MODULE P LANGUAGES AND IDES · P1

High-level and low-level languages: mark scheme

20 marks · OCR J277 spec 2.5.1 (characteristics and purpose of different levels of programming language: high-level languages, low-level languages; the differences between high-level and low-level programming languages)

How to use. 1 mark per bullet up to the maximum shown. **/** separates alternative words; **//** separates alternative answers for the same mark. Words in (brackets) are not needed for the mark. **...** at the start of a bullet means it builds on the bullet before it. Credit any other valid answer that shows the same understanding. In questions 3 and 4, answers should apply to the scenario given.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.5.1	A	**1**	Correct answer only. More than one extract given = 0.
1(b) J277 2.5.1	C	**1**	Correct answer only. C is machine code (binary), the CPU's own instructions. Do not accept B (assembly code still has to be translated).
1(c) J277 2.5.1	Any one from: A language that is close to / the same as the instructions the CPU / processor carries out // close to machine code A language written for (the instruction set of) a specific / one type of processor Machine code / binary instructions or assembly language (with each instruction matching one machine code instruction)	**1**	Max 1. Accept "a language the computer / CPU understands directly". Do not accept "a language that is hard to read" or "a basic / simple language" alone (NE).
2(a) J277 2.5.1	Row 1: **False** ... each statement is translated into <u>many</u> / one or more / several machine code instructions Row 2: **True** Row 3: **False** ... it will only run on that type / family of processor // it will not run on a different type of processor // it is not portable Row 4: **False** ... code written in a <u>low-level</u> language is closer to the CPU's own instructions // high-level code is further from the CPU's instructions / closer to English	**4**	1 mark per row. A false row needs the False tick **and** a correct rewrite for the mark. Two ticks in a row = 0 for that row. Row 3: accept "will run on the same type of processor only"; do not accept "will run on no other computer" (it runs on other computers with the same type of processor).

QUESTION	ANSWER	MARKS	GUIDANCE
3(a) J277 2.5.1	2 marks per reason: 1 mark for the reason, 1 mark for a linked explanation in context, e.g.: Low-level code can directly control / manipulate the hardware / specific memory locations … … so the program can read the soil sensor / switch the radio on and off directly Low-level code can use less memory / be more compact … … the sensor only has a few kilobytes of memory, so the program must fit in it Low-level code can run faster / more efficiently / needs fewer instructions to be executed … … so the processor spends less time working // the radio is on for less time … which saves battery power // the reading is taken and sent quickly Code can be written / optimised for this specific processor … … so it makes the best use of the sensor's simple processor	**4**	Max 4: max 2 per reason. The explanation mark must link to its reason. Generic answers with no link to the sensor: max 3. Do not accept "faster" or "better" on its own (NE); the answer must say what is faster, e.g. the code runs faster. Do not accept "easier to write" (wrong).
3(b) J277 2.5.1	1 mark for a drawback, 1 mark for a linked description, e.g.: Low-level code is hard to read / write / understand (uses binary or mnemonics, not English-like keywords) … … so the program takes longer to write // mistakes are more likely // the programmers need specialist knowledge of the processor Errors are hard to find / fix / debug … … so testing / fixing the program takes longer The code only works on this type of processor (not portable) … … so if the company changes to a sensor with a different processor, this part must be rewritten Many low-level instructions are needed for each task … … so the program is longer / takes more time to write	**2**	Max 2. Only one drawback is needed; if more are given, mark the best one. Do not accept "slower to run" or "uses more memory" (wrong). Do not accept "it is harder" with no detail.
4(a) J277 2.5.1	Any two from: High-level code is machine independent / portable // is not written for one type of processor The same (source) code can be translated into the machine code of each type of processor // the code is translated (compiled / interpreted) for each device So the programmers only need to write the app once // do not need to write a separate version for each type of device / processor	**2**	Max 2. Accept "a translator for each type of processor turns the same code into the right machine code". Do not accept "it works on all computers" with no reason (NE).

QUESTION	ANSWER	MARKS	GUIDANCE
4(b) J277 2.5.1	Any two from: Easier / quicker to read / write / understand // uses English-like keywords / meaningful names Easier to debug // errors are easier to find and fix One statement does the work of many machine code instructions // fewer lines of code needed // quicker to write the app Easier to maintain / update // other programmers can understand the code more easily The programmer does not need to know about the processor / hardware / memory locations Has built-in functions / libraries that can be used	**2**	Max 2. Do not credit portability / running on different devices again (asked for "other" benefits). Do not accept "faster" or "easier" alone (NE). Do not accept "runs faster" (wrong). Two answers with the same idea score once.
4(c) J277 2.5.1	Any one from: It must be translated (into machine code) before the CPU can execute it // needs a translator The translated code may run more slowly / less efficiently than (well-written) low-level code The translated code may use more memory than low-level code It cannot directly control / manipulate the hardware / specific memory locations The programmer has less control over exactly which machine code instructions are used	**1**	Max 1. Must be a comparison or a clear drawback: do not accept "it is slow" alone (NE). Do not accept "harder to read" or "not portable" (wrong way round).
5(a) J277 2.5.1	1 mark for a feature of high-level (or low-level) code, 1 mark for why this makes errors easier (or harder) to find, e.g.: High-level code uses English-like keywords / meaningful variable names // low-level code uses binary / mnemonics / memory addresses … … so the programmer can read and follow the logic / see what each line does more easily // can more easily spot the line that is wrong High-level programs have fewer lines (one statement does many low-level instructions) … … so there is less code to search through / trace High-level code uses structures such as if / while / named variables instead of jumps and memory addresses … … so the order the program runs in is easier to follow	**2**	Max 2. The second mark must explain the first. Do not accept "it is easier to debug" alone (repeats the question). Credit any other valid feature with a linked reason.

Total: 20 marks