(MAX_HEIGHT)]`
		`  NewGrid = ClearGrid(NewGrid) # 1`
		`  for ThisRow in range(Header.Height):  #`
		`  for NewColumn in range(Header.Width): # 4`
		`  NewGrid[ThisRow][NewColumn] = Grid[ThisRow][Header.Width - NewColumn - 1] # 3`
		`  DisplayImage(NewGrid, Header)`
		
		`def DisplayMenu():`
		`  print()`
		`  print("Main Menu")`
		`  print("=========")`
		`  print("L - Load graphics file")`
		`  print("D - Display image")`
		`  print("E - Edit image")`
		`  print("S - Save image")`
		`  print("M - Mirror image")  # 5`
		`  print("X - Exit program")`
		`  print()`
		
		`def Graphics():`
		`  Grid = [['' for Column in range(MAX_WIDTH)] for Row in range(MAX_HEIGHT)]`
		`  Grid = ClearGrid(Grid)`
		`  Header = FileHeader()`
		`  ProgramEnd = False`
		`  while not ProgramEnd:`
		`  DisplayMenu()`
		`  MenuOption = GetMenuOption()`
		`  if MenuOption == 'L':`
		`  Grid, Header = LoadFile(Grid, Header)`
		`  elif MenuOption == 'D':`
		`  DisplayImage(Grid, Header)`
		`  elif MenuOption == 'E':`
		`  Grid = EditImage(Grid, Header)`
		`  elif MenuOption == 'S':`
		`  SaveImage(Grid, Header)`
			6

		`elif MenuOption == 'M':`
		`  Grid = MirrorImage(Grid, Header) # 6`
		`elif MenuOption == 'X':`
		`  ProgramEnd = True`
		`else:`
		`  print("You did not choose a valid menu option. Try again")`
		`print("You have chosen to exit the program")`
		`Answer = input("Do you want to save the image as a graphics file? (Y/N) ")`
		`if Answer == "Y" or Answer == "y":`
		`  SaveFile(Grid, Header)`	

14	**1**	`def FindSecretChar(PixelValue, Key): # 4`
		`  Character = '_'  # 5`
		`  NewValue = PixelValue - Key`
		`  if NewValue == 0:`
		`  Character = ' ' # 6`
		`  elif NewValue in range(1, 27): # 1`
		`  Character = chr(ord('A') + NewValue - 1) # 2`
		`  return Character`
		
		`def LoadGreyScaleImage(FileIn, Grid, Header):`
		`  try:`
		`  Key = int(Header.Title[-1]) # 7`
		`  HiddenMessage = EMPTY_STRING`
		`  for Row in range(Header.Height):`
		`  for Column in range(Header.Width):`
		`  NextPixel = FileIn.readline()`
		`  PixelValue = int(NextPixel)`
		`  HiddenMessage = HiddenMessage + FindSecretChar(PixelValue, Key) # 3, 8`
		`  Grid[Row][Column] = ConvertChar(PixelValue)`
		`  print(HiddenMessage) # 9`
		`  except:`
		`  DisplayError("Image data error")`
		`  return Grid`
			9

15	**1**	`def CompressFile():`
		`  FileFound = False`
		`  Header = FileHeader()`
		`  FileName = input("Which graphics file do you want to compress? ") # 4`
		`  try:`
		`  FileIn = open(FileName + ".txt", 'r')`
		`  FileFound = True`
		`  FileOut = open("CMP" + FileName + ".txt", 'w') # 1, 5`
		`  HeaderLine = FileIn.readline()`
		`  HeaderLine = HeaderLine[0:-2] + "C" # 6`
		`  FileOut.write(HeaderLine + "\n")  # 2`
		`  ImageData = FileIn.readline()`
		`  PrevPixelChar = ImageData[0]`
		`  PixelCount = 0 # 7`
		`  for NextPixelChar in ImageData: # 8`
		`  if NextPixelChar == PrevPixelChar: # 2`
			12

```
        PixelCount += 1                    # 3
      else:
        FileOut.write(str(PixelCount) + "," + PrevPixelChar + "\n")
# 9, 10
        PrevPixelChar = NextPixelChar
        PixelCount = 1 # 12
    FileOut.write(str(PixelCount) + "," + PrevPixelChar + "\n") #
11
    FileOut.close()
    FileIn.close()
  except:
    if not FileFound:
      DisplayError("File not found")
    else:
      DisplayError("Error during compression")
```

Python 2

03	**1**	`# X, Product, Factor are integers # 1`
		`print "Enter an integer greater than 1: "  #`
		`X = int(raw_input())  # 2`
		`Product = 1 #`
		`Factor = 0  # 3`
		`while Product < X: # 4`
		`  Factor += 1  #`
		`  Product = Product * Factor # 5`
		`if X == Product: # 6`
		`  Product = 1 # 7`
		`  for N in range(1, Factor + 1): # 8`
		`  Product = Product * N #`
		`  print N  # 9`
		`else:`
		`  print "No result"`
			9

13	**1**	`def MirrorImage(Grid, Header): # 2`
		`  NewGrid = [[EMPTY_STRING for Column in range(MAX_WIDTH)] for Row in range(MAX_HEIGHT)]`
		`  NewGrid = ClearGrid(NewGrid) # 1`
		`  for ThisRow in range(Header.Height):`
		`  for NewColumn in range(Header.Width): # 4`
		`  NewGrid[ThisRow][NewColumn] = Grid[ThisRow][Header.Width - NewColumn - 1] # 3`
		`  DisplayImage(NewGrid, Header)`
		
		`def DisplayMenu():`
		`  print`
		`  print "Main Menu"`
		`  print "========="`
		`  print "L - Load graphics file"`
		`  print "D - Display image"`
		`  print "E - Edit image"`
		`  print "S - Save image"`
		`  print "M - Mirror image"  # 5`
		`  print "X - Exit program"`
		`  print`
		
		`def Graphics():`
		`  Grid = [['' for Column in range(MAX_WIDTH)] for Row in range(MAX_HEIGHT)]`
		`  Grid = ClearGrid(Grid)`
		`  Header = FileHeader()`
		`  ProgramEnd = False`
		`  while not ProgramEnd:`
		`  DisplayMenu()`
		`  MenuOption = GetMenuOption()`
		`  if MenuOption == 'L':`
		`  Grid, Header = LoadFile(Grid, Header)`
		`  elif MenuOption == 'D':`
		`  DisplayImage(Grid, Header)`
		`  elif MenuOption == 'E':`
		`  Grid = EditImage(Grid, Header)`
		`  elif MenuOption == 'S':`
		`  SaveImage(Grid, Header)`
			6

		`elif MenuOption == 'M':`
		`  Grid = MirrorImage(Grid, Header) # 6`
		`elif MenuOption == 'X':`
		`  ProgramEnd = True`
		`else:`
		`  print "You did not choose a valid menu option. Try again"`
		`print "You have chosen to exit the program"`
		`Answer = raw_input("Do you want to save the image as a graphics file? (Y/N) ")`
		`if Answer == "Y" or Answer == "y":`
		`  SaveFile(Grid, Header)`	

14	1	`def FindSecretChar(PixelValue, Key): # 4`
		`  Character = '_'  # 5`
		`  NewValue = PixelValue - Key`
		`  if NewValue == 0:`
		`  Character = ' ' # 6`
		`  elif NewValue in range(1, 27): # 1`
		`  Character = chr(ord('A') + NewValue - 1) # 2`
		`  return Character`
		
		`def LoadGreyScaleImage(FileIn, Grid, Header):`
		`  try:`
		`  Key = int(Header.Title[-1]) # 7`
		`  HiddenMessage = EMPTY_STRING`
		`  for Row in range(Header.Height):`
		`  for Column in range(Header.Width):`
		`  NextPixel = FileIn.readline()`
		`  PixelValue = int(NextPixel)`
		`  HiddenMessage = HiddenMessage + FindSecretChar(PixelValue, Key) # 3, 8`
		`  Grid[Row][Column] = ConvertChar(PixelValue)`
		`  print HiddenMessage # 9`
		`  except:`
		`  DisplayError("Image data error")`
		`  return Grid`
			9

15	1	`def CompressFile():`
		`  FileFound = False`
		`  Header = FileHeader()`
		`  FileName = raw_input("Which graphics file do you want to compress? ") # 4`
		`  try:`
		`  FileIn = open(FileName + ".txt", 'r')`
		`  FileFound = True`
		`  FileOut = open("CMP" + FileName + ".txt", 'w') # 1, 5`
		`  HeaderLine = FileIn.readline()`
		`  HeaderLine = HeaderLine[0:-2] + "C" # 6`
		`  FileOut.write(HeaderLine + "\n")  # 2`
		`  ImageData = FileIn.readline()`
		`  PrevPixelChar = ImageData[0]`
		`  PixelCount = 0 # 7`
		`  for NextPixelChar in ImageData: # 8`
		`  if NextPixelChar == PrevPixelChar: # 2`
			12

```
        PixelCount += 1                    # 3
      else:
        FileOut.write(str(PixelCount) + "," + PrevPixelChar + "\n")
# 9, 10
        PrevPixelChar = NextPixelChar
        PixelCount = 1 # 12
    FileOut.write(str(PixelCount) + "," + PrevPixelChar + "\n") #
11
    FileOut.close()
    FileIn.close()
  except:
    if not FileFound:
      DisplayError("File not found")
    else:
      DisplayError("Error during compression")
```

Pascal

03	**1**	`program question3;`
		`{$APPtype CONSOLE}`
		`uses SysUtils;`
		`var`
		`  X, Product, Factor, N: integer; // 1`
		`begin`
		`  write('Enter an integer greater than 1: ');  //`
		`  readln(X);  // 2`
		`  Product := 1; //`
		`  Factor := 0;  // 3`
		`  while Product < X do // 4`
		`  begin`
		`  Factor := Factor + 1;  //`
		`  Product := Product * Factor; // 5`
		`  end;`
		`  if X = Product // 6`
		`  then`
		`  begin`
		`  Product := 1;  // 7`
		`  for N := 1 to Factor do // 8`
		`  begin`
		`  Product := Product * N;  //`
		`  writeln(N);  // 9`
		`  end;`
		`  end`
		`  else`
		`  writeln('No result');`
		`  readln;`
		`end.`
			9

13	**1**	**`procedure MirrorImage(var Grid: TGrid; Header: FileHeader);`**
		`// 2`
		`var`
		`  NewGrid: TGrid;`
		`  ThisRow, NewColumn: integer;`
		`begin`
		`  ClearGrid(NewGrid);` `// 1`
		`  for ThisRow := 0 to Header.Height - 1 do`
		`  for NewColumn := 0 to Header.Width - 1 do` `// 4`
		`  NewGrid[ThisRow, NewColumn] := Grid[ThisRow,`
		`Header.Width - NewColumn - 1];` `// 3`
		`  DisplayImage(NewGrid, Header) ;`
		`  Grid := NewGrid;`
		`end;`
		
		`procedure DisplayMenu();`
		`begin`
		`  writeln;`
		`  writeln('Main Menu');`
		`  writeln('=========');`
		`  writeln('L - Load graphics file');`
		`  writeln('D - Display image');`
			6

		`  writeln('E - Edit image');`
		`  writeln('S - Save image');`
		`  **writeln('M - Mirror image');** // 5`
		`  writeln('X - Exit program');`
		`  writeln;`
		`end;`
		
		`procedure Graphics();`
		`var`
		`  Grid: TGrid;`
		`  Header: FileHeader;`
		`  ProgramEnd: boolean;`
		`  MenuOption: char;`
		`  Answer: char;`
		`begin`
		`  ClearGrid(Grid);`
		`  SetHeader(Header);`
		`  ProgramEnd := false;`
		`  while not ProgramEnd do`
		`  begin`
		`  DisplayMenu();`
		`  MenuOption := GetMenuOption();`
		`  case MenuOption of`
		`  'L': LoadFile(Grid, Header);`
		`  'D': DisplayImage(Grid, Header);`
		`  'E': EditImage(Grid, Header);`
		`  'S': SaveImage(Grid, Header);`
		`  **'M': MirrorImage(Grid, Header);** // 6`
		`  'X': ProgramEnd := true;`
		`  else`
		`  writeln('You did not choose a valid menu option. Try again');`
		`  end;`
		`  end;`
		`  writeln('You have chosen to exit the program');`
		`  write('Do you want to save the image as a graphics file? (Y/N) ');`
		`  readln(Answer);`
		`  if (Answer = 'Y') or (Answer = 'y')`
		`  then`
		`  SaveFile(Grid, Header);`
		`  readln;`
		`end;`	

14	**1**	**`function FindSecretChar(PixelValue, Key: integer): char;`** // 4 **`var`** **`  Character: char;`** **`  NewValue: integer;`** **`begin`** **`  Character := '_';`** // 5 **`  NewValue:= PixelValue - Key;`** **`  if NewValue = 0`** **`  then`**	**9**

		`Character := ' ' // 6`
		`else`
		`if (NewValue >= 1) And (NewValue <= 26) // 1`
		`then`
		`Character := chr(ord('A') + NewValue - 1); // 2`
		`FindSecretChar := Character;`
		`end;`
		
		`procedure LoadGreyScaleImage(var FileIn: text; var Grid: TGrid; var Header: FileHeader);`
		`var`
		`Row, Column: integer;`
		`NextPixel, HiddenMessage: string;`
		`PixelValue, Key: integer;`
		`Begin`
		`try`
		`Key := strToInt(rightStr(Header.Title, 1)); // 7`
		`HiddenMessage := EMPTY_STRING;`
		`for Row := 0 to Header.Height - 1 do`
		`for Column := 0 to Header.Width - 1 do`
		`begin`
		`readln(FileIn, NextPixel);`
		`PixelValue := strToInt(NextPixel);`
		`HiddenMessage := HiddenMessage + FindSecretChar(PixelValue, Key); // 3, 8`
		`Grid[Row, Column] := ConvertChar(PixelValue);`
		`end;`
		`writeln(HiddenMessage); // 9`
		`except`
		`DisplayError('Image data error');`
		`end;`
		`end;`	

15	**1**	`procedure CompressFile();`
		`var`
		`FileFound: boolean;`
		`Header: FileHeader;`
		`FileIn, FileOut: text;`
		`FileName, HeaderLine: string;`
		`Fields: array[0 .. 4] of string;`
		`i, PixelCount: integer;`
		`PrevPixelChar, NextPixelChar: char;`
		`begin`
		`FileFound := false;`
		`write('Which graphics file do you want to compress? ');`
		`readln(FileName); // 4`
		`try`
		`assignFile(FileIn, FileName + '.txt');`
		`reset(FileIn);`
		`FileFound := true;`
			12

```
    assignFile(FileOut, 'CMP' + FileName); // 1, 5
    rewrite(FileOut);
    readln(FileIn, HeaderLine);
    HeaderLine := leftStr(HeaderLine, length(HeaderLine) - 1)
+ 'C';  // 6
  writeln(FileOut, HeaderLine);                          // 2
  PixelCount := 1;  // 7
  read(FileIn, PrevPixelChar);
  while not eoln(FileIn) do  // 8
  begin
    read(FileIn, NextPixelChar);
    if NextPixelChar = PrevPixelChar // 2
      then
        PixelCount := PixelCount + 1 // 3
      else
        begin
          writeln(FileOut, PixelCount, ',', PrevPixelChar);
// 9, 10
          PrevPixelChar := NextPixelChar;
          PixelCount := 1; // 12
        end;
  end;
  writeln(FileOut, PixelCount, ',', PrevPixelChar); // 11
  closeFile(FileOut);
  closeFile(FileIn);
  except
    if not FileFound
      then
        DisplayError('File not found')
      else
        DisplayError('Error during compression');
  end;
end;
```

C#

03 | **1** | **9**

```
int x, product, factor; // 1
Console.Write("Enter an integer greater than 1: ");   //
x = Convert.ToInt32(Console.ReadLine());              // 2
product = 1;   //
factor = 0;    // 3
while (product < x)   // 4
{
    factor++;                      //
    product = product * factor; // 5
}
if (x == product)   // 6
{
    product = 1;   // 7
    for (int n = 1; n < factor + 1; n++)   // 8
    {
        product = product * n;   //
        Console.WriteLine(n);    // 9
    }
}
else
{
    Console.WriteLine("No result");
}
Console.ReadLine();
```

13 | **1** | **6**

```
private static void MirrorImage(string[,] grid, FileHeader
header)
{   // 2
    string[,] newGrid = new string[MaxHeight, MaxWidth];   // 1
    ClearGrid(newGrid);
    for (int thisRow = 0; thisRow < header.Height; thisRow++)
//
    {
        for (int newColumn = 0; newColumn < header.Width;
newColumn++)
// 4
        {
            newGrid[thisRow, newColumn] = grid[thisRow,
header.Width - newColumn - 1];
// 3
        }
    }
    DisplayImage(newGrid, header);
}

private static void DisplayMenu()
{
    Console.WriteLine();
    Console.WriteLine("Main Menu");
    Console.WriteLine("=========");
    Console.WriteLine("L - Load graphics file");
    Console.WriteLine("D - Display image");
    Console.WriteLine("E - Edit image");
    Console.WriteLine("S - Save image");
    Console.WriteLine("M - Mirror image");   // 5
```

```
        Console.WriteLine("X - Exit program");
        Console.WriteLine();
    }

    private static void Graphics()
    {
        string[,] grid = new string[MaxHeight, MaxWidth];
        ClearGrid(grid);
        FileHeader header = new FileHeader();
        bool programEnd = false;
        char menuOption;
        char answer;
        while (!programEnd)
        {
            DisplayMenu();
            menuOption = GetMenuOption();
            if (menuOption == 'L')
            {
                LoadFile(grid, header);
            }
            else if (menuOption == 'D')
            {
                DisplayImage(grid, header);
            }
            else if (menuOption == 'E')
            {
                EditImage(grid, header);
            }
            else if (menuOption == 'S')
            {
                SaveImage(grid, header);
            }
            else if (menuOption == 'M')
            {
                MirrorImage(grid, header);   // 6
            }
            else if (menuOption == 'X')
            {
                programEnd = true;
            }
            else
            {
                Console.WriteLine("You did not choose a valid menu
option. Try again");
            }
        }
        Console.WriteLine("You have chosen to exit the program");
        Console.Write("Do you want to save the image as a graphics
file? (Y/N) ");
        answer = Convert.ToChar(Console.ReadLine());
        if (answer == 'Y' || answer == 'y')
        {
            SaveFile(grid, header);
        }
    }
```

14	1		9

```
private static char FindSecretChar(int pixelValue, int key)   // 4
{
    char character;
    int newValue;
    character = '_'; // 5
    newValue = pixelValue - key;
    if (newValue == 0)
    {
        character = ' ';  // 6
    }
    else
    {
        if (newValue >= 1 && newValue <= 26)  // 1
        {
            character = ((char)((int)'A' + newValue - 1));  // 2
        }
    }
    return character;
}

private static void LoadGreyScaleImage(StreamReader fileIn,
string[,] grid, FileHeader header)
{
    string nextPixel;
    int pixelValue;
    int key;
    string hiddenMessage;
    try
    {
        key = Convert.ToInt32(header.title[header.title.Length -
1].ToString());  // 7
        hiddenMessage = EMPTY_STRING;
        for (int row = 0; row < header.Height; row++)
        {
            for (int column = 0; column < header.Width;
column++)
            {
                nextPixel = fileIn.ReadLine();
                pixelValue = Convert.ToInt32(nextPixel);
                hiddenMessage = hiddenMessage +
FindSecretChar(pixelValue, key); // 3, 8
                grid[row, column] = ConvertChar(pixelValue);
            }
        }
        Console.WriteLine(hiddenMessage);  // 9
    }
    catch (Exception)
    {
```

		`DisplayError("Image data error");`
		`}`
		`}`	

15	**1**	`private static void CompressFile()`
		`{`
		`bool fileFound = false;`
		`string headerLine, imageData;`
		`char prevPixelChar;`
		`int pixelCount = 0;  // 7`
		`FileHeader header = new FileHeader();`
		`Console.Write("Which graphics file do you want to compress? ");`
		`string fileName = Console.ReadLine();  // 4`
		`try`
		`{`
		`StreamReader filein = new StreamReader(fileName + ".txt");`
		`fileFound = true;  // 5`
		`StreamWriter fileOut = new StreamWriter("CMP" + fileName + ".txt");  // 1, 5`
		`headerLine = filein.ReadLine();`
		`headerLine = headerLine.Substring(0, headerLine.Length - 1) + "C";  // 6`
		`fileOut.WriteLine(headerLine);`
		`imageData = filein.ReadLine();`
		`prevPixelChar = imageData[0];`
		`foreach (char nextPixelChar in imageData)  // 8`
		`{`
		`if (nextPixelChar == prevPixelChar)  // 2`
		`{`
		`pixelCount++;  // 3`
		`}`
		`else`
		`{`
		`fileOut.WriteLine(pixelCount + "," + prevPixelChar);  // 9, 10`
		`prevPixelChar = nextPixelChar;`
		`pixelCount = 1;  // 12`
		`}`
		`}`
		`fileOut.WriteLine(pixelCount + "," + prevPixelChar);  // 11`
		`fileOut.Close();`
		`filein.Close();`
		`}`
		`catch (Exception)`
		`{`
		
		`if (!fileFound)`
		`{`
		`DisplayError("File not found");`
		`}`
		`else`
		`{`
		`DisplayError("Error during compression");`
			12

		} **}** **}**	

Java

03	**1**	`Console.write("Enter an integer greater than 1: ");  //`
		`int x = Integer.parseInt(Console.readLine());  // 2`
		`int product = 1;  //`
		`int factor = 0;  // 3  // 1`
		`while (product < x) {  // 4`
		`  factor = factor + 1;  //`
		`  product = product * factor;  // 5`
		`}`
		`if (x == product) {  // 6`
		`  product = 1;  // 7`
		`  for (int n = 1; n <= factor; n++) {  // 8`
		`  product = product * n;  //`
		`  Console.writeLine(n);  // 9`
		`  }`
		`} else {`
		`  Console.writeLine("No result");`
		`}`
			9

13	**1**	`void mirrorImage(String[][] grid, FileHeader header) { // 2`
		`  String[][] newGrid = new String[MAX_HEIGHT][MAX_WIDTH];  //`
		`1`
		`  clearGrid(newGrid);`
		`  for (int row = 0; row < header.height; row++) {`
		`//`
		`  for (int column = 0; column < header.width; column++)`
		`{// 4`
		`  newGrid[row][column] = grid[row][header.width - 1 -`
		`column];  // 3`
		`  }`
		`  }`
		`  displayImage(newGrid, header);`
		`}`
		
		`void displayMenu() {`
		`  Console.writeLine();`
		`  Console.writeLine("Main Menu");`
		`  Console.writeLine("=========");`
		`  Console.writeLine("L - Load graphics file");`
		`  Console.writeLine("D - Display image");`
		`  Console.writeLine("M - Mirror image");  // 5`
		`  Console.writeLine("E - Edit image");`
		`  Console.writeLine("S - Save image");`
		`  Console.writeLine("X - Exit program");`
		`  Console.writeLine();`
		`}`
		`void graphics() {`
		`  String[][] grid = new String[MAX_HEIGHT][MAX_WIDTH];`
		`  clearGrid(grid);`
		`  FileHeader header = new FileHeader();`
		`  boolean programEnd = false;`
		`  while (!programEnd) {`
		`  displayMenu();`
		`  char menuOption = GetMenuOption();`
		`  if ( menuOption == 'L') {`
		`  loadFile(grid, header);`
			6

		`} else if ( menuOption == 'D') {`
		`displayImage(grid, header);`
		`} else if (menuOption == 'M') {`
		`mirrorImage(grid, header);  // 6`
		`} else if ( menuOption == 'E') {`
		`editImage(grid, header);`
		`} else if ( menuOption == 'S') {`
		`saveImage(grid, header);`
		`} else if ( menuOption == 'X') {`
		`programEnd = true;`
		`} else {`
		`Console.writeLine("You did not choose a valid menu option. Try again");`
		`}`
		`}`
		`Console.writeLine("You have chosen to exit the program");`
		`Console.write("Do you want to save the image as a graphics file? (Y/N) ");`
		`String answer = Console.readLine();`
		`if ( answer.equals("Y") || answer.equals("y")) {`
		`saveFile(grid, header);`
		`}`
		`}`	

14	**1**	`char findSecretChar(int pixelValue, int key) {  // 4`
		`char character = '_';  // 5`
		`int newValue = pixelValue - key;`
		`if (newValue == 0) {`
		`character = ' ';  // 6`
		`} else if (newValue >= 1 && newValue < 27) {  // 1`
		`character = (char)((int)('A') + newValue - 1);  // 2`
		`}`
		`return character;`
		`}`
		
		`void loadGreyScaleImage(BufferedReader fileIn, String[][] grid, FileHeader header) {`
		`try {`
		`int key = Integer.parseInt(header.title.charAt(header.title.length()-1) + "");  // 7`
		`String hiddenMessage = EMPTY_STRING;`
		`for (int row = 0; row < header.height; row++) {`
		`for (int column = 0; column < header.width; column++) {`
		`String nextPixel = fileIn.readLine();`
		`int pixelValue = Integer.parseInt(nextPixel);`
		`grid[row][column] = convertChar(pixelValue);`
		`hiddenMessage = hiddenMessage + findSecretChar(pixelValue, key);  // 3, 8`
		`}`
		`}`
		`Console.println(hiddenMessage);  // 9`
			9

		`} catch (Exception e) {`
		`displayError("Image data error");`
		`}`
		`}`	

15	**1**	`void compressFile ()`
		`{`
		`Console.write("Enter filename to compress: ");`
		`String fileName = Console.readLine();  // 4`
		`try {`
		`BufferedReader fileIn = new BufferedReader(new FileReader(fileName + ".txt"));  // 5`
		`BufferedWriter fileOut = new BufferedWriter(new FileWriter("CMP" + fileName + ".txt"));  // 1, 5`
		`String headerLine = fileIn.readLine();`
		`headerLine = headerLine.substring(0, headerLine.length()-1) + "C";  // 6`
		`fileOut.write(headerLine + "\n");`
		`String imageData = fileIn.readLine();`
		`char previousPixelChar = imageData.charAt(0);`
		`int pixelCount = 0;  // 7`
		`for (int pos = 0; pos < imageData.length(); pos++) {  // 8`
		`char nextPixelChar = imageData.charAt(pos);`
		`if (nextPixelChar == previousPixelChar) {  // 2`
		`pixelCount++;  // 3`
		`} else {`
		`fileOut.write(pixelCount + "," + previousPixelChar + "\n"); // 9, 10`
		`previousPixelChar = nextPixelChar;`
		`pixelCount = 1;  // 12`
		`}`
		`}`
		`fileOut.write(pixelCount + "," + previousPixelChar + "\n");`
		`// 11`
		`fileIn.close();`
		`fileOut.close();`
		`} catch (IOException e) {`
		`}`
		`}`
			12