CAMBRIDGE IGCSE (0478)

Module P: Languages and IDEs

Workbooklet mark scheme · 4 parts · 81 marks

Contents

Part	Topic	Marks
P1	High-level and low-level languages: mark scheme	20
P2	Assembly language and assemblers: mark scheme	20
P3	Translators: compilers and interpreters: mark scheme	20
P4	IDE tools and how they help a programmer: mark scheme	21
	Total	**81**

MODULE P LANGUAGES AND IDES · P1

High-level and low-level languages: mark scheme

20 marks · Cambridge 0478 syllabus 4.2.1 (explain what is meant by a high-level language and a low-level language, including the advantages and disadvantages of each: ease of reading and writing code, ease of debugging code, machine independence, direct manipulation of hardware)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. 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(a), answers should apply to the scenario given.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 4.2.1	Any two from: Uses English-like words / keywords // is close to human language Is easy for a human / programmer to read / write / understand Must be translated (by a compiler / interpreter) into machine code before it can be run // the CPU cannot execute it directly One statement / line can carry out many actions // is translated into many machine code instructions Is machine independent // can be used on different types of processor / computer Cannot directly manipulate the hardware // the programmer does not need to know about the hardware	**2**	max 2. Do not accept "a language that is easy" or "an advanced language" alone.
1(b) 0478 4.2.1	Any two from: Is close to the language / instructions of the processor // close to machine code Is machine code (binary) // assembly language (mnemonics) Is hard for a human / programmer to read / write / understand One instruction / line carries out a single action // each instruction matches one machine code instruction Is machine dependent // only runs on the type of processor it was written for Can directly manipulate the hardware // can directly access specific memory locations / registers	**2**	max 2. Machine code and assembly language given together score one mark. Do not accept "a language that is hard" or "a basic language" alone.
2(a) 0478 4.2.1	A	**1**	More than one extract given scores 0.

QUESTION	ANSWER	MARKS	GUIDANCE

2(b)
0478 4.2.1
	Any two from:
	It uses English-like keywords (e.g. WHILE, OUTPUT) // extract C is in binary / 1s and 0s
	It uses meaningful names / identifiers for data (e.g. Count) // extract C does not show what the data is / uses memory addresses
	It uses familiar mathematical symbols (e.g. <, +)
	Its structure (loop / indentation) is easy to see // the purpose of each line is clear
	Each statement does more work // fewer statements are needed for a task // the programmer does not need to write each machine code instruction
	The programmer does not need to know the processor's instructions / the binary codes
		2	max 2. Two answers with the same idea score once. Do not accept "it is easier to understand" with no reason (repeats the question).

3(a) 0478 4.2.1	Two advantages, one mark for each advantage and one mark for each linked expansion. Any four from: Can directly manipulate the hardware … … so the program can read the (position) sensors / send signals to the motors directly Can directly access specific memory locations / registers … … e.g. where the sensor readings are stored / where the motor signals are sent The code can run faster / more efficiently … … so the arm reacts quickly / in real time to sensor readings // so the arm stops / moves at the right moment The code can use less memory … … so it fits in the limited memory of the arm's processor The code can be written / optimised for the arm's specific processor … … so it uses the processor's own instructions	**4**	max 4, max 2 per advantage. An expansion mark must be linked to its advantage. Answers not linked to the robotic arm: max 3. Do not accept "faster" alone (must say the code runs faster). Do not accept "easier to write / debug" (wrong).
3(b) 0478 4.2.1	One mark for the disadvantage and one mark for a linked description. Any two from: The code is hard to read / write / understand … … so the section takes longer to write // mistakes are more likely // programmers need detailed knowledge of the arm's processor / hardware The code is hard to debug … … so errors take longer to find / correct The code is machine dependent // only runs on this type of processor … … so if the arm's processor is replaced with a different type, the section must be rewritten Many instructions are needed for each task … … so the code is long // it takes longer to write	**2**	max 2. Only one disadvantage is needed; if more are given, mark the best one. Do not accept "runs slower" or "uses more memory" (wrong).

QUESTION	ANSWER	MARKS	GUIDANCE
4(a) 0478 4.2.1	Any two from: The program is not written for one type of processor // can be run on different types of processor (after it is translated) So the same program can be used on the office (desktop) computers and on the tablets (in the vans) The program only needs to be written once // no need to write a separate version for each type of device / processor ... which saves time / cost for the company // makes the program easier to update on all the devices	**2**	max 2. For 2 marks the answer must refer to the different devices / processors in the scenario. Do not accept "it works on any computer" with no explanation.
4(b) 0478 4.2.1	Any one from: Easier to read / write / understand (English-like) Easier to debug // errors are easier to find and correct Fewer lines of code are needed // one statement carries out many actions // quicker to write Easier to maintain / update // other programmers can understand it The programmer does not need to know about the processor / hardware	**1**	max 1. Do not credit machine independence / portability again. Do not accept "faster" or "easier" alone. Do not accept "runs faster" (wrong).
4(c) 0478 4.2.1	One mark for the disadvantage and one mark for a linked explanation. Any two from: Cannot directly manipulate the hardware / access specific memory locations ... ... so it is not suitable for programs that must control devices directly (e.g. device drivers / embedded control) Must be translated (into machine code) before it can be run ... ... so a translator (compiler / interpreter) is needed // translation takes extra time The translated code may be less efficient ... ... so it may run more slowly / use more memory than code written in a low-level language	**2**	max 2. Only one disadvantage is needed. Do not accept "it is slow" alone (must compare with low-level code). Do not accept "harder to read" or "machine dependent" (wrong way round).
5(a) 0478 4.2.1	One mark for a feature and one mark for a linked explanation. Any two from: High-level code uses English-like keywords / meaningful identifiers // low-level code is in binary / mnemonics / uses memory addresses ... ... so the programmer can follow the logic / see what each line does // can more easily spot the line that is wrong High-level programs have fewer lines (one statement does many actions) ... ... so there is less code to check / trace High-level code uses clear structures such as loops and selection (not jumps to memory addresses) ... ... so the order the program runs in is easier to follow	**2**	max 2. Do not accept "it is easier to debug" alone (repeats the question).

Total: 20 marks

MODULE P LANGUAGES AND IDES · P2

Assembly language and assemblers: mark scheme

20 marks · Cambridge 0478 syllabus 4.2.2 (assembly language is a form of low-level language that uses mnemonics; an assembler translates it into machine code), with 4.2.1 (advantages and disadvantages of low-level languages), 4.2.3 (compiler) and 3.1.4 (an instruction set's commands are machine code)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. Credit any other valid answer that shows the same understanding. Candidates do not need to know what any particular mnemonic does.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 4.2.1, 4.2.2	One mark for each correct row: A: High-level language B: Machine code C: Assembly language	**3**	A row with more than one tick scores 0.
1(b) 0478 4.2.2	One mark for the meaning, one mark for the example: A short code / abbreviation (of letters) that represents an instruction / operation / command (used in assembly language instead of binary) LDA // ADD // STO	**2**	Accept "a word that is easy to remember that stands for a machine code instruction". Accept a whole line from program C (e.g. "LDA 6"). Do not accept a number alone (6, 7, 8), a binary pattern from program B, or a word from program A (e.g. if, then).
2(a) 0478 4.2.1, 4.2.2, 4.2.3	One mark for each correct row: Form of low-level language: True A compiler translates assembly language: False Allows direct manipulation of the hardware: True Will run on any other type of processor without being changed: False	**4**	A row with both boxes ticked scores 0.
3(a) 0478 4.2.2, 3.1.4	Any two from: Assembly language uses mnemonics // (text) codes, but machine code is (written in) binary // 1s and 0s Assembly language must be translated (by an assembler) before it can be executed, but machine code can be executed directly by the CPU // without translation Machine code is the CPU's instruction set // the commands the CPU processes, but assembly language represents these commands (using mnemonics) Assembly language is easier (for a person) to read / write / understand / debug than machine code	**2**	max 2. Each point must make the difference clear: both sides stated or clearly implied (e.g. "assembly uses mnemonics instead of binary"). Do not accept "one is high-level and one is low-level" (both are low-level).

QUESTION	ANSWER	MARKS	GUIDANCE
4(a) 0478 4.2.2, 3.1.4	One mark for: An assembler (is the translator needed) Any two from: The CPU can only execute // process machine code // binary instructions The CPU's instruction set is (made of) machine code commands Assembly language uses mnemonics // is not binary, so the CPU cannot execute it directly The assembler converts the mnemonics // assembly language (instructions) into machine code // binary	**3**	max 3. Do not award the translator mark for compiler or interpreter. Do not accept "the computer cannot read it" alone (too vague). Accept "translates it so the CPU can understand it" only with machine code // binary.
5(a) 0478 4.2.1, 4.2.2	One mark for each requirement linked to a feature of assembly language, one mark for the matching expansion (max 2 requirements): **Small memory:** assembly language // low-level code can be memory efficient // produce a small program ... ... because the programmer writes each instruction for the processor // controls exactly which instructions // memory locations are used, so there are no unnecessary instructions **Control the hardware directly:** assembly language allows direct manipulation of the hardware ... ... because its instructions are (mnemonics for) the processor's own machine code // instruction set, so the driver can access specific memory locations // registers // the print head directly **Respond quickly:** assembly language code can execute quickly // efficiently ... ... because it can be written specifically for that processor // uses the processor's own instructions, with no unnecessary instructions	**4**	max 4: max 2 per requirement, max 2 requirements. The expansion mark needs a matching feature mark. Credit any other valid expansion that explains the link. "It runs faster" earns only the feature mark for "respond quickly"; the expansion needs a reason. Do not accept "it does not need a translator" (it needs an assembler).
5(b) 0478 4.2.1	Any two from: Harder to read // write // understand (than a high-level language) Harder to debug // errors are harder to find and fix Machine dependent // only runs on one type of processor // must be rewritten for a different type of processor Takes longer to write // more lines of code are needed (for the same task) The engineer must know the processor's instruction set // hardware in detail	**2**	max 2. Do not accept "it runs slower", "it needs a compiler" or "it uses more memory" (all wrong for assembly language). Do not accept "harder to use" alone (too vague).

Total: 20 marks

MODULE P LANGUAGES AND IDES · P3

Translators: compilers and interpreters: mark scheme

20 marks · Cambridge 0478 syllabus 4.2.3 (operation of a compiler and an interpreter), 4.2.4 (advantages and disadvantages)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 4.2.3	Compiler Interpreter	**2**	Either order. Do not accept assembler (it translates assembly language, not a high-level language).
2(a) 0478 4.2.3	Produces an executable file: Compiler Translates one line of code, executes it, then moves on to the next line: Interpreter Translates high-level language into machine code: Both Gives one error report listing every error it detects in the code: Compiler Halts when it reaches an error, then continues from that point once the error is corrected: Interpreter	**5**	One mark for each correct row. More than one tick in a row: no mark for that row.
3(a) 0478 4.2.4	Writing and testing: Interpreter (1) Reason, any one from: Errors are reported as soon as they are found // at the line where they occur Execution stops at the error, so it can be corrected and the program continued from that point Code can be run / tested straight away // without translating the whole program each time a change is made Easier to debug // find and correct errors Final program: Compiler (1) Reason, any one from: Produces an executable file (to load onto the arms) The arms do not need a translator to run the program The program is translated once // not translated each time it runs // runs faster The source code is not supplied with the program // the source code is protected	**4**	A reason can only be credited if the translator in that row is correct. Do not accept "faster" with no reference to execution / running (e.g. "faster to translate").
4(a) 0478 4.2.3	Any three from: Lines 01 to 04 are translated and executed // "Greenhouse log started" is output and a reading is input Execution stops at line 05 // when the first error is found The error on line 05 is reported / displayed The error on line 07 is not reported (yet) // it is only found after line 05 is corrected and execution continues	**3**	max 3. Do not accept "both errors are reported together".

QUESTION	ANSWER	MARKS	GUIDANCE
4(b) 0478 4.2.3	Any two from: The whole code is translated (before any of it is executed) // none of the program is executed An error report is produced for the whole code The report lists both errors // lines 05 and 07 No executable file is produced (until the errors are corrected)	**2**	max 2. Do not accept "it stops at line 05".
5(a) 0478 4.2.4	One mark for each advantage, one mark for its expansion, for two advantages: The shops do not need a translator / interpreter (1) ... // only the executable file is needed to run the program ... so it is simpler to install / run (1) The source code is not supplied (1) ... so the shops / others cannot see / copy / change the company's code (1) The program has already been translated (1) ... so it is not translated each time it runs // it executes faster (1)	**4**	max 4. Expansion only creditable if it matches its advantage. Do not accept "more secure" or "faster" alone.

Total: 20 marks

MODULE P LANGUAGES AND IDES · P4

IDE tools and how they help a programmer: mark scheme

21 marks · Cambridge 0478 syllabus (exams 2026-2028) 4.2.5 (explain the role of an IDE in writing program code and the common functions IDEs provide: code editors, run-time environment, translators, error diagnostics, auto-completion, auto-correction, prettyprint)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. 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. Do not credit brand names of software. Do not credit functions of an operating system (e.g. memory management) as IDE functions.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 4.2.5	Row 2 ticked: Software that provides the functions needed to write, translate, run and test program code in one program	**1**	More than one box ticked = 0. Row 1 is a compiler, row 3 an operating system, row 4 assembly language.
2(a) 0478 4.2.5	One mark for each correct function: Types `REP`, IDE offers `REPEAT`: auto-completion Types `OUPTUT`, IDE changes it to `OUTPUT`: auto-correction Keywords, identifiers and strings in different colours, consistent indentation: prettyprint High-level code converted into machine code: translator Program run inside the IDE, outputs shown for testing: run-time environment	**5**	Accept "compiler" // "interpreter" for translator. Accept "autocomplete", "autocorrect", "pretty print(ing)". Code editor and error diagnostics are not needed. If the same function is written in more than one row, credit only the row where it is correct.
3(a) 0478 4.2.5	Error diagnostics	**1**	Accept "error reporting" // "debugger". Do not credit "translator" / "compiler" / "code editor".
3(b) 0478 4.2.5	Any two from: It identifies / shows the line number (line 03) where the error is // highlights the line with the error ... so the programmer can find the error quickly // knows where to look It describes / explains the error // says a quotation mark is missing ... ... so the programmer knows what to change // to add the missing " (after `gallery` / before the comma) It may suggest a correction It reports the error before the program is run // it can report all the errors found	**2**	max 2. Do not credit "it finds errors" / "tells you there is an error" alone.

QUESTION	ANSWER	MARKS	GUIDANCE
3(c) 0478 4.2.5	Any two from: Prettyprint shows strings / keywords / identifiers in different colours ... the rest of line 03 (`, Gallery`) is shown in the string colour // `Gallery` is not shown in the identifier colour ... so the programmer can see where the string should have ended // that the string has not been closed ... (so) the error can be seen as it is typed // before the program is translated / run Consistent indentation / layout makes the code easier to read // errors easier to see	**2**	max 2. "Makes code look neat" alone is not enough. Credit a general point about colour-coding plus one point applied to line 03.
4(a) 0478 4.2.5 (4.2.3)	Any two from: The program is written in a high-level language // source code The processor / CPU can only execute / understand machine code // binary The translator converts the high-level code into machine code / binary ... ... so the program can be run / tested inside the IDE // without needing a separate translator program	**2**	max 2. Do not credit "to translate the code" alone (repeats the question).
4(b) 0478 4.2.5	Any two from: Lets the program be run / executed inside the IDE // without leaving the IDE Shows the outputs (e.g. the total number of visitors) // the programmer can enter test inputs (e.g. visitor numbers) ... so the actual output can be compared with the expected output Shows errors that happen while the program is running // where the program stops	**2**	max 2. "Tests the program" alone is not enough.

QUESTION	ANSWER	MARKS	GUIDANCE
5(a) 0478 4.2.5	One mark for each correct function (max 3). One mark for each matching explanation of how it helps (max 3). For example: Auto-completion // ... suggests / finishes keywords / identifiers as he types, so code is written faster // with fewer spelling mistakes Auto-correction // ... corrects misspelt keywords / common mistakes automatically, so fewer (syntax) errors reach the translator Prettyprint // ... colours / indents the code, so it is easier to read // errors are easier to spot Error diagnostics // ... show the line / location and a description of each error, so errors are found and corrected quickly Run-time environment // ... runs the program and shows its output inside the IDE, so he can test it without changing to another program (Built-in) translator // ... converts the code to machine code inside the IDE, so he can translate and run with one action // without a separate program Code editor (with line numbers / auto-indent) // ... he can write and change code and find a line given in an error message	**6**	max 6. An explanation scores only if it matches the function given. A function with no explanation scores 1. Listing more than three functions without explanations scores max 3. The role of an IDE (all the functions are in one program, so he does not need to switch between programs) may be credited in place of one explanation mark. Debugging tools (breakpoints, stepping, watching variables) may be credited under error diagnostics.

Total: 21 marks