01	1	**3 marks for AO2 (apply)**
		Stop marking at the first error.
		(Compare) 30 with 21/position 3;
		(Compare) 30 with 31/position 5;
		(Compare) 30 with 27/position 4;
			3

01	2	**1 mark for AO1 (understanding)** (The array) must be ordered/sorted;	1

Qu	Part	Marking guidance	Total marks
02		**3 marks for AO1 (understanding)** Start at the beginning (of the array/list); compare each element/item until the value being searched for is found; or the end of the array/list is reached;	3

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>03</td><td>1</td><td>

4 marks for AO2 (apply)

<table>
<tr><th>animalToFind</th><th>validAnimal</th><th>start</th><th>finish</th><th>mid</th></tr>
<tr><td>wolf</td><td>False</td><td>0</td><td>7</td><td>3</td></tr>
<tr><td></td><td></td><td>4</td><td></td><td>5</td></tr>
<tr><td></td><td></td><td>6</td><td></td><td>6</td></tr>
<tr><td></td><td></td><td>7</td><td></td><td>7</td></tr>
<tr><td></td><td>True</td><td></td><td></td><td></td></tr>
</table>

1 mark for correct `animalToFind`, `validAnimal` and `finish` columns and no other values;
1 mark for correct `start` column and no other values;
1 mark for having 5 as second value in `mid` column;
1 mark for the rest of `mid` column being correct and 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

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

Question	Part	Marking guidance	Total marks
03	**2**	**3 marks for AO3 (design), 4 marks for AO3 (program)** **Note to Examiners:** As the question asks them to write a routine for the list `fruits`, if they have used a series of connected selection statements rather than an iteration statement they can still gain marks C and D. **<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 asking the user to input a word and storing the input in a variable; **Mark B** for using iteration; **Mark C** for attempting to check **each** index location within the list/array `fruits`; **<u>Program Logic</u>** **Mark D** for a loop which starts at one end of the list/array `fruits` and **could** correctly iterate through each index to the other end; **Mark E** for correctly comparing all 6 fruits against the word to find; **Mark F** for code which correctly **processes** a match; **Mark G** for outputting **only one** of the two messages `True` **and** `False` depending on the result of the match; **Maximum 5 marks** if they have not written their own linear search. **Maximum 6 marks** if any errors in code. **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. **I.** any rewritten code that defines `fruit`	7

C# Example 1 (fully correct)

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

```
string[] fruits = { "banana", "apple", "orange",
"pear", "grape", "pineapple" };
string wordToFind = Console.ReadLine();
bool found = false;
int count = 0;
while (found == false && count < fruits.Length)       (Part of D,
{                                                      Part of F)
   if (fruits[count] == wordToFind)                          (E)
   {
      found = true;                                          (F)
   }
   count += 1;
}
if (found == true)                                    (Part of D)
{
   Console.WriteLine("True");                         (Part of G)
}
else
{
   Console.WriteLine("False");                        (Part of G)
}
```

I. Indentation in C#

A. `Write` in place of `WriteLine`

C# Example 2 (fully correct)

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

```
string[] fruits = { "banana", "apple", "orange",
"pear", "grape", "pineapple" };
string wordToFind = Console.ReadLine();
bool found = false;
foreach (string fruit in fruits)                      (Part of D)
{
   if (fruit == wordToFind)                                  (E)
   {
      found = true;                                   (Part of F)
      break;                                          (Part of F)
   }
}
if (found == true)                                    (Part of F)
{
    Console.WriteLine("True");                        (Part of G)
}
else
{
    Console.WriteLine("False");                       (Part of G)
}
```

I. Indentation in C#

A. `Write` in place of `WriteLine`

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

```
fruits = ["banana", "apple", "orange", "pear",
"grape", "pineapple"]
wordToFind = input()
found = False
count = 0
while found == False and count <= len(fruits) - 1:     (Part of D,
                                                        Part of F)
   if fruits[count] == wordToFind:                     (E)
      found = True                                     (Part of F)
   count += 1                                          (Part of D)
if found == True:
   print("True")                                       (Part of G)
else:
   print("False")                                      (Part of G)
```

Python Example 2 (fully correct)
All design marks are achieved (**Marks A**, **B** and **C**)

```
fruits = ["banana", "apple", "orange", "pear",
"grape", "pineapple"]
wordToFind = input()
found = False
for fruit in fruits:                                   (D)
   if fruit == wordToFind:                             (E)
      found = True                                     (Part of F)
      break                                            (Part of F)
if (found == True):                                    (Part of F)
   print("True")                                       (Part of G)
else:
   print("False")                                      (Part of G)
```

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

```
Dim fruits() As String = {"banana", "apple", "orange",
"pear", "grape", "pineapple"}
Dim wordToFind As String = Console.ReadLine()
Dim found As Boolean = False
Dim count As Integer = 0
While found = False And count <= fruits.GetLength(0) – 1   (Part of D, Part of F)
   If fruits(count) = wordToFind Then                      (E)
      found = True                                         (Part of F)
   End If
   count = count + 1                                       (Part of D)
End While
If found = True Then
   Console.WriteLine("True")                               (Part of G)
Else
   Console.WriteLine("False")                              (Part of G)
End If
```

I. Indentation in VB.NET
A. `Write` in place of `WriteLine`

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

```
Dim fruits() As String = {"banana", "apple",
"orange", "pear", "grape", "pineapple"}
Dim wordToFind As String = Console.ReadLine()
Dim found As Boolean = False
For Each fruit in fruits                  (D)
   If fruit = wordToFind Then             (E)
      found = True                        (Part of F)
      Exit For                            (Part of F)
   End If
Next
If found = True Then                      (Part of D)
   Console.WriteLine("True")              (Part of G)
Else
   Console.WriteLine("False")             (Part of G)
End If
```

I. Indentation in VB.NET
A. `Write` in place of `WriteLine`

Question	Part	Marking guidance	Total marks
03	3	**Mark is for AO2 (apply)** The list / array `fruits` is not ordered / sorted;	1

Question	Part	Marking guidance	Total marks
03	**4**	**3 marks for AO2 (apply)** **1 mark for each** error corrected. **Maximum 2 marks** if any logic errors **I.** syntax and other minor errors while rewriting, eg spelling, missing colons, brackets. **I.** other terms for `SUBROUTINE` as long as they make sense. **Line 1** `SUBROUTINE diffCurrencies(x)` ; **Line 6** `FOR i ← 7 TO 0 STEP -1` ;;	3

Question	Part	Marking guidance	Total marks
04		**3 marks for AO1 (understanding)** **Maximum 3 marks** from the following: • start at the beginning/end of a list/array; • iterates sequentially through the list; • compare the contents of **each** position with the data being searched; • if it matches, the item has been found (and the search ends); • if the end/beginning of the list/array is reached without finding the search term then item is not in the list/array; **Max 2 marks** if any errors such as referring to binary search or having the items in order	3