OCR GCSE (J277)

Module P: Languages and IDEs

Workbooklet mark scheme · 3 parts · 61 marks

P2 is IGCSE (Cambridge) only and not on J277.

Contents

Part	Topic	Marks
P1	High-level and low-level languages: mark scheme	20
P3	Translators: compilers and interpreters: mark scheme	21
P4	IDE tools and how they help a programmer: mark scheme	20
	Total	**61**

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).
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.

QUESTION	ANSWER	MARKS	GUIDANCE
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

MODULE P LANGUAGES AND IDES · P3

Translators: compilers and interpreters: mark scheme

21 marks · OCR J277 spec 2.5.1 (purpose of translators; characteristics, benefits and drawbacks of a compiler and an interpreter)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. "..." means the mark depends on the point before it. Credit any other valid answer at J277 level. Assemblers are not required for J277: give no credit for them, but do not penalise them if the rest is correct.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.5.1	To convert / translate source code / program code / high-level (or low-level) code into machine code / binary / object code	**1**	Accept: so the code can be executed / run by the CPU / computer. Do not accept "converts it into a language the computer understands" with no idea of machine code / execution, or "translates English into code".
1(b) J277 2.5.1	The CPU can only execute / understand machine code / binary (instructions) High-level code uses English-like statements / is not machine code ... ... so (the CPU) cannot execute it directly The translator converts the high-level code into machine code (so it can be executed)	**2**	Max 2. Credit any other valid reason why the high-level code cannot be run as it is.
2(a) J277 2.5.1	Translates the whole program before any of it is run: Compiler Must be on the computer every time the program is run: Interpreter Converts source code written in a high-level language into machine code: Both Produces a file that can be run without the original source code: Compiler Runs each line straight after translating it: Interpreter	**5**	1 mark per correct row. More than one tick in a row: 0 for that row.

QUESTION	ANSWER	MARKS	GUIDANCE
3(a) J277 2.5.1	Identification: compiler (1) Justification, max 2: It produces an executable file ... ... which can be copied onto all (400) machines / run on the machines The source code is not given out / the machines only get the executable ... ... so other businesses cannot see / copy / change the source code The machines do not need a translator / compiler installed to run the program The program is translated only once / does not need translating each time it runs (on the machines) The compiled program runs / executes faster (on the machines) than an interpreted version	**3**	Justification marks only if compiler is identified. "It is faster" with no comparison or reason: NE. Accept "runs faster than if it were interpreted". Do not accept "it is more secure" alone.
3(b) J277 2.5.1	The (whole) program / source code must be compiled again / recompiled (after the change) A new executable file is produced ... ... which must be copied / installed onto (each of) the ticket machines	**2**	Max 2. Do not accept "only the changed line is compiled".
4(a) J277 2.5.1	Lines 01 and 02 are translated and run / "Stock check started" is output and the number of boxes is input (Translation and) running stops at line 03 / when the first error is reached The error on line 03 is reported (with its line number) The error on line 06 is not reported / found (yet) / is only found once line 03 is fixed and the program runs again	**3**	Max 3. Do not accept "both errors are reported" or "nothing is output".
4(b) J277 2.5.1	The whole program is translated / checked before any of it is run Both errors / the errors on lines 03 and 06 are reported ... together / in a list / at the end of translation No executable file is produced The program does not run / nothing is output / no input is asked for	**3**	Max 3. Do not accept "it stops at line 03".
5(a) J277 2.5.1	An interpreter translates each line while the program is running / every time the program is run / each time a line in a loop is run A compiled program has already been translated into machine code (before it is run) ... ... so the CPU only has to execute it / no translating is done while it runs	**2**	Max 2. Accept the reverse argument from either side. Do not accept "compilers are faster" alone.

Total: 21 marks

MODULE P LANGUAGES AND IDES · P4

IDE tools and how they help a programmer: mark scheme

20 marks · OCR J277 spec 2.5.2 (common tools and facilities in an IDE: editors, error diagnostics, run-time environment, translators; how each can be used to help a programmer develop a program) and 2.5.1 (characteristics of a compiler and an interpreter)

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. Credit any other valid answer that shows the same understanding. A description must add more than repeating the name of the tool or feature (e.g. "line numbers number the lines" is NE).

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.5.2	1 mark for each correct row: Lets the programmer type in the code and change it: Editor Converts the high-level code into machine code: Translator Displays the fare that the program outputs while it is tested: Run-time environment Displays a message giving the line number of a mistake and what it is: Error diagnostics Automatically indents the lines inside an `if` statement: Editor	**5**	2+ ticks in a row = 0 marks for that row. Row 4: the translator may detect the error, but the tool that reports / displays it is error diagnostics.
2(a) J277 2.5.2	1 mark each to max 2: Error messages give a line number (e.g. Line 07) ... ... so the programmer can find / go straight to the line with the error (quickly) The programmer can refer to / find a particular line (e.g. when stepping through / setting a breakpoint / explaining the code to someone else) Easier to see / keep track of where you are in a long program	**2**	"Numbers each line" alone is NE (repeats the feature). "Helps find errors" alone is NE: needs the link to locating the line given in an error message or a specific line.
2(b) J277 2.5.2	1 mark each to max 2: Makes the code easier to read / understand Keywords stand out from identifiers / variables / strings // the structure (e.g. where an `if` starts) is easier to see A misspelt keyword is not coloured / shows up ... ... so (syntax) errors can be spotted (and corrected) before the program is run / more quickly	**2**	"Keywords are coloured" alone is NE (repeats the feature). "Looks nicer" = 0.

QUESTION	ANSWER	MARKS	GUIDANCE
3(a) J277 2.5.2	1 mark each to max 2: Shows the line number / location of the error (line 07) ... ... so the programmer knows where to look // can go straight to the line Describes / explains the error / what is wrong (an `endif` is missing) ... ... so the programmer knows what to change / how to fix it (add `endif` before the `print` line) (Some IDEs) suggest a fix Reports the error before / without the program having to be run // reports all the syntax errors found	**2**	"Finds errors" / "tells you there is an error" alone is NE. Answers must go beyond saying an error exists.
4(a) J277 2.5.2	1 mark each to max 4: Set a breakpoint (e.g. at line 05) ... ... the program stops / pauses at that line (so the values can be checked) Step through the program / execute the code one line at a time ... ... to see which lines run / the path taken through the code // to see that line 06 is skipped / not executed when age is 65 Use a variable watch / watch window ... ... to see the value of `age` / `fare` as each line runs // to see that `fare` stays at 12.0 Run the program in the run-time environment with test data (e.g. 64, 65, 66) and compare the output with the expected result ... ... to narrow down the boundary where it goes wrong (65 gives the full fare, 66 gives half) Identify that the condition on line 05 is the cause // > should be >=	**4**	Max 3 if no tool is named (breakpoint, stepping, watch, debugger, trace, run-time environment). Accept "use the debugger to ..." with a valid description. Do not accept "use the error diagnostics to find the error message": this logic error produces no error message. Allow sensible references to AI assistants that explain code, with a valid description.
5(a) J277 2.5.2	1 mark each to max 2: Lets the program be run / executed inside the IDE (without leaving it / without another program) Shows the output of the program (e.g. the fare) // an output window / console ... ... so the output can be checked / compared against the expected result The programmer can enter test data (e.g. an age) while the program runs Shows (run-time) errors that happen while the program runs // lets the programmer see where the program crashes	**2**	"Runs the program" and "shows the output" are separate marks. "Tests the program" alone is NE.
5(b) J277 2.5.1, 2.5.2	1 mark for the benefit, 1 mark for the expansion, e.g.: Translates and runs the code one line at a time / line by line ... ... (so) it stops at the line where an error is found // the error can be located easily (and fixed) *or* No need to compile / translate the whole program first ... ... so changes can be tested quickly / straight away // part of the program can be run even if later lines are unfinished	**2**	Mark as a linked pair: the expansion must follow the benefit given. Do not accept "runs faster" (interpreted code runs more slowly than compiled code) or "produces an executable" (that is a compiler).

QUESTION	ANSWER	MARKS	GUIDANCE
5(c) J277 2.5.1	Compiler	**1**	Correct answer only. Do not accept "assembler" (not on J277) or "translator" alone.

Total: 20 marks