2.5 – Programming languages and Integrated Development Environments – Past Exam Questions – Solutions

2022

4	(b)	(ii)	• high-level • stops // crashes • no • executable • without	5 (AO1 1b, AO2 1b)	Ignore spelling errors.

Sample Paper

5	a		• To convert it to binary/machine code • The processor can only understand machine code	1 (AO1 1a)	Maximum 1 mark
	b		• Compiler translates all the code in one go… • …whereas an interpreter translates one line at a time • Compiler creates an executable… • …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	4 (AO1 1b)	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.
	e		• 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.

2021

	(b)		• Transistor has two **states** • 1 represents on, 0 represents off • Each transistor stores one bit • Multiple transistors used to store a binary value	2	Allow values for BP1

2020

2	(b)		1 mark per bullet to max 2 • **Easier/quicker** for humans to **write** • **Easier/quicker** to **read / understand / remember** • **Easier/quicker** to **maintain / debug / spot errors** • …because code is closer to English / uses English words • Less code to write • …because one HLL instruction represents many assembly instructions • Portable (between processors) // will work with different types of computer	2 AO1 1b(2)	Accept "human language" as English for BP4 "Easier to use" is too vague.
2	(c)		1 mark per bullet to max 2 • Each character (in character set) has a **unique** (binary) number/value • Each character in the **string** is assigned its associated number/value • The (binary) value of each character is stored/combined (in order) • … by example e.g. The binary value for D, then for r, then for u • Uses ASCII/Extended ASCII/Unicode	2 AO2 1a(2)	

2019

2	**(c)**		1 mark per bullet to max 4, 2 mark max per method • Compiler • …translates code **in one go** / all at once • …produces an executable file // does not need to be compiled again • Interpreter • …translates code line by line. • …will be interpreted / translated every time it is run.	4 AO1 1b (4)	Mark first method only in each section

2018

7	**(a)**	**(ii)**	1 mark per bullet, max 2. • aimed at humans//understandable by humans / programmers • English like structure / syntax • Must be translated/compiled/interpreted (before it can be run) • Allows programmer to deal with the problem instead of considering the underlying hardware // an abstraction from the hardware // hardware independent // portable	2	Allow examples of keywords (eg IF / ELSE / WHILE) as 2nd bullet point. Do not award marks for naming languages such as Java , Python, etc. Do not award marks for stating what a high level language isn't (i.e. describing what low level code is). Do not allow "easy to use" Do not allow 'has to be converted' without into what i.e machine code etc.
7	**(b)**		1 mark per bullet, max 4. e.g. • Editor • …to enable **program code** to be entered/edited • Error diagnostics / debugging • …to display information about errors (syntax / run-time) / location of errors • … suggest solutions • Run-time environment • …to enable to the program to be run • … check for run time errors / test the program • Translator / compiler / interpreter • …to convert the high level code into machine code / low level code / binary • ...to enable to code to be executed / run	4	One mark for identifying, one mark for describing. Accept description of a tool without (or with incorrect) naming of the tool. Allow sensible descriptions which go across pairs or name other tools sensibly (e.g. editor / highlighting syntax) Allow any sensible tool that an IDE provides (e.g. auto documentation, help tools, pretty printing etc.)

2016

4	f		1 mark for identification, 1 for matching description e.g. • Error diagnostics/debugger • …highlight errors/suggest changes • Run-time environment • …Lets you run/test the program • Text editor • …highlight key words • …auto-indent • …to type/edit source code • …Auto-complete • …highlight syntax errors • Versioning tools • …To allow for tracing back • …To create new files with changes • Stepping/breakpoints • …Allow tracing of algorithms	4	Do not allow auto-documentation. Can get description mark, without identification/incorrect identification Allow: • Variable watch/window • See how the values change Do not allow compiler/interpreter

4	g		Max 2 for compiler, 2 for interpreter Compiler • To convert to low-level in one go • Create an executable//export the file • To distribute the software • Users will have no access to source code... • ...so no-one can edit/steal/copy the code/program • Use for error detection Interpreter • To convert to low-level line by line • To test the program // to find errors • stops running when it finds an error//shows the location of the error when found • it is quicker (compared to compiler) to re-interpret than re-compile	4	The uses must be different for compiler and interpreter

2015

5	a	i	High level code : • human oriented code / written by programmers • contains words for commands / closer to English/natural language • Machine independent /Portable to different systems • Needs to be translated before it can be executed. • Problem based • One (high level) command equates to many machine code instructions. Machine code: • Code for the CPU to execute / not readily understandable by humans • binary instructions • specific to a particular (type of) computer / not portable to different systems • does not need to be translated [max 2 marks for each type of code]	4	Award marks for correct points about machine code made under high level code and vice versa. Do not accept Machine code is in Hex
		ii	• To translate the high level code into machine code • To pick up (syntax) errors	1	Translate to object code is acceptable Accept "errors" on its own, but do not accept answers referring specifically to logic or runtime errors.

Extra Questions

11			1 mark for feature, 1 for benefit. Max 2 per feature. e.g. • Auto-complete • Can view identifiers / avoid spelling mistakes • Colour coding text / syntax highlighting • Can identify features quickly / use to check code is correct • Stepping • Run one line at a time and check result • Breakpoints • Stop the code at a set point to check value of variable(s) • Variable watch / watch window • Check values of variables and how they change during the execution • Error diagnostics • Locate and report errors / give detail on errors	6 AO1.1 (3) AO1.2 (3)	Question states when writing the code, therefore use of compiler / producing .exe etc. are not awarded marks Accept any suitable features e.g. traces, crash dump, stack contents, cross-references, line numbers, auto-indent **Examiner's Comment:** Most candidates achieved some credit for factual recall. However, weaker candidates often answered debugger rather than explaining the specific features of the debugger which would have been creditworthy.
	b		1 mark per bullet to max 3 e.g. • Provides a text editor / allows the code to be written • Provides debugging tools / allows the code to be tested • Provides a translator/compiler/interpreter / provides a run-time environment / allows the code to be run • Description of key feature e.g. colour coding keywords, autocomplete, breakpoints etc.	3 AO1.2 (3)	<u>Examiner's Comments</u> It was clear that nearly all candidates had experience of using an IDE and that they could successfully identify a number of features that an IDE provides.

17	a		• Debugging tools allow inspection of variable values (1 – AO 1.1) this can allow run-time detection of errors (1 – AO 1.2). • Code can be examined as it is running (1 – AO 1.1) which allows logical errors to be pinpointed (1 – AO 1.2). • IDE debugging can produce a crash dump (1 – AO 1.1), which shows the state of variables at the point where an error occurs (1 – AO 1.2). • It can display stack contents (1 – AO 1.1) which show the sequencing through procedures / modules (1 – AO 1.2). • It can step through code (1 – AO 1.1), which allows the programmer to watch the effects each line of code (1 – AO 1.2). • The insertion of a break-point (1 – AO 1.1) allows the program to be stopped at a predetermined point in order to inspect its state (1 – AO 1.2).	6	1 mark (AO 1.1) for each correct identification up to a maximum of three identifications plus up to a further 1 mark (AO 1.2) for each of three valid descriptions.
	b		• A (single) program (1) used for developing programs (1) made from a number of components (1).	2	Up to 2 marks for a valid description.

20			1 mark per bullet, max 2 for each tools Breakpoints • Use to test the program works up to/at specific points • Check variable contents at specific points • Can set a point where the program stops running Stepping • Can set the program to run line by line • Slow down/watch execution • Find the point where an error occurs	4	