Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/32**

Paper 32 Advanced Theory **May/June 2022**

MARK SCHEME

Maximum Mark: 75

Published

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the May/June 2022 series for most Cambridge IGCSE, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

This document consists of **9** printed pages.

PUBLISHED

Generic Marking Principles

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptors for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1:

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

GENERIC MARKING PRINCIPLE 2:

Marks awarded are always **whole marks** (not half marks, or other fractions).

GENERIC MARKING PRINCIPLE 3:

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

GENERIC MARKING PRINCIPLE 4:

Rules must be applied consistently, e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

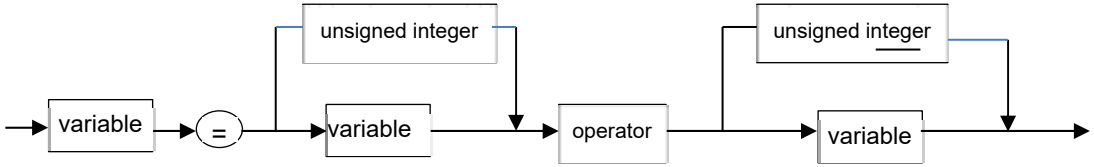
GENERIC MARKING PRINCIPLE 5: Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).
GENERIC MARKING PRINCIPLE 6: Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

Question	Answer	Marks
1(a)	`BuildingRegister.BuildingID ← 1067` `BuildingRegister.BuildingGroup ← "house"`	**2**
1(b)(i)	**One** mark: `TYPE BuildingType =` **One** mark: `(house, bungalow, apartment, farm)` `TYPE BuildingType = (house, bungalow, apartment, farm)`	**2**
1(b)(ii)	`DECLARE BuildingGroup : BuildingType`	**1**
1(b)(iii)	`BuildingRegister.BuildingGroup ← house`	**1**
1(c)(i)	`PRIVATE OwnerName : STRING`	**1**
1(c)(ii)	To ensure that attributes can only be accessed by the class's own methods To enforce encapsulation // ensure they are hidden	**2**

Question	Answer	Marks
2(a)	`studies(sam, history).` `tutors(nina, sam).`	**2**
2(b)	`freya, hua` // `hua, freya`	**1**
2(c)	**one** mark for correct use of `X` **one** mark for two other variables in correct positions **one** mark for three correct clauses in any order **one** mark for correct syntax `teaches(R, S),` `studies(X, S),` `tutors(R, X).`	**4**

Question	Answer	Marks
3(a)	Protocol **one** mark, description **one** mark, max **four** Any **two** from • HTTP(S) (1) for sending and receiving web pages / hypertext documents (1) • FTP (1) for sending and receiving **files** over a network / between devices (1) • SMTP (1) for **sending/uploading** emails /push protocol (1) • POP(3) (1) for **receiving/downloading** emails /pull protocol (1) • IMAP (1) for **receiving/downloading** emails /pull protocol (1)	**4**
3(b)	Layer **one** mark, matching function **one** mark, max **four** Any **two** from • Transport (1) handles packets (1) • Internet (1) handles transmission of data using **IP addresses** // provides (optimal) route (1) • Network Access (Interface) // (Data) Link // Physical (1) Handles how data is physically sent (1)	**4**

Question	Answer	Marks
4(a)	• An unsigned integer,`12`, is used instead of the last variable // `12` is not a valid variable • The variable `Z` is not a valid variable / missing an unsigned integer after the `Z`	**2**
4(b)	One **mark** per bullet point • `<variable> ::= <letter><unsigned_integer>` • `<unsigned_integer> ::= <digit>|<digit><digit>` • `<digit> ::= 1 | 2 | 3` and `<operator> ::= + | - | *` • `<assignment_statement> ::= <variable> =` • `<variable><operator><variable>`	**5**

Question	Answer	Marks
4(c)(i)	One **mark** adding both boxes… unsigned integer unsigned integer One **mark** for correct position(s) and connector(s) … One **mark** … rest correct (assignment statement)  	**3**
4(c)(ii)	**Max three** **One** mark for `<assignment_statement>::=<variable>=` **One** mark two or three correct options **or two** marks if all four options correct `<variable><operator><variable>` `|<variable><operator><unsigned_integer>` `|<unsigned_integer><operator><variable>` `|<unsigned_integer><operator><unsigned_integer>` *`<assignment_statement>::=<variable>= <variable><operator><variable>|<variable><operator>`* *`<unsigned_integer>|<unsigned_integer><operator><variable>`* *`|<unsigned_integer><operator><unsigned_integer>`* **or** **One** mark for each section `<operand>::=<variable>|<unsigned_integer>` `<assignment_statement>::=<variable>=` `<operand><operator><operand>` *`<operand>::=<variable>|<unsigned_integer>`* *`<assignment_statement>::=<variable>=<operand><operator><operand>`*	**3**

Question	Answer	Marks
5	SIMD (1) many/array **processors execute the same instruction using different data sets** (1) MISD (1) many **processors** (using different instructions) **use the same data set** (1) MIMD (1) many **processors** (using different instructions) **using different data sets** (1)	**6**

Question	Answer	Marks
6(a)	**1 mark** per correct output column **INPUT** **OUTPUT** **A** | **B** | **E** | **F** 0 | 0 | 0 | 0 0 | 1 | 1 | 0 1 | 0 | 1 | 0 1 | 1 | 0 | 1	**2**
6(b)(i)	Half adder	**1**
6(b)(ii)	Purpose of E: **Sum** Purpose of F: **Carry**	**2**

Question	Answer	Marks
7(a)	Any **three** from MP1 enquiry made to Certificate Authority (CA) MP2 enquirer's details checked by CA MP3 if enquirer details verified by CA then public key is agreed MP4 CA creates/issues certificate that includes the enquirers public key MP5 encrypting data sent to/by CA with the CA's public/private key	**3**
7(b)(i)	MP1 The **message** is hashed with (the agreed hashing algorithm)… MP2 … to produce a message **digest** MP3 The message **digest** is then encrypted with the sender's private key to form the digital signature	**3**
7(b)(ii)	Any **four** from MP1 The message together with the digital signature is decrypted using the receiver's private key MP2 The digital signature received is decrypted with the sender's public key to recover the message digest sent MP3 The decrypted message received is hashed with the agreed hashing algorithm to reproduce the message digest of the message received MP4 The two message digests are compared MP5 … if they are the same the message has **not** been altered // if they are different the message has been altered	**4**

Question	Answer	Marks
8(a)	`INTEGER` `9` // `LENGTH(MyList) - 1` `Index + 1` `"Value not found"` (or any similar phrase)	**4**
8(b)(i)	The list to be searched must be ordered/sorted	**1**

Question	Answer	Marks
8(b)(ii)	Any **four** from MP1 Find the middle item / index MP2 Check the value of middle item in the list to be searched MP3 If equal item searched for is found MP4 If this is not equal/greater/less than the item searched for MP5 … discard the half of the list that does not contain the search item MP6 Repeat the above steps until the item searched for is found MP7 … or there is only one item left in the list and it is not the item searched for // lower bound > / = upper bound	**4**
8(b)(iii)	As the number of items in the list **increases the time** to search the list increases	**1**
8(c)	MP1 Linear search O(n) **and** Binary search $O(\log_2 n)$ / O(Log n) MP2 time to search increases linearly in relation to the number of items in the list for a linear search and logarithmically for a Binary search MP3 time to search increases less rapidly for a binary search and time to search increases more rapidly for a linear search	**3**

Question	Answer	Marks
9	To trap (some) **runtime** errors To prevent a program halting unexpectedly To produce meaningful error messages for these errors Example divide by zero // end of file // file not found	**4**