AQA

GCSE
COMPUTER SCIENCE
8525/1A, 8525/1B, 8525/1C

Paper 1 Computational thinking and programming skills

Mark scheme

June 2025

Version: 1.0 Final

2 5 6 G 8 5 2 5 / 1 / M S

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.

This mark scheme contains the correct responses which we believe that candidates are most likely to give. Other valid responses are possible to some questions and should be credited. Examiners should refer responses that are not covered by the mark scheme, but which they deem creditworthy, to a Team Leader.

No student should be disadvantaged on the basis of their gender identity and/or how they refer to the gender identity of others in their exam responses.

A consistent use of 'they/them' as a singular and pronouns beyond 'she/her' or 'he/him' will be credited in exam responses in line with existing mark scheme criteria.

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. Also used to denote a valid answer that goes beyond the expectations of the GCSE syllabus.

R - means reject answer as not creditworthy

NE - means not enough

I - means ignore

DPT - in some questions a specific error made by a candidate, if repeated, could result in the candidate failing to gain 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.

Note to Examiners

In the real world minor syntax errors are often identified and flagged by the development environment. To reflect this, all responses in a high-level programming language will assess a candidate's ability to create an answer using precise programming commands/instructions but will avoid penalising them for minor errors in syntax.

When marking program code, examiners must take account of the different rules between the languages and only consider how the syntax affects the logic flow of the program. If the syntax is not perfect but the logic flow is unaffected then the response should not be penalised.

The case of all program code written by students is to be ignored for the purposes of marking. This is because it is not always clear which case has been used depending on the style and quality of handwriting used.

Examiners must ensure they follow the mark scheme instructions exactly. If an examiner is unsure as to whether a given response is worthy of the marks they must escalate the question to their team leader.

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.

Question	Part	Marking guidance	Total marks
01	**1**	**Mark is for AO2 (apply)** **C** Line number 6; **R.** If more than **one** lozenge shaded	1

Question	Part	Marking guidance	Total marks
01	**2**	**Mark is for AO2 (apply)** **C** This program uses concatenation; **R.** If more than **one** lozenge shaded	1

Question	Part	Marking guidance	Total marks
01	**3**	**Mark is for AO2 (apply)** <; // Less than; **R.** =	1

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>01</td><td>4</td><td>

4 marks for AO2 (apply)

MP1: for the first value of column a **and** the first value of column b **and** the first value of column c **all correct**;

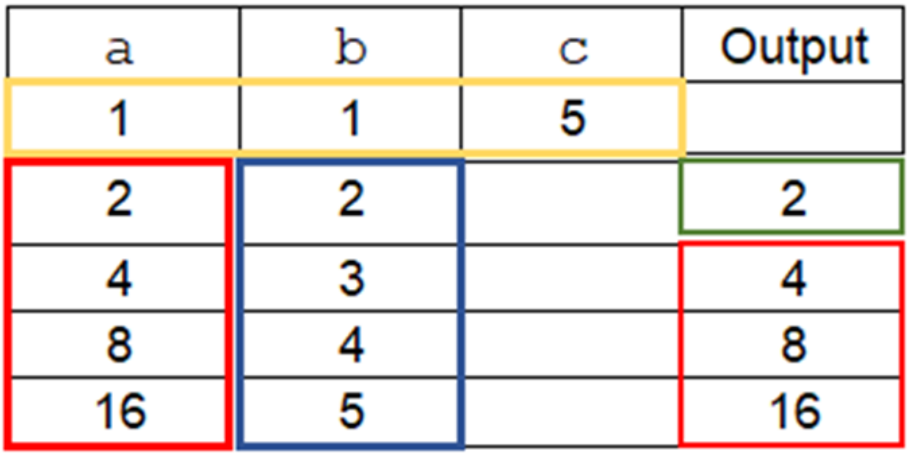
MP2: for the correct first output (2);

MP3: for the rest of column a correct **and** the rest of the Output column correct **and** neither containing any other values at the bottom of the columns;

MP4: for the rest of column b correct **and** containing no other values;

Maximum 3 marks if any errors.

I. Different rows used as long as the order within columns is clear
I. Duplicate values on consecutive rows within a column



<table>
<tr><th>a</th><th>b</th><th>c</th><th>Output</th></tr>
<tr><td>1</td><td>1</td><td>5</td><td></td></tr>
<tr><td>2</td><td>2</td><td></td><td>2</td></tr>
<tr><td>4</td><td>3</td><td></td><td>4</td></tr>
<tr><td>8</td><td>4</td><td></td><td>8</td></tr>
<tr><td>16</td><td>5</td><td></td><td>16</td></tr>
</table>

</td><td>4</td></tr>
</table>

Question	Part	Marking guidance	Total marks
02	**1**	**3 marks for AO3 (program)** **<u>C#</u>** L1 `i % 5 == 0`; L2 `Console.WriteLine("Free biscuit");` L3 `Console.WriteLine("Nothing free");` **A.** `Write` in place of `WriteLine`; **I.** missing `Console.` **<u>Python</u>** L1 `i % 5 == 0`; L2 `print("Free biscuit");` L3 `print("Nothing free");` **<u>VB.NET</u>** L1 `i Mod 5 = 0`; L2 `Console.WriteLine("Free biscuit");` L3 `Console.WriteLine("Nothing free");` **A.** `Write` in place of `WriteLine`; **I.** missing `Console.` **I.** Case **I.** Minor spelling mistakes in output messages **DPT.** Missing parentheses	3

Question	Part	Marking guidance	Total marks
02	2	**4 marks for AO3 (program)** **Program Logic** **Mark A** for using user input and storing the result in a variable for **one** of the number of biscuits, pastries **or** cakes; **Mark B** for using user input and storing the results in **three** variables correctly for the number of biscuits, pastries **and** cakes; **Mark C** for expressions that calculate the total value correctly; **Mark D** for outputting the total value; **DPT.** Repeated identical syntax errors in the three input commands **I.** Case **I.** Messages or no messages with input and output statements **I.** Gaps/spaces throughout the code, except where to do so would explicitly alter the logic of the code in a way that makes it incorrect **Maximum 3 marks** if any errors in code **Note to examiners** In C#/VB.NET examples, explicit variable declarations are not shown. Refer to the specific language type issues section of the appropriate Marking Guidance document. Any correct variable declarations in student answers should be accepted. **C# Example 1 (fully correct)** `totalValue = 0;` `biscuits = Convert.ToInt32(Console.ReadLine());` **A, Part of B** `pastries = Convert.ToInt32(Console.ReadLine());` **Part of B** `cakes = Convert.ToInt32(Console.ReadLine());` **Part of B** `totalValue = biscuits * 1 + pastries * 2.5 + cakes * 3;` **C** `Console.WriteLine(totalValue);` **D** **A.** `Write` in place of `WriteLine` **I.** missing `Console.` **Python Example 1 (fully correct)** `biscuits = int(input())` **A, Part of B** `pastries = int(input())` **Part of B** `cakes = int(input())` **Part of B** `totalValue = biscuits * 1 + pastries * 2.5 + cakes * 3` **C** `print(totalValue)` **D**	4

		VB.NET Example 1 (fully correct) `totalValue = 0` `biscuits = Console.ReadLine()` **A, Part of B** `pastries = Console.ReadLine()` **Part of B** `cakes = Console.ReadLine()` **Part of B** `totalValue = biscuits * 1 + pastries * 2.5 + cakes * 3` **C** `Console.WriteLine(totalValue)` **D** **A.** `Write` in place of `WriteLine` **I.** missing `Console.`	

Question	Part	Marking guidance	Total marks
03		**2 marks for AO3 (design) and 6 marks for AO3 (program)** **<u>Program Design</u>** **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 and regardless of whether the solution works. **Mark A** for using meaningful variable names throughout; **Mark B** for the use of a selection construct using the number of students; **<u>Program Logic</u>** **Mark C** for using user input and storing the result in a variable correctly for the number of students; **Mark D** for **one** correct Boolean condition to check number of students; **A.** if the **only error** is quotes included around numeric values; **Mark E** for correctly checking the number of students under all required conditions; **R.** if any quotes included around numeric values; **Mark F** for **one** expression that calculates total correctly, linked to the condition for the intended pathway; **Mark G** for expressions that calculate total correctly for each of the three pathways; **Mark H** for outputting **only one** total in an appropriate location that covers all included pathways; **A.** if 100 is output appropriately as part of a literal string under the correct condition; **DPT** use of £ sign in calculation or value assigned to variable **DPT** for quotes around numbers **I.** Case **I.** Messages or no messages with input and output statements **I.** Gaps/spaces throughout the code, except where to do so would explicitly alter the logic of the code in a way that makes it incorrect. **Maximum 7 marks** if any errors in code **Note to examiners** In C#/VB.NET examples, explicit variable declarations are not shown. Refer to the specific language type issues section of the appropriate Marking Guidance document. Any correct variable declarations in student answers should be accepted.	8

C# Example 1 (fully correct)

All design marks are achieved (**Marks A and B**)

```
students = Convert.ToInt32(Console.ReadLine());         C
if (students > 25) {                                    D
   total = 100;                                         F
}
else if (students > 15) {                               E
   total = 20 + (3 * students);                         Part G
}
else {
   total = 20 + (5 * students);                         Part G
}
Console.WriteLine(total);                               H
```

A. `Write` in place of `WriteLine`

I. missing `Console.`

Python Example 1 (fully correct)

All design marks are achieved (**Marks A and B**)

```
students = int(input())                                 C
if students > 25:                                       D
   total = 100                                          F
elif students > 15:                                     E
   total = 20 + (3 * students)                          Part G
else:
   total = 20 + (5 * students)                          Part G
print(total)                                            H
```

Python Example 2 (fully correct)

All design marks are achieved (**Marks A and B**)

```
students = int(input())                                 C
if students <= 15:                                      D
   print(20 + 5 * students)                             F
elif students > 15 and students <= 25:                  E
   print(20 + 3 * students)                             Part G
else:
   print(100)                                           Part G
```

Python Example 3 (7 marks)

All design marks are achieved (**Marks A and B**)

No **Mark E** because `> 25` is caught by `students > 15` condition.

```
students = int(input())                                 C
if students <= 15:                                      D
   total = 20 + (5 * students)                          F
elif students > 15:                                     Part E
   total = 20 + (3 * students)                          Part G
elif students > 25:                                     Part E
   total = 100                                          Part G
print(total)                                            H
```

<u>VB.NET Example 1 (fully correct)</u>

All design marks are achieved (**Marks A and B**)

```
 students = Console.ReadLine()                                C
 If students > 25 Then                                        D
    total = 100                                               F
 ElseIf students > 15 Then                                    Part E
    total = 20 + (3 * students)                               Part G
 Else                                                         Part E
    total = 20 + (5 * students)                               Part G
 End If
 Console.WriteLine(total)                                     H
```

A. `Write` in place of `WriteLine`

I. missing `Console.`

<u>VB.NET Example 2 (7 Marks)</u>

All design marks are achieved (Marks A and B)

No Mark E because > 25 is caught by students > 15 condition.

```
total = 0
students = Console.ReadLine()                                 C
If students <= 15 Then                                        D
   total = 20 + (5 * students)                                F
ElseIf students > 15 Then                                     Part E
   total = 20 + (3 * students)                                Part G
ElseIf students > 25 Then                                     Part E
   total = 100                                                Part G
End If
Console.WriteLine(total)                                      H
```

A. `Write` in place of `WriteLine`

I. missing `Console.`

Question	Part	Marking guidance	Total marks
04	1	**Mark is for AO1 (recall)** **D** Checks that the input is reasonable; **R.** If more than **one** lozenge shaded	1

Question	Part	Marking guidance	Total marks
04	2	**1 mark for AO3 (design), 5 marks for AO3 (program)** **Program Design** **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 and regardless of whether the solution works. **Mark A** for the use of an iteration structure that exists within their language, which attempts to repeat until a first name with a valid length is entered; **Program Logic** **Mark B** for finding the length of **at least one** string correctly; **Mark C** for **one** Boolean condition that attempts to check the first name or length of first name entered against one of the two required conditions (even if the length usage is incorrectly coded or missing); **A.** if the **only error** is quotes included around numeric values; **Mark D** if **all** required checks on the length of the first name have been carried out **and** combined correctly; **R.** if any quotes included around numeric values **Mark E** for code that allows the re-inputting of a first name within an iteration structure; **Mark F** for a structure that would output **either** `Name accepted` or `Name not accepted` correctly in **all** circumstances; **A.** if conditions are not fully correct; **A.** equivalent messages; **I.** Case **I.** Messages or no messages with input statements **I.** Gaps/spaces throughout the code, except where to do so would explicitly alter the logic of the code in a way that makes it incorrect. **Maximum 5 marks** if any errors in code. **Note to examiners** In C#/VB.NET examples, explicit variable declarations are not shown. Refer to the specific language type issues section of the appropriate Marking Guidance document. Any correct variable declarations in student answers should be accepted.	6

<u>C# Example 1 (fully correct)</u>

The design mark is achieved (**Mark A**)

```
name = Console.ReadLine();
while (name.Length < 2 || name.Length > 14) {      BCD, Part E

   Console.WriteLine("Name not accepted");         Part F
   name = Console.ReadLine();                      Part E
}
Console.WriteLine("Name accepted");                Part F
```

A. `Write` in place of `WriteLine`

I. missing `Console.`

<u>Python Example 1 (fully correct)</u>

The design mark is achieved (**Mark A**)

```
name = input()
while len(name) < 2 or len(name) > 14:             BCD, Part E
   print("Name not accepted")                      Part F
   name = input()                                  Part E
print("Name accepted")                             Part F
```

<u>Python Example 2 (fully correct)</u>

The design mark is achieved (**Mark A**)

```
check = False                                      Part E
while check == False:                              Part E
   name = input()                                  Part E
   if len(name) > 1 and len(name) < 15:            BCD
      print("Name accepted")                       Part F
      check = True
   else:
      print("Name not accepted")                   Part F
```

<u>Python Example 3 (fully correct)</u>

The design mark is achieved.

Check that all pathways are covered when using nested if statements and all required messages are included.

```
while True:                                        Part E
   name = input()
   if len(name) > 1:                               BC
      if len(name) < 15:                           D
         print("Name accepted")                    Part F
         break
      else:
         print("Name not accepted")                Part F
   else:
      print("Name not accepted")                   Part F
```

Python Example 4 (partially correct – 4 marks)

The design mark is not achieved.
There is no use of iteration (A) so the name cannot be re-input (E).

```
name = input()
if len(name) < 2 or len(name) > 14:                    BCD
   print("Name not accepted")                          Part F
   name = input()
print("Name accepted")                                 Part F
```

VB.NET Example 1 (fully correct)

The design mark is achieved (**Mark A**)

```
name = Console.ReadLine()
While name.Length < 2 Or name.Length > 14          BCD, Part E
   Console.WriteLine("Name not accepted")          Part F
   name = Console.ReadLine()                       Part E
End While
Console.WriteLine("Name accepted")                 Part F
```

A. `Write` in place of `WriteLine`;
I. missing `Console.`

VB.NET Example 2 (fully correct)

The design mark is achieved (**Mark A**)

```
While True                                          Part E
   name = Console.ReadLine()                        Part E
   If Len(name) < 2 Or Len(name) > 14 Then          BCD
      Console.WriteLine("Name not accepted")        Part F
   Else
      Exit While
   End If
End While
Console.WriteLine("Name accepted")                  Part F
```

A. `Write` in place of `WriteLine`;
I. missing `Console.`

VB.NET Example 3 (fully correct)

The design mark is achieved.
Check that all pathways are covered when using nested if statements and all required messages are included.

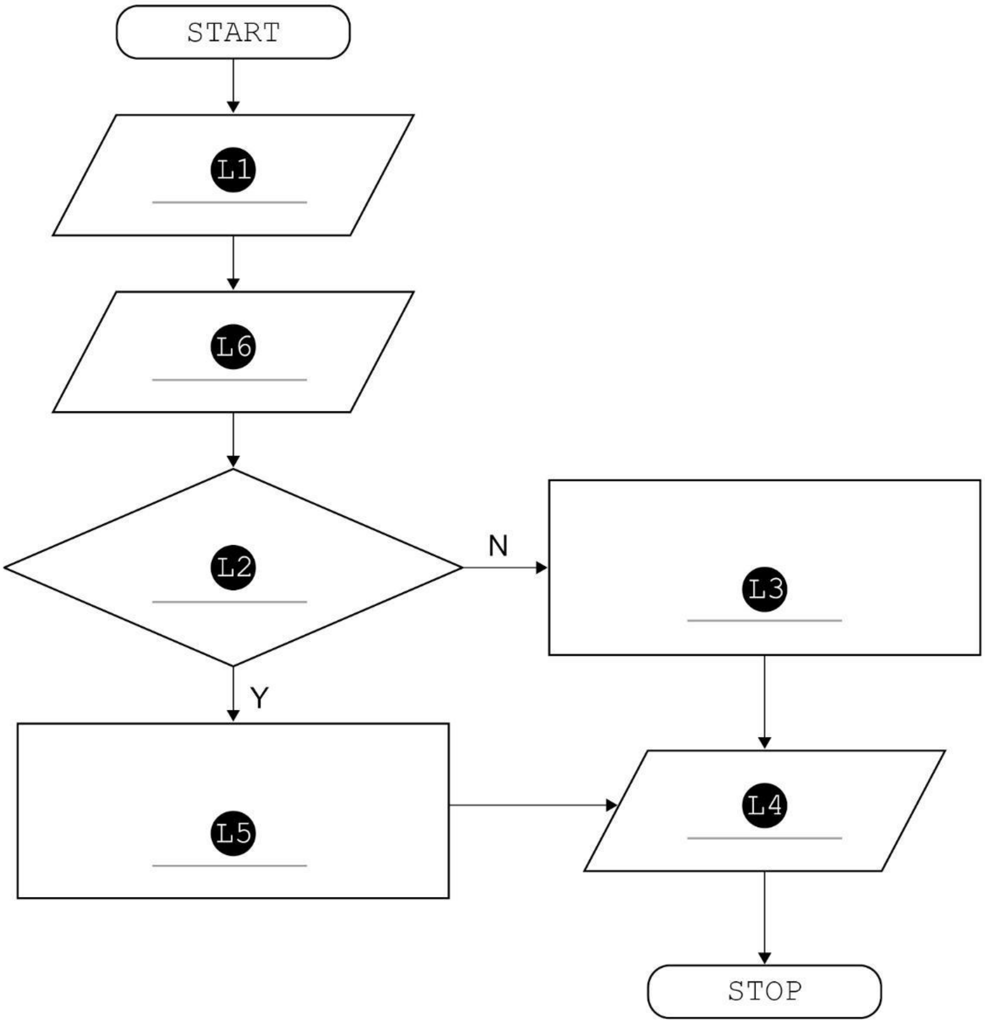
```
While True                                         Part E
   name = Console.ReadLine()
   If Len(name) > 1 Then                           BC
      If Len(name) < 15 Then                       D
         Console.WriteLine("Name accepted")        Part F
         Exit While
      Else
         Console.WriteLine("Name not accepted")    Part F
      End If
   Else
      Console.WriteLine("Name not accepted")       Part F
   End If
End While
```

A. `Write` in place of `WriteLine`;
I. missing `Console`.

Question	Part	Marking guidance	Total marks
04	**3**	**Mark is for AO1 (recall)** **B** Data that is at the limits of the allowed range; **R.** If more than **one** lozenge shaded	1

Question	Part	Marking guidance	Total marks
04	**4**	**Mark is for AO3 (test)** 1 // 15; **A.** if both correct answers are given;	1

Question	Part	Marking guidance	Total marks
04	**5**	**Mark is for AO1 (recall)** Erroneous; **A.** Invalid; **A.** Description of a type of erroneous test data e.g. data of the wrong type; **I.** Minor spelling mistakes **R.** Typical **R.** Extreme **R.** Boundary **R.** Normal	1

Question	Part	Marking guidance	Total marks
04	**6**	**3 marks for AO3 (design)** **MP1**: for L1, L6 (**I.** Order of L1 and L6) **and** L4; **MP2**: for L2; **MP3**: for L3 and L5; **I.** any pseudocode if labels are given **A.** The **exact given** pseudo-code statements if labels are omitted; L1 `firstName ← USERINPUT` L2 `LEN(firstName) > 3` L3 `username ← firstName + lastName` L4 `OUTPUT username` L5 `username ← SUBSTRING(0, 2, firstName) + lastName` L6 `lastName ← USERINPUT`  	3

Question	Part	Marking guidance	Total marks
05	1	**Mark is for AO1 (recall)** (Specifies the) type of value/data a variable stores/has; **R.** Information in place of Data // Defines the type of operations that can be performed on a value (stored in a variable);	1

Question	Part	Marking guidance	Total marks
05	2	**Mark is for AO2 (apply)** The values would be concatenated if they were strings; // The values need to be added to the other values; // The arithmetic operator can only be used on numeric values;	1

Question	Part	Marking guidance	Total marks

05	3	**Mark is for AO2 (apply)**
		C#
		Maximum of 1 mark from:
		`double;`
		`float;`
		`decimal;`
		Python
		`float;`
		VB.NET
		Maximum of 1 mark from:
		`Double;`
		`Single;`
		A. `Decimal;`
		I. Case
		R. Real
			1

Question	Part	Marking guidance	Total marks
05	4	**1 mark for AO3 (design), 4 marks for AO3 (program)**	5

Program Design
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 and regardless of whether the solution works.

Mark A for the use of an iteration structure that exists within their language;

Program Logic

Mark B for correct conditions within an iteration structure to correctly loop five times;

Mark C for using user input **once** in a logically appropriate place;

Mark D for correctly calculating the total from the inputted values;

Mark E for correctly outputting total in a logically appropriate place outside the iteration structure;

I. Case
I. Messages or no messages with input and output statements
I. Gaps/spaces throughout the code, except where to do so would explicitly alter the logic of the code in a way that makes it incorrect.

Maximum 4 marks if any errors in code.

Note to examiners
In C#/VB.NET examples, explicit variable declarations are not shown. Refer to the specific language type issues section of the appropriate Marking Guidance document. Any correct variable declarations in student answers should be accepted.

C# Example 1 (fully correct)
The design mark is achieved (**Mark A**)

```
total = 0;                                          Part of D
num = 0;                                            Part of C
for (i = 0; i < 5; i++) {                           B
   num = Convert.ToInt32(Console.ReadLine());       Part of C
   total = total + num;                             Part of D
}
Console.WriteLine(total);                           E
```

A. `Write` in place of `WriteLine;`
I. missing `Console.`

C# Example 2 (fully correct)

The design mark is achieved (**Mark A**)

```
total = 0;                                          Part of D
counter = 0;                                        Part of B
num = 0;                                            Part of C
while (counter < 5) {                               Part of B
   counter = counter + 1;                           Part of B
   num = Convert.ToInt32(Console.ReadLine());       Part of C
   total = total + num;                             Part of D
}
Console.WriteLine(total);                           E
```

A. `Write` in place of `WriteLine;`

I. missing `Console.`

Python Example 1 (fully correct)

The design mark is achieved (**Mark A**)

```
total = 0                                           Part of D
for i in range(5):                                  B
   num = int(input())                               C
   total = total + num                              Part of D
print(total)                                        E
```

Python Example 2 (fully correct)

The design mark is achieved (**Mark A**)

```
total = 0                                           Part of D
count = 0                                           Part of B
while count < 5:                                    Part of B
   count = count + 1                                Part of B
   num = int(input())                               C
   total = total + num                              Part of D
print(total)                                        E
```

VB.NET Example 1 (fully correct)

The design mark is achieved (**Mark A**)

```
total = 0                                           Part of D
num = 0                                             Part of C
For i = 0 to 4                                      B
   num = Console.ReadLine()                         Part of C
   total = total + num                              Part of D
Next
Console.WriteLine(total)                            E
```

A. `Write` in place of `WriteLine;`

I. missing `Console.`

VB.NET Example 2 (fully correct)

The design mark is achieved (**Mark A**)

```
num = 0                                            Part of C
counter = 0                                        Part of B
total = 0                                          Part of D
While counter < 5                                  Part of B
   counter = counter + 1                           Part of B
   num = Console.ReadLine()                        Part of C
   total = total + num                             Part of D
End While
Console.WriteLine(total)                           E
```

A. `Write` in place of `WriteLine;`

I. missing `Console.`

Question	Part	Marking guidance	Total marks
06	1	**3 marks for AO2 (apply)**	3

1 mark for each correct change in correct order;

I. Blanks for repeated values.

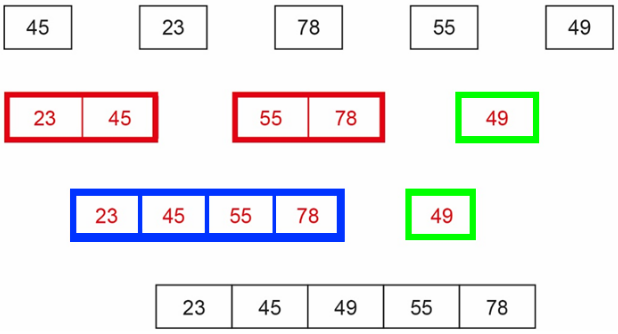
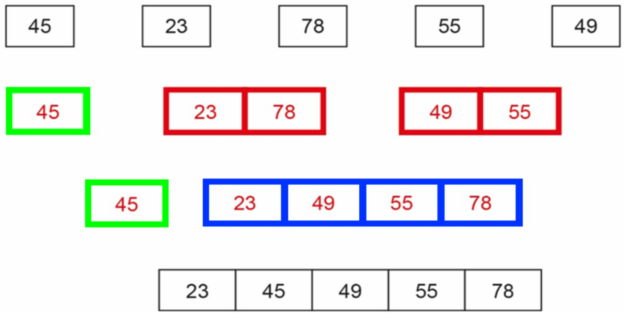
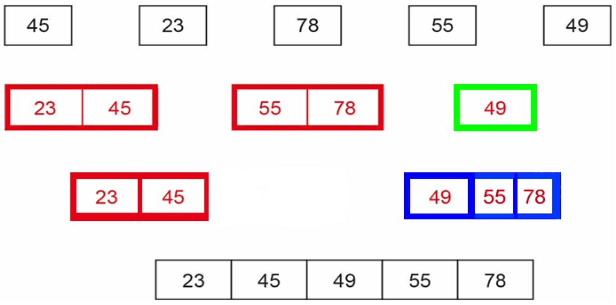
Maximum 2 marks if grid not fully correct.

The correct sequence is:

45	23	78	55	49
23	45	78	55	49
23	45	55	78	49
23	45	55	49	78
23	45	49	55	78

If the response shows the sort completed in a single pass, as follows, then **award 1 mark**;

45	23	78	55	49
23	45	55	49	78
23	45	49	55	78
23	45	49	55	78

Question	Part	Marking guidance	Total marks
06	**2**	**3 marks for AO2 (apply)** **MP1:** for correctly sorting pairs (Red Boxes); **MP2:** for correctly leaving 1 element on its own after sorting pairs (**Green** Boxes); **MP3:** for correctly sorting 3 or 4 elements (Blue Boxes); **Maximum two marks** if any errors **I.** if the final row has been rewritten by the student **Example One (fully correct)**  45 23 78 55 49 23 45 55 78 49 23 45 55 78 49 23 45 49 55 78 **Example Two (fully correct)**  45 23 78 55 49 45 23 78 49 55 45 23 49 55 78 23 45 49 55 78 **Example Three (fully correct)**  45 23 78 55 49 23 45 55 78 49 23 45 49 55 78 23 45 49 55 78	3

Question	Part	Marking guidance	Total marks
06	**3**	**2 marks for AO1 (understanding)** **Advantage** of bubble sort **Maximum of 1 mark from:** ● Bubble sort is simpler to code // Algorithm requires fewer lines of code; ● Bubble sort can be quicker to sort small lists/arrays; **A.** Bubble sort will use less memory // bubble sort only requires one additional memory location (whereas merge sort requires more than one) ; **Disadvantage** of bubble sort **Maximum of 1 mark from:** ● Bubble sort does not sort (large) lists/arrays as quickly as merge sort; ● Bubble sort is less efficient; **NE.** Answers that have not been qualified (e.g. bubble sort is simple).	**2**

Question	Part	Marking guidance	Total marks
06	**4**	**2 marks for AO2 (apply)** ● A binary search is more efficient (on average) / takes less time (on average) ; // a linear search would be less efficient (on average) because it (potentially) must check **all** (2500) elements; ● (because) fewer comparisons will need to be made (on average) to locate the search term; // (For this scenario) a binary search would (take at maximum 12 searches as it) halves the number of values **each time**;	2

Question	Part	Marking guidance	Total marks
07	1	**Mark is for AO2 (apply)** **Maximum of 1 mark** from: ● To determine if a number/`x` is odd/even // To determine if a number/`x` is a multiple of 2; ● To return `True` if a number/`x` is even // To return `False` if a number is odd; ● To return `True` if a number/`x` is a multiple of 2 // To return `False` if a number/`x` is not (a multiple of 2); ● To return `True` if the remainder (of the integer division) is 0 // To return `False` if the remainder is 1 / not 0;	1

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>07</td><td>2</td><td><b>2 marks for AO2 (apply)</b>
<table>
<tr><th>Program Code</th><th>Output</th></tr>
<tr><td><code>OUTPUT number</code></td><td>10;</td></tr>
<tr><td><code>OUTPUT x</code></td><td>0;</td></tr>
</table>
</td><td>2</td></tr>
</table>

Question	Part	Marking guidance	Total marks
07	3	**Mark is for AO2 (apply)** `True`; **I.** Case	1

Question	Part	Marking guidance	Total marks
07	4	**Mark is for AO2 (apply)** `number`; **I.** Case	1

Question	Part	Marking guidance	Total marks
07	5	**2 marks for AO1 (understanding)** **Maximum of 2 marks** from: • (is) modularised // broken down into sections/chunks // (uses) subroutines // named/out of line blocks of code; • (uses) parameters; • (uses) return values; • (uses) local variables; • (uses) selection; • (uses) iteration; • (uses) well documented interfaces;	2

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>08</td><td>1</td><td>
<b>5 marks for AO2 (apply)</b>
<b>MP1:</b> for <code>b1</code> column <b>and</b> <code>b2</code> column correct <b>and</b> no other values in either column;
<b>MP2:</b> for the first value of <code>i</code> initialised to <code>0</code>;
<b>MP3:</b> for the last three rows of column <code>i</code> correct <b>and</b> no other values;
<b>MP4:</b> for the first value in <code>new</code> set to <code>'0'</code> <b>or</b> the first value in <code>new</code> set to an empty string/blank value followed by <code>'0'</code>;
<b>MP5:</b> for the last three rows of column <code>new</code> correct <b>and</b> no other values;
<b>Maximum 4 marks</b> if any errors.
<table>
<tr><th>b1</th><th>b2</th><th>new</th><th>i</th></tr>
<tr><td>'0010'</td><td>'0111'</td><td>''</td><td>0</td></tr>
<tr><td></td><td></td><td>'0'</td><td>1</td></tr>
<tr><td></td><td></td><td>'01'</td><td>2</td></tr>
<tr><td></td><td></td><td>'010'</td><td>3</td></tr>
<tr><td></td><td></td><td>'0101'</td><td></td></tr>
</table>
<b>I.</b> Different rows used as long as the order within columns is clear
<b>I.</b> Duplicate values on consecutive rows within a column
<b>I.</b> Missing quotes used around strings.
</td><td>5</td></tr>
</table>

Question	Part	Marking guidance	Total marks
08	2	**Mark is for AO2 (apply)** So the algorithm does not attempt to access a character that does not exist (in the string); // So the algorithm does not attempt to use an index value that is out-of-range; // Because indexing (of strings) starts at 0 rather than 1; **A.** To make sure it doesn't crash (when run as a program) ; **A.** To prevent a run-time error from happening;	1

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>09</td><td></td><td><b>2 marks for (AO2 apply)</b>
<b>MP1:</b> for any <b>three</b> columns correct;
<b>MP2:</b> for <b>all</b> columns correct;
<table>
<tr><th></th><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th></tr>
<tr><td>Algorithm totals the five values</td><td>✓</td><td>✓</td><td></td><td></td><td></td></tr>
<tr><td>Algorithm does <b>not</b> total the five values</td><td></td><td></td><td>✓</td><td>✓</td><td>✓</td></tr>
</table>
<b>R.</b> column if more than 1 tick in that column</td><td>2</td></tr>
</table>

Question	Part	Marking guidance	Total marks
10	1	**Mark is for AO2 (apply)** `Runner('200 m', 10, 32.59);` **I.** Case **R.** if quotes, commas or parentheses missing	1

Question	Part	Marking guidance	Total marks
10	2	**Mark is for AO2 (apply)** `.time`; **I.** Case **R.** if period is missing	1

Question	Part	Marking guidance	Total marks
10	3	**1 mark for AO3 (design), 5 marks for AO3 (program)** **Program Design** **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 and regardless of whether the solution works. **Mark A** for attempting to calculate the length of the array `times`; **Program Logic** **Mark B** for initialising a variable to an appropriate numeric value to store the fastest time or the index to the fastest time; **Mark C** for using a loop to correctly iterate through an array of any length; **Mark D** for correctly checking values (on at least 2 occasions) in the `times` array are smaller than the current fastest time (even if the indexing is incorrect); **Mark E** correctly store the current fastest time/position of the fastest time for each iteration (indexing must be correct); **Mark F** for outputting the fastest time from a variable **once**, in an appropriate place; **Note to Examiners** If used built in routines to find the smallest value in an array or to sort the array then do **not** award **marks D**, **E** **I.** Case **I.** Gaps/spaces throughout the code, except where to do so would explicitly alter the logic of the code in a way that makes it incorrect. **Maximum 5 marks** if any errors in code. **Note to examiners** In C#/VB.NET examples, explicit variable declarations are not shown. Refer to the specific language type issues section of the appropriate Marking Guidance document. Any correct variable declarations in student answers should be accepted.	6

C# Example 1 (fully correct)
All design marks are achieved (**Mark A**)

```
length = times.Length;                           Part of C
best = times[0];                                 B, Part of D
for (i = 0; i < length; i++) {                   Part of C
   if (times[i] < best) {                        Part of D
      best = times[i];}                          E
}
Console.WriteLine(best);                         F
```

A. `Write` in place of `WriteLine;`
I. missing `Console.`

C# Example 2 ((fully correct)
All design marks are achieved (**Mark A**)

```
best = times[0];                                 B, Part of D
foreach (time in times) {                        C
   if (time < best) {                            Part of D
      best = time;}                              E
}
Console.WriteLine(best);                         F
```

A. `Write` in place of `WriteLine;`
I. missing `Console.`

Python Example 1 (fully correct)
All design marks are achieved (**Mark A**)

```
length = len(times)                              Part of C
best = times[0]                                  B, Part of D
for i in range(length):                          Part of C
   if times[i] < best:                           Part of D
      best = times[i]                            E
print(best)                                      F
```

Python Example 2 (fully correct)
All design marks are achieved (**Mark A**)

```
best = times[0]                                  B, Part of D
for time in times:                               C
   if time < best:                               Part of D
      best = time                                E
print(best)                                      F
```

Python Example 3 (fully correct)
All design marks are achieved (**Mark A**)

```
lowest = 0                                       B, Part of D
for i in range(0, len(times)):                   C
   if times[i] < times[lowest]:                  Part of D
      lowest = i                                 E
print(times[lowest])                             F
```

VB.NET Example 1 (fully correct)

All design marks are achieved (**Mark A**)

```
length = times.Length                                  Part of C
best = times(0)                                        B, Part of D
For i = 1 to length - 1                                Part of C
   If times(i) < best Then                             Part of D
      best = times(i)                                  E
   End If
Next
Console.WriteLine(best)                                F
```

A. `Write` in place of `WriteLine`;

I. missing `Console.`

VB.NET Example 2 (fully correct)

All design marks are achieved (**Mark A**)

```
best = times(0)                                        B, Part of D
For Each time In times                                 C
   If time < best Then                                 Part of D
      best = time                                      E
   End If
Next
Console.WriteLine(best)                                F
```

A. `Write` in place of `WriteLine`;

I. missing `Console.`

Question	Part	Marking guidance	Total marks
11	**1**	**Mark is for AO2 (apply)** **B** `s`; **R.** If more than **one** lozenge shaded	1

Question	Part	Marking guidance	Total marks
11	**2**	**Mark is for AO2 (apply)** **A** `s`; **R.** If more than **one** lozenge shaded	1

Question	Part	Marking guidance	Total marks
11	**3**	**Mark is for AO2 (apply)** **C** 6; **R.** If more than **one** lozenge shaded	1

Question	Part	Marking guidance	Total marks
11	4	**Mark is for AO2 (apply)** **C** `sys*em`; **R.** If more than **one** lozenge shaded	1

Question	Part	Marking guidance	Total marks
11	5	**2 marks for AO3 (design), 8 marks for AO3 (program)** **<u>Program Design</u>** **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 and regardless of whether the solution works. **Mark A** for creating a new iteration construct that exists within their language and inputting a letter within it; **Mark B** for the use of a selection structure for **both** the messages about winning and losing; **<u>Program Logic</u>** **Mark C** for an iteration structure that correctly allows a letter to be entered a maximum of 8 times (**I.** any reference to exiting the loop based on the number of asterisks); **Mark D** for an iteration structure that exits correctly when there are no asterisks left in `hidden` and there is a correct boolean joining operator for BOTH conditions (See **Mark C** for second condition); **Mark E** for correctly calling the `findLetter` subroutine within an iteration structure (even if the parameters are incorrect or missing); **Mark F** for including the correct parameters in the correct order when calling the `findLetter` subroutine; **Note:** the middle parameter should match the variable name their user input has been assigned to. **Mark G** for updating the `hidden` variable with the return value from the call to `findLetter`; **R.** if hidden is re-initialised within the iteration structure **Mark H** for correctly checking to see if there are any asterisks left in `hidden` and handling it appropriately; **Mark I** for displaying the value of `hidden` after **each attempt**; **Mark J** for displaying `You won` **and** `You lost` in appropriate places under the correct conditions;	10

Note to Examiners
If used built in routines to check if a string contains another string/character or to count the number of asterisks then do **not** award **marks D and H**

I. Case
I. Messages or no messages with input statements
I. Gaps/spaces throughout the code, except where to do so would explicitly alter the logic of the code in a way that makes it incorrect.

Maximum 9 marks if any errors in code.

Note to examiners
In C#/VB.NET examples, explicit variable declarations are not shown. Refer to the specific language type issues section of the appropriate Marking Guidance document. Any correct variable declarations in student answers should be accepted.

C# Example 1 (fully correct)
All design marks are achieved (**Marks A and B**)

```
won = false;                                          Part D
for (i = 0; i < 8; i++) {                             C
   letter = Console.ReadLine();
   hidden = findLetter(word, letter, hidden);         EFG
   Console.WriteLine(hidden);                         I
   won = true;                                        Part D
   for (j = 0; j < hidden.Length; j++) {              Part H
      if (hidden[j] == '*') {                         Part H
         won = false;                                 Part H
      }
   }
   if (won == true) {                                 Part D
      break;                                          Part D
   }
}
if (won == false) {                                   Part J
   Console.WriteLine("You lost");                     Part J
} else {                                              Part J
   Console.WriteLine("You won");                      Part J
}
```

A. `Write` in place of `WriteLine;`
I. missing `Console.`

C# Example 2 (fully correct)

All design marks are achieved (**Marks A and B**)

```
won = false;                                           Part D
x = 0;                                                 Part C
while (x < 8 && won == false)) {                       Part CD
   letter = Console.ReadLine();
   hidden = findLetter(word, letter, hidden);          EFG
   Console.WriteLine(hidden);                          I
   won = true;                                         Part D
   y = 0;                                              Part H
   while (y < hidden.Length) {                         Part H
      if (hidden[j] == '*') {                          Part H
         won = false;                                  Part H
      }
      y = y + 1;                                       Part H
   }
   x = x + 1;                                          Part C
}
if (won == false) {                                    Part J
   Console.WriteLine("You lost");                      Part J
} else {                                               Part J
   Console.WriteLine("You won");                       Part J
}
```

A. `Write` in place of `WriteLine;`

I. missing `Console.`

Python Example 1 (fully correct)

All design marks are achieved (**Marks A and B**)

```
won = False                                            Part D
for i in range(8):                                     C
   letter = input()
   hidden = findLetter(word, letter, hidden)           EFG
   print(hidden)                                       I
   won = True                                          Part D
   for j in range(len(hidden)):                        Part H
      if hidden[j] == "*":                             Part H
         won = False                                   Part H
   if won == True:                                     Part D
      break                                            Part D
if won == False:                                       Part J
   print("You lost")                                   Part J
else:                                                  Part J
   print("You won")                                    Part J
```

Python Example 2 (fully correct)

All design marks are achieved (**Marks A and B**)

```
won = False                                         Part D
i = 0                                               Part C
while i < 8 and won == False:                       Part CD
   letter = input()
   hidden = findLetter(word, letter, hidden)        EFG
   print(hidden)                                    I
   won = True                                       Part D
   j = 0                                            Part H
   while j < len(hidden):                           Part H
      if hidden[j] == "*":                          Part H
         won = False                                Part H
      j = j + 1                                     Part H
   i = i + 1                                        Part C
if won == False:                                    Part J
   print("You lost")                                Part J
else:                                               Part J
   print("You won")                                 Part J
```

Python Example 3 (fully correct)

All design marks are achieved (**Marks A and B**)

```
won = False                                         Part D
for count in range(8):                              C
   letter = input()
   hidden = findLetter(word, letter, hidden)        EFG
   print(hidden)                                    I
   if word == hidden:                               Part D, H
      won = True                                    Part D
      print("You won")                              Part J
      break                                         Part D
if won == False:                                    Part J
   print("You lost")                                Part J
```

VB.NET Example 1 (fully correct)
All design marks are achieved (**Marks A and B**)

```
won = False                                         Part D
For i = 0 To 7                                      C
   letter = Console.ReadLine()
   hidden = findLetter(word, letter, hidden)        EFG
   Console.WriteLine(hidden)                        I
   won = True                                       Part D
   For j = 0 To hidden.Length - 1                   Part H
      If hidden(j) = "*" Then                       Part H
         won = False                                Part H
      End If
   Next
   If won = True Then                               Part D
      Exit For                                      Part D
   End If
Next
If won = False Then                                 Part J
   Console.WriteLine("You lost")                    Part J
Else                                                Part J
   Console.WriteLine("You won")                     Part J
End If
```

A. `Write` in place of `WriteLine;`
I. missing `Console.`

VB.NET Example 2 (fully correct)
All design marks are achieved (**Marks A and B**)

```
won = False                                         Part D
x = 0                                               Part C
While x < 8 And won = False                         Part CD
   letter = Console.ReadLine()
   hidden = findLetter(word, letter, hidden)        EFG
   Console.WriteLine(hidden)                        I
   won = True                                       Part D
   y = 0                                            Part H
   While y < hidden.Length                          Part H
      If hidden(j) = "*" Then                       Part H
         won = False                                Part H
      End If
      y = y + 1                                     Part H
   End While
   x = x + 1                                        Part C
End While
If won = False Then                                 Part J
   Console.WriteLine("You lost")                    Part J
Else                                                Part J
   Console.WriteLine("You won")                     Part J
End If
```

A. `Write` in place of `WriteLine;`
I. missing `Console.`

VB.NET Example 3 (fully correct)

All design marks are achieved (**Marks A and B**)

```
won = False                                             Part D
For count = 1 To 8                                      C
   letter = Console.ReadLine()
   hidden = findLetter(word, letter, hidden)            EFG
   Console.WriteLine(hidden)                            I
   If word = hidden Then                                Part D, H
      won = True                                        Part D
      Console.WriteLine("You won")                      Part J
      Exit For                                          Part D
   End If
Next
If won = False Then                                     Part J
   Console.WriteLine("You lost")                        Part J
End If
```

A. `Write` in place of `WriteLine;`

I. missing `Console.`