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

```
SUBROUTINE Authenticate(user, pass)
   us ← ['dave', 'alice', 'bob']
   ps ← ['abf32', 'woof2006', '!@34E$']
   z ← 0
   correct ← false
   WHILE z < 3
      IF user = us[z] THEN
         IF pass = ps[z] THEN
            correct ← true
         ENDIF
      ENDIF
      z ← z + 1
   ENDWHILE
   RETURN correct
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 algorithm shown in **Figure 3** is used to check if the start of an instruction for a particular assembly language is valid.

The string representation of the assembly language instruction is stored in the variable `instr`

Characters in the string are indexed starting at zero. For example `instr[2]` would access the third character of the string stored in the variable `instr`

Figure 3

```
code ← ''
i ← 0
WHILE instr[i] ≠ ':' AND i < 4
   code ← code + instr[i]
   i ← i + 1
ENDWHILE
valid ← False
IF code = 'ADD' OR code = 'SUB' OR code = 'HALT' THEN
   valid ← True
ENDIF
```

0 2 . 1 Shade **one** lozenge to show the most appropriate data type of the variable `i` in the algorithm in **Figure 3**.

[1 mark]

A Character ⬭

B Integer ⬭

C Real ⬭

D String ⬭

0 2 . 2 State the data type of the variable `valid` in the algorithm in **Figure 3**.

[1 mark]

____

0 2 . 3 State the final value of the variable `valid` in the algorithm in **Figure 3** for the following different starting values of `instr`

[3 marks]

Value of `instr`	**Final value of** `valid`
`ADD R0, R1`	
`ADD: R0, R1`	
`HALT`	

0 2 . 4 State what an assembly language program must be translated into before it can be executed by a computer.

[1 mark]

0 2 . 5 State **two** reasons why a programmer, who can program in both high-level and low-level languages, would usually choose to develop in a high-level language rather than a low-level language.

[2 marks]

Reason 1

Reason 2

0 2 . 6 Develop an algorithm, using either pseudo-code **or** a flowchart, that:

- initialises a variable called `regValid` to `False`
- sets a variable called `regValid` to `True` if the string contained in the variable `reg` is an uppercase `R` followed by the character representation of a single numeric digit.

Examples:
- if the value of `reg` is `R0` or `R9` then `regValid` should be `True`
- if the value of `reg` is `r6` or `Rh` then `regValid` should be `False`

You may wish to use the subroutine `isDigit(ch)` in your answer. The subroutine `isDigit` returns `True` if the character parameter `ch` is a string representation of a digit and `False` otherwise.

[3 marks]

0 3 **Figure 3** shows an algorithm, represented using pseudo-code, that calculates the delivery cost for an order from a takeaway company.

Figure 3

```
orderTotal ← USERINPUT
deliveryDistance ← USERINPUT
deliveryCost ← 0.0
messageOne ← "Minimum spend not met"
messageTwo ← "Delivery not possible"
IF deliveryDistance ≤ 5 AND orderTotal > 0.0 THEN
   IF orderTotal > 50.0 THEN
      deliveryCost ← 1.5
      OUTPUT deliveryCost
   ELSE IF orderTotal > 25.0 THEN
      deliveryCost ← (orderTotal / 10) * 2
      OUTPUT deliveryCost
   ELSE
      OUTPUT messageOne
   ENDIF
ELSE
   OUTPUT messageTwo
ENDIF
```

0 3 . 1 Using **Figure 3**, complete the table.

[2 marks]

Input value of `orderTotal`	**Input value of** `deliveryDistance`	**Output**
55.5	2	
35.0	5	

0 3 . 2 State how many possible values the result of the comparison `deliveryDistance ≤ 5` could have in the algorithm shown in **Figure 3**.

[1 mark]

---

0 3 . 3 State the most suitable data type for the following variables used in **Figure 3**.

[2 marks]

Variable identifier	**Data type**
`deliveryCost`	
`messageOne`	

0 3 . 4 State **one** other common data type that you have **not** given in your answer to Question **02.3**.

[1 mark]

---

0 4 **Figure 2** shows an algorithm, represented using pseudo-code.

- Line numbers are included but are not part of the algorithm.

Figure 2

```
1    num ← USERINPUT
2    IF NOT(num > 1) OR num > 20 THEN
3       OUTPUT "False"
4    ELSEIF num > 1 AND num < 15 THEN
5       OUTPUT "Almost"
6    ELSEIF num MOD 5 = 0 THEN
7       OUTPUT "True"
8    ELSE
9       OUTPUT "Unknown"
10   ENDIF
```

The modulus operator is used to calculate the remainder after dividing one integer by another.

For example:
- `14 MOD 3` evaluates to `2`
- `24 MOD 5` evaluates to `4`

0 4 . 1 Where is a relational operator **first** used in the algorithm in **Figure 2**?

Shade **one** lozenge.

[1 mark]

A Line number 1 ⬭

B Line number 2 ⬭

C Line number 3 ⬭

D Line number 6 ⬭

0 4 . 2 In the algorithm in **Figure 2**, what will be the output when the user input is `5`?

Shade **one** lozenge.

[1 mark]

A `Almost` ⬭

B `False` ⬭

C `True` ⬭

D `Unknown` ⬭

0 4 . 3 Which value input by the user would result in `True` being output by the algorithm in **Figure 2**?

Shade **one** lozenge.

[1 mark]

A `-1` ⬭

B `10` ⬭

C `20` ⬭

D `21` ⬭

0 4 . 4 Rewrite **line 2** from the algorithm in **Figure 2** **without** using the `NOT` operator.

The algorithm must still have the same functionality.

[1 mark]

____

____

____

0 4 . 5 A user inputs a value into the algorithm in **Figure 2**.

State **one** value that the user could input that would result in an output of `Unknown`

[1 mark]

____