0 1 The subroutine in **Figure 3** is used to authenticate a username and password combination.

- Array indexing starts at 0.
- Line numbers are included but are not part of the algorithm.

Figure 3

```
1     SUBROUTINE Authenticate(user, pass)
2        us ← ['dave', 'alice', 'bob']
3        ps ← ['abf32', 'woof2006', '!@34E$']
4        z ← 0
5        correct ← false
6        WHILE z < 3
7           IF user = us[z] THEN
8              IF pass = ps[z] THEN
9                 correct ← true
10             ENDIF
11          ENDIF
12          z ← z + 1
13       ENDWHILE
14       RETURN correct
15    ENDSUBROUTINE
```

0 1 . 1 Complete the trace table for the following subroutine call:

```
Authenticate('alice', 'woof2006')
```

[3 marks]

z	correct

0 1 . 2 State the value that is returned by the following subroutine call:

```
Authenticate('bob', 'abf32')
```

[1 mark]

0 1 . 3 Lines 7 and 8 in **Figure 3** could be replaced with a single line. Shade **one** lozenge to show which of the following corresponds to the correct new line.

[1 mark]

A `IF user = us[z] OR pass = ps[z] THEN` ⬭

B `IF user = us[z] AND pass = ps[z] THEN` ⬭

C `IF NOT (user = us[z] AND pass = ps[z]) THEN` ⬭

0 1 . 4 A programmer implements the subroutine shown in **Figure 3**. He replaces line 9 with

```
RETURN true
```

He also replaces line 14 with

```
RETURN false
```

Explain how the programmer has made the subroutine more efficient.

[2 marks]

0 2 The algorithms shown in **Figure 4** and **Figure 5** both have the same purpose.

The operator `LEFTSHIFT` performs a binary shift to the left by the number indicated.

For example, `6 LEFTSHIFT 1` will left shift the number `6` by one place, which has the effect of multiplying the number `6` by two giving a result of 12

Figure 4

```
result ← number LEFTSHIFT 2
result ← result - number
```

Figure 5

```
result ← 0
FOR x ← 1 TO 3
   result ← result + number
ENDFOR
```

0 2 . 1 Complete the trace table for the algorithm shown in **Figure 4** when the initial value of `number` is 4

You may not need to use all rows of the trace table.

[2 marks]

result

0 2 . 2 Complete the trace table for the algorithm shown in **Figure 5** when the initial value of `number` is 4

You may not need to use all rows of the trace table.

[2 marks]

x	result

0 2 . 3 The algorithms in **Figure 4** and **Figure 5** have the same purpose.

State this purpose.

[1 mark]

0 2 . 4 Explain why the algorithm shown in **Figure 4** can be considered to be a more efficient algorithm than the algorithm shown in **Figure 5**.

[1 mark]

Turn over for the next question

0 3 The two C# programs in **Figure 5** output the value that is equivalent to adding together the integers between 1 and an integer entered by the user.

For example, if the user entered the integer `5`, both programs would output `15`

Figure 5

Program A

```
Console.Write("Enter a number: ");
int num = Convert.ToInt32(Console.ReadLine());
int total = 0;
for (int i = 1; i < num + 1; i++) {
   total = total + i; }
Console.WriteLine(total);
```

Program B

```
Console.Write("Enter a number: ");
int num1 = Convert.ToInt32(Console.ReadLine());
int num2 = num1 + 1;
num2 = num1 * num2;
num2 = num2 / 2;
Console.WriteLine(num2);
```

0 3 . 1 Shade **one** lozenge to indicate which of the statements is true about the programs in **Figure 5**.

[1 mark]

A Both programs are equally efficient. ⬭

B Program A is more efficient than Program B. ⬭

C Program B is more efficient than Program A. ⬭

0 3 . 2 Justify your answer for Question **03.1**.

[2 marks]