Mark scheme

Question			Answer/Indicative content	Marks	Guidance
1	a		1 mark each to max 2 • (machine code) does not need to be translated / compiled / interpreted • Direct control of **hardware / memory** • Faster execution time • Code can be optimised / shorter code / use less memory • Can program for specific hardware • **Assembly language** is fast to translate.	2 (AO1)	"More efficient" by itself is TV. Mark first answer on each line. BP6 relates to Assembly language being a one-to-one direct mapping to machine code. **Examiner's Comments** These questions assessed understanding of low-level languages and the use of compilers to translate from high-level code. Many candidates were able to do this well and clearly understood the need for high-level languages to be translated into machine code before the processor can execute this. The questions, however, did not ask just for this but specifically asked for reasons and benefits. Where candidates left their response as simply a discussion of the difference between high-level and low-level code or between translators and compilers, it was difficult for examiners to award marks. Better responses were able to give clear reasons for the use of low-level code (such as direct control of hardware and faster execution of code) and the benefits of using a compiler instead of an interpreter (such as being able to distribute an executable file with no access to source code and not needing to translate code again once this has been done).
	b		1 mark each to max 3 • Can produce an **executable file** • program/code **runs/executes** faster (than interpreted version) • end users do not need translator • Can be run **again/multiple times** without re-	3 (AO1)	Allow in reverse (e.g. "interpreter translates every time") Do not allow "no access to source code" unless clearly talking about end user. Allow if in context of distribution. Do not allow descriptions of how a compiler translates (e.g. "translates whole code in one go") "Faster / quicker" by itself is TV

			translating / only needs to translate **once** • **End users** have no access to **source code** / distributed **with no source code**… • …cannot steal/copy/modify code/program • Shows **all/multiple** errors / shows errors **at the end** (of compilation) / creates **error file** • Compiler can optimise the code		**Examiner's Comments** These questions assessed understanding of low-level languages and the use of compilers to translate from high-level code. Many candidates were able to do this well and clearly understood the need for high-level languages to be translated into machine code before the processor can execute this. The questions, however, did not ask just for this but specifically asked for reasons and benefits. Where candidates left their response as simply a discussion of the difference between high-level and low-level code or between translators and compilers, it was difficult for examiners to award marks. Better responses were able to give clear reasons for the use of low-level code (such as direct control of hardware and faster execution of code) and the benefits of using a compiler instead of an interpreter (such as being able to distribute an executable file with no access to source code and not needing to translate code again once this has been done).
			Total	**5**	

2	a		One mark per bullet point
			
			• Ask the user for **two** inputs and store/use these
			• Open pilots.txt (for write/append)
			• Write **both** inputs **to opened text file**
			• Close text file
				4 (AO3 2b)	Award BP4 for implicit closing of file (e.g. using `with…` in Python)
					
					Example answer :
					`dronepilotData = open("pilots.txt")`
					`pilotCode = input("enter code")`
					`dob = input("enter date of birth")`
					`dronepilotData.writeLine(pilotCode, dob)`
					`dronepilotData.close()`
					
					Allow data to be written either by simply writing both values or by concatenating into one string with a separating comma.

	b		One mark per bullet point to max 6
			
			• Function header `pilotValid()` with (at least one) parameter
			• Checks array element [i,0] for **each** item…
				6 (AO3 2b)	Example answer :
					`function pilotValid(pilotCode)`
					`  total = 0`
					`  status = ""`
					`  for i = 0 to 5`
					`  if journeys[i,0] == pilotCode then`
					`  temp =`

			• …calculates total
			• …casting to **float / real**
			• Checks if 9 hours or fewer
			• **Returns** a value
			• …returns "warning" and "valid" correctly
					`float(journeys[i,1])`
					`  total = total + temp`
					`  endif`
					`next i`
					`if total > 9 then`
					`  status = "warning"`
					`else`
					`  status = "valid"`
					`endif`
					`return status`
					`endfunction`
					
					Do not allow casting to integer for BP4, data shows some journeys have decimal places.
					
					Allow FT for BP5 if attempt made at calculation.
					Allow FT for BP7 if value is output instead of returned.

			Total	**10**	
3	a		Mark in pairs, 1 mark for advantage, 1 mark for expansion. e.g. • Easier / quicker to create… • …. Language closer to English / uses keywords • Provides higher level of abstraction from the underlying processor… • … don't have to (usually) deal with allocating memory • Portable / processor independent… • … can write once for many processor types (e.g. mobile)	4 (AO1 1b, AO2 1b)	Accept other reasonable advantages and descriptions
	b		1 mark per row **Statement** / **Compiler** / **Interpreter** / **Both**	3 (AO1 1a)	

			<table><tr><td>Translates high-level code to low-level instructions.</td><td></td><td></td><td>✓</td></tr><tr><td>Produces an executable file.</td><td>✓</td><td></td><td></td></tr><tr><td>Program needs to be translated every time it is run.</td><td></td><td>✓</td><td></td></tr></table>		
			Total	**7**	

4		i	• Attempt at using selection
			• Calculates pay correctly for pilots with **less than** 2 years experience
			• Calculates pay correctly for pilots with 2 to 5 years experience. **Must include both 2 and 5**.
			• Calculates pay correctly for pilots with more than 5 years experience
				4 (AO3 2b)	Example answer :
					
					`experience = input("Enter years of experience")`
					`miles = input("Enter miles flown")`
					
					`totalPay = 0`
					`if exp <2 then`
					`  totalPay = 120+(0.45*miles)`
					`elseif exp <=5 then`
					`  totalPay = 150+(0.65*miles)`
					`else`
					`  totalPay = 180+(0.85*miles)`
					`end if`
					`print(totalPay)`
					
					FT for BP4 only where a reasonable attempt at calculating pay has been made.

		ii	• totalPay • (experience, miles) / (miles, experience)	2 (AO3 2c)	`totalPay = calculatePay(experience, miles)`
			Total	**6**	
5			1 mark per row	4 (AO1 1b)	No mark if more than 1 tick for that row. Allow other indications of choice (e.g. cross) as

<table>
<tr>
<td></td><td></td><td></td>
<td>
<table>
<tr><th>Statement</th><th>Low-level</th><th>High-level</th></tr>
<tr><td>The same language can be used on computers that use different hardware</td><td></td><td>✓</td></tr>
<tr><td>It allows the user to directly manipulate memory</td><td>✓</td><td></td></tr>
<tr><td>It allows the user to write English-like words</td><td></td><td>✓</td></tr>
<tr><td>It always needs to be translated into object code or machine code</td><td></td><td>✓</td></tr>
</table>
</td>
<td></td>
<td>long as clear.

<u>Examiner's Comments</u>

This question was answered well by many candidates. The strongest responses showed a good understanding of the difference between low-level and high-level languages. Incorrect responses tended to be on the first row related to portability. A high-level language such as Python is portable, with translators available for many different types of processors. A low-level language is specific to one type of processor.</td>
</tr>
<tr><td></td><td></td><td></td><td>Total</td><td>4</td><td></td></tr>
<tr><td>6</td><td></td><td>i</td><td>• Multiplication
• Division</td><td>2
(AO1 1a)</td><td>Accept other correct answers that mean the same Accept floor / integer division / division with no remainder (Python v2.x)

<u>Examiner's Comments</u>

This question was answered well by the majority of candidates.</td></tr>
<tr><td></td><td></td><td>ii</td><td>• high-level
• stops / crashes
• no
• executable
• without</td><td>5
(AO1 1b)
(AO2 1b)</td><td>Ignore spelling errors.

<u>Examiner's Comments</u>

This question was answered well by the majority of candidates.</td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>7</td><td></td></tr>
<tr><td>7</td><td>a</td><td></td><td>• To convert it to binary/machine code
• The processor can only understand machine code</td><td>1
(AO1 1a)</td><td>Maximum 1 mark</td></tr>
<tr><td></td><td>b</td><td></td><td>• Compiler translates all the code in one go…
• …whereas an interpreter translates one line at a time
• Compiler creates an executable…</td><td>4
(AO1 1b)</td><td>1 mark to be awarded for the correct identification and one for a valid description up to a maximum of 4 marks.
No more than 2 marks for answers relating only to interpreters and no more than 2 marks for answers only relating to compilers.</td></tr>
</table>

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

- …whereas an interpreter does not/executes one line at a time
- Compiler reports errors at the end…
- …whereas an interpreter stops when it finds an error

</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td>Total</td><td>5</td><td></td></tr>
<tr><td>8</td><td>a</td><td></td><td>

One mark per row

<table>
<tr><th>Statement</th><th>High-level language</th><th>Low-level language</th></tr>
<tr><td>Uses English-like keywords such as <code>print</code> and <code>while</code></td><td>✓</td><td></td></tr>
<tr><td>Must be translated before the processor can execute code</td><td>✓</td><td></td></tr>
<tr><td>Code written is portable between different processors</td><td>✓</td><td></td></tr>
<tr><td>Requires the programmer to understand the processor's registers and structure</td><td></td><td>✓</td></tr>
</table>

</td><td>4</td><td>Accept other markings that indicate a choice has been made (e.g. a cross, etc)</td></tr>
<tr><td></td><td>b</td><td></td><td>

1 mark per bullet, max 4

e.g.

- Editor
- …to enable program code to be entered / edited
- Error diagnostics / debugger

</td><td>4</td><td>Allow other tools available in an IDE with suitable expansion (e.g. breakpoints, watch window, stepping, pretty printing, etc)</td></tr>
</table>

			• …to display information about errors / location of errors / suggest solutions • Run-time environment • …to enable program to be run / to check for runtime errors / test the program		
			Total	**8**	
9			• Error diagnostics (any example) • Run-time environment • Editor (any feature such as auto-correct, auto-indent) • Translator • Version control • Break point • Stepping	2 (AO1 1a)	1 mark per bullet to a maximum of 2 marks. Only 1 example per bullet, e.g. auto-correct and auto-indent would only gain 1 mark.
			Total	**2**	