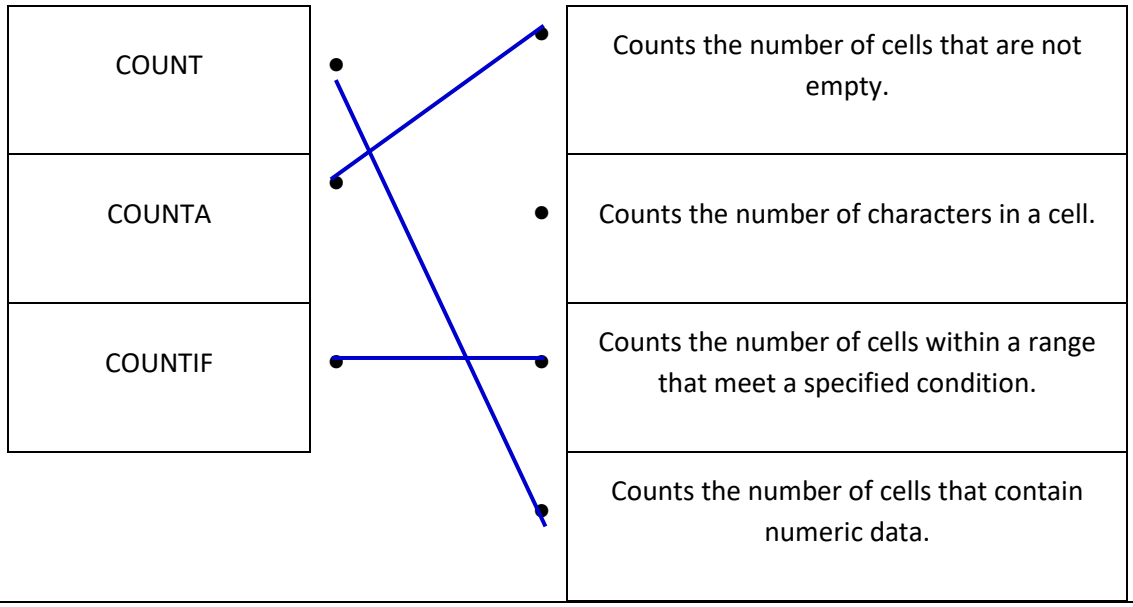
Sec 4 O Level Prelim 2022 Computing Marking Scheme

<table>
<tr>
<td>1</td>
<td>
Function Description


</td>
<td>[3]</td>
</tr>
<tr>
<td>2(a)</td>
<td>When the Central Processing Unit (CPU) receives an instruction to write data to the memory, the uni-directional address bus carries the destination address from CPU to memory. The data bus carries the data to be written. The speed of a CPU is usually measured in gigahertz.

Temporary memory refers to Random Access Memory. This memory is volatile, which means all data and instructions stored on it are erased when the computer is switched off.</td>
<td>[1]
[1]
[1]

[1]
[1]</td>
</tr>
</table>

2(b)	**(i)**	A **bit** is the smallest unit of data in a computer, which can take on the value of either 0 or 1. A **byte** is a unit of data made up of eight bits	[1] [1]
	(ii)	139 MB = **139 000** KB 2 450 000 000 B = **2.45** GB	[1] [1]

3

Problem	Effect	Method of keeping data safe
Power failure	If the power supply to a computer fails, data that is in the process of being written to a storage device may become corrupted and data that is stored in volatile memory but not yet written to a storage device will be lost.	Set up a backup power supply or uninterruptible power supply (UPS) so storage devices can complete any write operations in case of a power failure.
Human error	EITHER ONE: • Storage devices may be accidentally damaged during transport. • Multiple users working on the same file may accidentally overwrite each other's data.	EITHER ONE: • Make regular backups of data. • Use adequate protection when transporting storage devices. • Set up rules when collaborating with multiple users to prevent them from writing to the same file at the same time.
Computer Virus	Virus may purposely damage and corrupt data as a way of attacking the computer.	EITHER ONE: Avoid opening email/chat attachments or files from unknown sources. Install and configure a firewall to prevent them from spreading through the network. Install anti-virus and anti-spyware software, as well as perform regular scans and updates.

Each correct box [1m]. Penalise for repeated effect / method. [6]

4 (a)	Compilers and interpreters translate/convert source code (which humans can understand) into machine code (which only computers understand). 1 mark for general explanation 2 marks if source code and machine code are explained	[2]
4(b)	1) In a compiler, source code is translated into machine code completely before the compiled program can be run. [1] In an interpreter, the source code to machine code translation happens while the interpreted program is running concurrently. [1] 2) In a compiler, syntax errors are detected (during the compilation phase) before the program is run. [1] In an interpreter, syntax errors may interrupt the running of a program. [1] *Alternative answer:* *The compiler is not required to run the program after compilation is completed. [1]* *The interpreter needs to be run every time the program is used (because the program can only be run through the interpreter). [1]*	[4]
5(a)	**Input:** time_spent: number of hours travelling to work every day **Output:** Average number of hours spent travelling to work **Processes required:** Sum up the total number of hours spent travelling for one month and divide by the number of days	[1] [1] [1]
5(b)	**Input:** equipment_list: equipment requiring electrical energy consumption_list: energy consumption of the list of equipment **Output:** Electrical equipment with highest energy consumption **Processes required:** Step 1: Compare the energy consumption of first equipment with the energy consumption of second equipment in the consumption_list. Pick the higher consumption Step 2: Repeat step 1 for subsequent items in the consumption_list until the end of the list has been reached. Step 3: Obtain the equipment from the equipment_list with the highest energy consumption.	[1] [1] [1]

6 (a)

```
01    move = ["Rock", "Scissors", "Paper"]
02    quitGame = FALSE
03    WHILE quitGame == FALSE    #loop until game ends
04        OUTPUT("Enter 1 for Rock.")
05        OUTPUT("Enter 2 for Scissors.")
06        OUTPUT("Enter 3 for Paper.")
07        OUTPUT("Enter 0 to exit.")
08        INPUT user
09        IF user != 0
10            IF user < 1 or user > 3:
11                OUTPUT ("ENTER a valid choice!")
12          CONTINUE #skips the rest of the code below and
repeats at line 03
13            ENDIF
14            computer = RANDINT(1,3)
15            OUTPUT("You picked ", move[user - 1])
16            OUTPUT("Computer played", move[computer - 1])
17            IF computer == user
18                OUTPUT("It is a tie!")
19            ELSEIF computer == 1 and user == 2
20                OUTPUT("Computer wins!")
21            ELSEIF computer == 2 and user == 3
22                OUTPUT("Computer wins!")
23            ELSEIF computer == 3 and user == 1
24                OUTPUT("Computer wins!")
25            ELSE
26                OUTPUT("You win!")
27            ENDIF
28        ELSE
29            quitGame = True
30            OUTPUT("Thank you for playing, goodbye!")
31    ENDWHILE
```

Error 1: **Line 02:** **`quitGame = TRUE`**

Correction: **`quitGame = FALSE`**

Error 2 **Line 10:** **`IF user < 1 or user < 3`**

Correction: **`IF user < 1 or user > 3`**

Error 3 **Line 17:** **`IF computer != user`**

Correction `IF computer == user`

Error 4 Line 21: `ELSEIF computer == 2 and user == 0:`

Correction `ELSEIF computer == 2 and user == 3`

[4]

(b)

Error type 1: Syntax error [1m]

Description: Result of incorrect source code that does not follow the rules of the language [1m]

Error type 2 Runtime error [1m]

Description: Error that occurs while the program is running (e.g. input data that has not been properly validated by a format check to ensure that it is a valid integer) [1m]

[4]

(c)

Test case condition	Test data
Normal	2
	3
Error	3.0
	" " or "two"

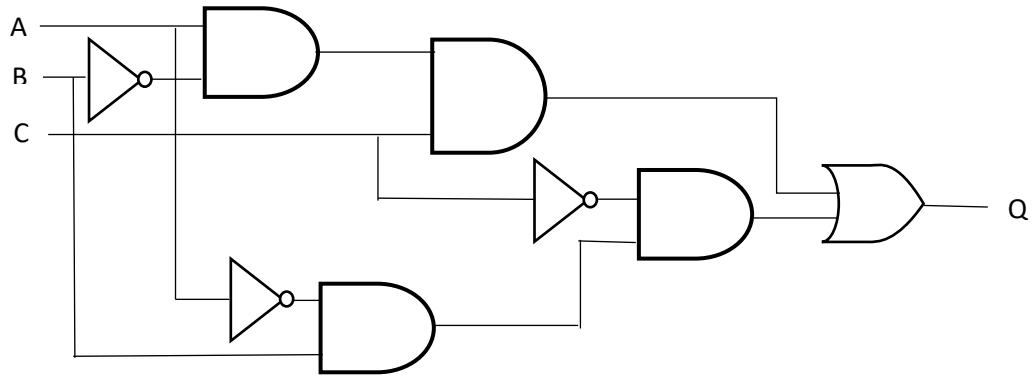
Note: reject out of range test date (e.g. 0, 4, etc), those belong to Boundary.

[4]

7 (a)

X	P	Q	R	S	G	H	OUTPUT
	0	**1**	**2**	**3**	**0**	**0**	
1				**2**	**1**		
9	**3**		**4**		**2**	**1**	
13		**4**			**3**		
28				**1**	**4**		
33	**14**		**6**		**5**	**2**	
34		**24**			**6**		
46				**0**	**7**		**44**

[1]

7(b)	H counts the number of X inputs divisible by 3.	[1]
8	 1 mark for every 2 correct logic gate.	[4]
9	To convert a binary number to its hexadecimal equivalent, we split the number into groups of four binary digits, starting from the right. If a group does not have the full set of four binary digits, add leading zeros to that group until it is a full set of four binary digits. Then convert each four-digit binary group into its corresponding hexadecimal digit using a conversion table. Hexadecimal value: BD_{16}	[1] [1] [1]
10(a)	A series of computing devices connected together.	[1]
(b)	A hub broadcasts the data packet to all computers in the network. A switch only broadcasts it to the computer that is meant to receive it.	[1] [1]
(c)	All the devices are connected to the central server. Data that needs to be shared are hosted on the central server and access can be controlled through the server. Additional devices can be easily added by connecting them to the central server. [Accept other reasonable answers]	[1] [1] [1]
(d)	**(i)**	

1	0	0	1	0	1	1	0

		Correct parity bit value 1 [1m] Correct prepended parity bit position at the beginning of data [1m]	[2]
	(ii)	Any 1 from the following. Maximum 1 mark. • Even number of incorrect bits (or 2 errors / even errors) can cancel out each other. • Error involving interchange of bit positions will not affect parity. • Not able to detect where the error occurred. • Not able to detect all types of errors.	[1]

11	Sample algorithm
	
	`Contestant_List = []`
	`Winner_Name = ""`
	`Winner_Score = 0`
	`Winner_Index = 0`
	
	`FOR Count = 1 to 20:`
	`  INPUT Name`
	`  INPUT Score`
	`  Contestant_List += [[Name, Score]]`
	`  IF Score > Winner_Score:`
	`  Winner_Name = Name`
	`  Winner_Score = Score`
	`  Winner_Index = Count - 1`
	`  ENDIF`
	`NEXT`
	
	`OUTPUT "The array index of the contestant with the highest score is", Winner_Index`
	`OUTPUT "The winner is", Winner_Name, "with a score of", Winner_Score`	

	<u>Award 1 mark for each of the following steps shown, up to a maximum of 8 marks</u> - initialisation of array for contestant list - initialisation of variable for winner name (^^) - initialisation of variable for winner score (^^) - initialisation of variable for array index of winner (^^) (maximum of 2 marks awarded for (^^) as indicated) - loop control for 20 contestants - input of contestant's name - input of contestant's score - input of name & score into contestant list array - checking for highest score - updating winner name if highest score condition is met (&&) - updating winner score if highest score condition is met (&&) - updating array index of winner if highest score condition is met (&&) (maximum of 2 marks awarded for (&&) as indicated) - output of array index of winner in contestant list - output of winner's name and score	