GCST

1.2.2 APPLICATIONS GENERATION

TOPIC WISE EXAM QUESTIONS

ANSWERS

A-LEVEL OCR

GCST

2023

1	(a)	(i)	e.g. • **Fewer** mistakes (likely to be made) // **More** accurate • Faster as you can apply the same formula to multiple cells // By example • What-if analysis can be performed • Values can be changed and results **automatically (re)calculated** (by using formulas) • Can be shared electronically	1	Do not accept "faster" on its own without clarification of what/why it is faster.
1		(ii)	e.g. • Database/DBMS • …to store/query/sort data about customers/staff/stock • Word processor • …to create documents / letters / invoices for clients/staff • Presentation software • …to create presentations for clients/staff • Email software • …for staff to communicate with each other or with customers • Graphics manipulation • …to produce adverts / images for sales • Web browser • ….to view websites to purchase materials/stock // view competitor's website	4	Mark in pairs – one mark for naming type of application software, one for the example. Application type must be correct to give example. Do not accept brand names for first mark but FT for example. Ignore brand names if description given after E.g. Outlook / Email application Accept other sensible application software (such as CAD, Desktop Publishing). Do not accept special purpose / bespoke / utility software. Do not accept spreadsheet (given in question) Example must be relevant to the business
1		(iii)	• No access to <u>source code</u> • Cannot modify//improve to meet business needs • Cannot fix bugs • (Usually) cost to purchase licences // licence conditions to meet//ongoing fees	3	Do not award a reverse of the mark point by describing open source

2022

(b)	(i)	• To convert (high-level or assembly) code to <u>low level/machine code</u>	1	Do not allow answers referring to making the program executable, given in question.
	(ii)	• Compiler translates code all at once/before it's executed • Interpreter translates code line by line / during runtime • Compiler produces executable file for reuse // Doesn't need to be translated everytime it is run • Interpreter needs to re-translate next time program is run • Compiler lists all errors//Compiled code doesn't run if there are any errors • Interpreter stops at the first error • Compiled programs have the source code hidden • Interpreted programs have the source code visible	4	Mark answers in pairs Max 2 marks per answer space

(c)		*(table below)*	5	One mark per row. No mark if more than one/no box is ticked. Accept other marks that clearly indicate choice (e.g. X)

	Lexical analysis	Syntax analysis	Code generation
Comments and whitespace are removed	x		
Keywords are replaced with tokens	x		
Object code is created			x
Symbol table created for variables	x		
Builds an abstract syntax tree		x	

(d)		• To make the program run faster// code is more efficient • To make the program use fewer resources/less memory	2	

AS - Level

2	(a)		1 mark per bullet up to a maximum of 2 marks, e.g: • Word Processing o Writing letters to customers • Spreadsheets o Completing accounts • Presentation Software o Create\Show business plans • DTP o Creating marketing literature • Graphics Package o Editing photographs of procedures/marketing photos	2 AO1.1 (1) AO2.1 (1)	1 Mark for a suitable package and 1 mark for a relevant example for that package. Do not allow: • Non-business software (E.g. games) • Brand names (e.g Word/ Excel) • Database software
2	(b)		1 mark per bullet up to a maximum of 2 marks, e.g: • Disk Defragmentation… • …To keep optimal r/w speed for her HDD • File management… • …To allow easy access to her file system • Disk Drivers… • … To allow her to use new peripheral devices • System Clean-up… • … to keep her system free of redundant files • Anti-Virus/Malware…	4 AO1.1 (2) AO1.2 (2)	1 Mark for a suitable utility and 1 mark for a relevant example for that utility. Do not accept task manager Accept: • Compression Software … • … to make the file size smaller • Backup Software … • … to make copies of files

2	(c)		**Mark Band 3–High Level** **(7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of open and closed source software; the material is generally accurate and detailed. The candidate has covered all 4 sections of cost, usability, security and Support available and for the top of this mark band will have covered all 4 well. Evidence/examples will be explicitly relevant to the explanation. *There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.* **Mark Band 2-Mid Level** **(4-6 marks)** The candidate demonstrates reasonable knowledge and understanding of open and closed source software; the material is generally accurate but at times underdeveloped. The candidate has covered all at least 2 of the 4 sections of cost, usability, security and Support available. Evidence/examples are for the most part implicitly relevant to the explanation. *There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.* **Mark Band 1-Low Level** **(1-3 marks)** The candidate demonstrates a basic knowledge of how the layers of open and closed source software; the material is basic and contains some inaccuracies. The candidate makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides nothing more than an unsupported assertion. *The information is basic and communicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.* **0 marks** No attempt to answer the question or response is not worthy of credit.	AO1.1 (2) AO1.2 (2) AO2.1 (2) AO3.3 (3)	<u>**Knowledge:**</u> **Cost** Open Source • (generally) free to use • May have to purchase maintenance contracts • Staff training if "non-standard" Closed Source • (sometimes) have to pay to license the software • If paid will (usually) it will come with some level of support **Usability/extensibility** Open Source • Tends to have a lower focus on UI • Source code released (under license) • Source can be edited • Can be redistributed (under license) Closed Source • Professionally developed • Distributed with a restrictive license • Only executable/object code is distributed//source code not distributed • Cannot be redistributed **Security** Open Source • potentially massive bank of volunteer developers working on the product • Many of the contributors may not be professional • Code available to be scrutinised by anyone… • …but this may include people with malicious intentions Closed Source • Closed teams of developers • More work scrutiny for code • security fixes usually addressed quicker **Support Available** Open Source • Source code released (under license) • Source can be edited • Open communities mean lots of support options could be available <u>**Closed Source**</u> • <u>Support may be available from the company producing the software.</u> • <u>**Application:**</u> **Cost** Open Source • Lower overheads to company • Extra staff training and hardware cost could lead to total cost of ownership being higher Closed Source • Support from vendor can lead to quicker fixes. **Usability** Open Source • The ability to edit source code means bespoke functionality can be developed in house • Lower focus on UI can mean a harder to use product (leading to higher training costs) Closed Source • Due to professional development, finish tends to be a higher standard

					• As the organisation protects their IP they generally tend to be less buggy as the organisation reputation/business model will rely on it • Lack of source code means extra features can only be developed by the vendor. **Security** Open Source • Tends to be less secure as more people working on it, not always under rigorous oversight • No paid developers mean people may not work on security fixes straight away Closed Source • Developers work under tighter standards • Code being scrutinised more will mean less likely to be ship with bugs • Professional standards lead to quicker turnaround of bugs **Support Available** Open Source • Editable source code means could self-support • Open communities mean there is vast amounts of knowledge available. **<u>Evaluation:</u>** **Open Source** • Open source would lead to potential cost savings if Charlie looked to self support by using online communities or handling the code herself. **Closed Source** • If Charlie lacked technical skills, the better UI design from professional developers may make the UX smoother • Charlies would have a legal obligation to the data stored for her business. Closed source tighter security may strengthen this

GCST

2020

	(c)		e.g. • Encryption • …scrambles meaning of data files with a key • Defragmentation • …organises file segments on secondary storage • Compression • … reduces size of files • Backup • …makes regular copies of files in case of loss • Disk Checker	4 AO1.1	Mark in pairs, 2 marks per example. Accept other sensible examples of utility software.
	(d)	i	• Source code / program code is freely available • …to edit/amend recompile.	2 AO1.1	
		ii	e.g. • Can modify code and adapt IDE to her needs • Is likely to be financially free of cost. • Can recompile to work on different systems • Has the benefit of a community potentially improving the system • Can learn from others • Can ensure no backdoors / malware	1 AO2.1	Do not accept simply seeing the code (previous question).
	(e)	i	• Sections of code / program file • Written by other authors / already written • Containing useful routines • Suitable example (e.g. GUI routines, database access routine, encryption, graphics)	3 AO1.1 (2) AO2.1 (1)	Maximum 2 for definition, 1 for example
		ii	• Save time… • …because no need to rewrite code • Use expertise of others.. • …to complete tasks that require specialist knowledge / abstract away complexity. • Has already been tested/the programmer doesn't have to test it themselves • Making debugging easier/saving time	2 AO1.2	
		iii	• May (significantly) increase size of compiled file… • …as library contains many routines that aren't being used. • Not written by the programmer • …so introduces uncertainty / require further testing / programmer needs to spend time familiarising themselves with it	2 AO1.2	
		iv	• Links the main program to libraries • ..can either include them in the final executable code • ..or get the executable code to point to the external libraries	3 \ AO1.2	Accept terms static and dynamic for bullets 2 and 3, but only if these are explained.

6	a		– A program with one purpose/piece of system software – …used for the upkeep/maintenance of the system (1 per -, max 2)	2 AO1.1	
	b		– Application performs tasks for the user (rather than computer). – Performs generic (rather than specific) tasks	1 AO1.2	

	f		– The comments such as those on the first line, (and whitespace) are removed. – Variable names/identifiers like 'count' are added to a symbol table. – Reserved words/statement components are tokenized. For example 'WHILE' (1 mark per -, max 3)	3 AO2.2	
	g		Syntax analysis	1 AO1.1	

AS - Level

1	a		- Open source has the <u>source code</u> freely available… - … to amend/copy/redistribute/recompile - Whereas closed source is distributed in binary form only/the source code is not made available… - There are licensing conditions restricting the redistribution/there is no permission to amend the (program) code (1 per - , max 4)	4 (AO1.1)	
	b		- Compilers translate the source code prior to distribution - Meaning the user gets an executable program (which makes amending of the program much more difficult). - Interpreters translate source code every time the program is run - meaning the user needs the source code to run the program - (1 per - , max 2)	2 (AO2.1)	
	c		An Assembler	1 (AO1.1)	

GCST

2018

c			- Free of cost - Right to inspect/amend/recompile source code - Can tailor the program to their specific needs - Code open for bugs to be spotted and fixed. (1 Mark per -, Max 2)	2 (AO1.2)

2017

di		Saves time/money as prewritten (1) Draws on expertise of other programmers (1) Pre-tested (so likely to work) (1) Can have been written in a different language (1) (Max 2)	2 (AO1.2)	
dii		**Mark Band 3–High Level (7-9 marks)** The candidate demonstrates a thorough knowledge and understanding of how source code is compiled and library code incorporated. The material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples will be explicitly relevant to the explanation. The candidate provides a thorough discussion which is well balanced. Evaluative comments are consistently relevant and well-considered. There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated. **Mark Band 2-Mid Level (4-6 marks)** The candidate demonstrates reasonable knowledge and understanding of how source code is compiled and library code incorporated; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context	AO1.1 (2) AO1.2 (2) AO2.1 (2) AO3.3 (3) 9	Points may include but are not limited to: **AO1 Knowledge and Understanding** The compiler is effectively a group of programs. The stages of compilation are: lexical analysis, syntax analysis, code generation and optimisation. A linker is then used to combine the object code with the library code to make the final executable. **AO2.1 Application** Source code is input into a compiler program. The first stage is lexical analysis in which.. Comments and whitespace are removed Variables, and subroutines stored in symbol table Which also holds data such as scope and data type Code is converted to a series of tokens The series of tokens and symbol table is passed onto the next stage, syntax

			provided although one or two opportunities are missed. Evidence/examples are for the most part implicitly relevant to the explanation. The candidate provides a sound discussion, the majority of which is focused. Evaluative comments are for the most part appropriate, although one or two opportunities for development are missed. There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence. **Mark Band 1-Low Level (1-3 marks)** The candidate demonstrates a basic knowledge of how source code is compiled and/or library code incorporated; the material is basic and contains some inaccuracies. The candidate makes a limited attempt to apply acquired knowledge and understanding to the context provided. The candidate provides a limited discussion which is narrow in focus. Judgments if made are weak and unsubstantiated. The information is basic and communicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear. **0 marks** No attempt to answer the question or response is not worthy of credit.	analysis: Here the code is checked to ensure it follows the rules of the language. This is often accomplished by placing the tokens into a (abstract syntax) tree. Where it breaks the rules of the language errors are generated. If no rules are broken then it's passed on to the next stage… ..Which is code generation. Here the object code (accept machine code) is created. (i.e. the binary that is executed by the processor) This code may be inefficient.. .. it may contain unnecessary instructions or groups of instructions that can be replaced by simpler ones. Code from the library is likely already compiled. And may well have been written in a different language to the main program. The main program source code will have contained lines importing the library code. A program called a linker can incorporate the code from the library with the main program… …into a single executable file. An alternative approach is for the main executable to link to the compiled library code (i.e. dynamic linking). **AO3.3 Evaluation** Lexical analysis is necessary to put the code into a format which can be
				read and processed (i.e. parsed) by the syntax analyser. Syntax Analysis is necessary to ensure the code is valid in as much as it meets all the structural rules of the language. This guarantees it will run (though it might not do as expected and may still have occurrences of run-time errors). Code generation is necessary to turn the code into a format that the processor can understand (i.e. binary machine code). The code optimisation whilst not necessary, does ensure the code runs quicker or using less memory. Linking is necessary to ensure the library code is incorporated into the final program.

1			• The user running the program will not necessarily have the library installed on their machine (1) therefore the relevant code needs to be included within the final executable (1) – it is the job of the linker to combine this code (1).	3	Up to 3 marks for a valid explanation.
			Total	**3**	
2			Relatively error free / has already been tested Ready to use / saves time/already been written Used multiple times / common tasks / reduces repeated code Programmer expertise Different source languages	3	**Examiner's Comments** Excellently answered, very few candidates had any problem with this question and most candidates gained at least two out of the three marks.
			Total	**3**	
3			2 Marks from this section • Statements / tokens are checked… • … against the rules / grammar of the language • valid example given 3 Marks from this section • Errors reported as a list • Error diagnostics given • Detail added to symbol table… • …eg data type / scope / address • Receives output from lexical analysis / passes code to code generation	5	No syntax check **Examiner's Comments** Again the Principal Examiner was looking for more technical knowledge with this question and it was apparent from the range of answers given that this was one question that showed true understanding of the subject.