0 1 The three examples of code shown in **Figure 2** are all equivalent to one another.

Figure 2

Example 1	Example 2	Example 3

a ← 4
b ← 3
IF a = b THEN
   c ← a + b
ENDIF
	MOV R0, #4
	 MOV R1, #3
	 CMP R0, R1
	 BNE end
	 ADD R2, R0, R1
	end:
	 HLT
		1001 0000 0100 0000
		1001 0001 0011 0000
		0100 0000 0001 0000
		1010 0101 0000 0000
		1100 0010 0000 0001
		1111 0000 0000 0000

0 1 . 1 Shade **one** lozenge to show the statement that is true about **Figure 2**.

[1 mark]

A None of the examples of code is in a low-level language. ⬭

B Only one of the examples of code is in a low-level language. ⬭

C Only two of the examples of code are in low-level languages. ⬭

D All three of the examples of code are in low-level languages. ⬭

0 1 . 2 Explain why a developer, who is good at both low-level and high-level programming, would normally use high-level languages when writing programs.

[4 marks]

___

___

___

___

___

___

___

___

___

___

0 1 . 3 Statements **A** and **B** refer to two different types of program translator.

Statement A: This type of translator can convert a high-level language program into machine code. The source code is analysed fully during the translation process. The result of this translation can be saved, meaning the translation process does not need to be repeated.

Statement B: This type of translator was used to convert the code in **Example 2** to the code in **Example 3** in **Figure 2**.

State the type of program translators referred to in statements **A** and **B**.

[2 marks]

Statement **A**: ______________________________

____

Statement **B**: ______________________________

____

Turn over for the next question

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 Programming languages can be classified as low-level or high-level.

Shade **two** lozenges to show the statements that are true about code written using a low-level language instead of a high-level language.

[2 marks]

A	The code more closely resembles English.	⬭
B	The code is easier to write.	⬭
C	The code is not translated using a compiler.	⬭
D	The code is quicker to write.	⬭
E	The code can directly manipulate computer registers.	⬭
F	The code never needs to be translated before being executed.	⬭

0 4 Assemblers and interpreters are two types of program translator.

0 4 . 1 State the purpose of an assembler.

[1 mark]

0 4 . 2 Explain how an interpreter works.

[4 marks]

Turn over for the next question

0 5 . 1 **Figure 4** shows three programs (**A**, **B**, **C**) that add two numbers and output the result. The programs are written in different programming languages.

Figure 4

A	**B**	**C**

`x = 14`
`y = 3`
`z = x + y`
`OUTPUT(z)`
	`LDR R0, #14`
	`LDR R1, #3`
	`ADD R2, R0, R1`
	`STR R2, 63`
	`OUT R2`
		`0000 00001110`
		`0001 00000011`
		`0110 00010000`
		`1010 10111111`
		`1110 00000000`

Identify the type of programming language used for each program shown in **Figure 4** by writing **A**, **B** or **C** in the correct row of **Table 2**.

You **must** only use each letter once.

[2 marks]

Table 2

	A, B or C
Assembly language	
High-level language	
Machine code	

0 5 . 2 State **one** advantage of writing programs in assembly language instead of a high-level language.

[1 mark]

____

____

0 5 . 3 Shade **one** lozenge to show which statement is true about program translators.

[1 mark]

A A compiler translates all the original program code before execution. ⬭

B Compiled code still needs the original program code to execute. ⬭

C Compiled code executes more slowly than code that is being interpreted. ⬭

D Interpreters generate machine code directly. ⬭

Turn over for the next question

0 6 The three examples of code shown in **Figure 4** are all equivalent to one another.

Figure 4

Example 1	Example 2	Example 3

a ← 4
b ← 3
IF a = b THEN
   c ← a + b
ENDIF
	MOV R0, #4
	MOV R1, #3
	CMP R0, R1
	BNE end
	ADD R2, R0, R1
	end:
	HLT
		1001 0000 0100 0000
		1001 0001 0011 0000
		0100 0000 0001 0000
		1010 0101 0000 0000
		1100 0010 0000 0001
		1111 0000 0000 0000

0 6 . 1 Shade **one** lozenge to show the statement that is true about **Figure 4**.

[1 mark]

A None of the examples of code is in a low-level language. ⬭

B Only one of the examples of code is in a low-level language. ⬭

C Only two of the examples of code are in low-level languages. ⬭

D All three of the examples of code are in low-level languages. ⬭

0 6 . 2 Explain why a developer, who is good at both low-level and high-level programming, would normally use high-level languages when writing programs.

[4 marks]

___

___

___

___

___

___

___

___

___

___

0 6 . 3 Statements **A** and **B** refer to two different types of program translator.

Statement A: This type of translator can convert a high-level language program into machine code. The source code is analysed fully during the translation process. The result of this translation can be saved, meaning the translation process does not need to be repeated.

Statement B: This type of translator was used to convert the code in **Example 2** to the code in **Example 3** in **Figure 4**.

State the type of program translators referred to in statements **A** and **B**.

[2 marks]

Statement **A**: ______________________________

____

Statement **B**: ______________________________

____

0 7 Which **two** statements are true about machine code?

Shade **two** lozenges.

[2 marks]

A Machine code is directly executed by the processor. ⬭

B Machine code is easily understood by humans. ⬭

C Machine code is shorter than high-level code. ⬭

D Machine code is similar to English. ⬭

E Machine code is specific to a family of processors. ⬭

F Machine code is translated using a compiler. ⬭

0 8 Describe **three** differences between high-level programming languages and low-level programming languages.

[3 marks]

1 ____

____

____

____

2 ____

____

____

____

3 ____

____

____

____

0 9 Describe how compilers and interpreters operate.

[6 marks]