Data Transmission

2.2 Methods of error detection

Marking scheme

Q1)

(a) 1 mark per correctly placed tick

Received byte	Byte transmitted correctly	Byte transmitted incorrectly
1 1 0 0 1 0 0 0		✓
0 1 1 1 1 1 0 0		✓
0 1 1 0 1 0 0 1	✓	

[3]

(b) **(i)** byte number: **7**

column number: **6**

[2]

(ii) Any **two** from:

- letter “A”(byte 7) transmitted as odd parity (three 1s)
- column 6 has odd parity (seven 1s)
- intersection of byte 7 and column 6 indicates incorrect bit value

[2]

(c) **190**

[1]

(d) Any **one** from:

- 2 bits interchanged (e.g. 1 → 0 and 0 → 1) that won't change parity value
- even number of bits/digits are transposed
- If there are multiple errors in the same byte/column, that still produce the same parity bit, the error will not be detected

[1]

Q2)

(a)

1	1	1	1	1	0	0	0

0	0	0	0	0	1	1	1

[2]

(b) 1 mark for error detection method and 1 mark for description

- Check sum
- … sum of bits is transmitted and checked against the sum of the received bits

- Check digit
- … a digit that is calculated (e.g. using modulo-11) and transmitted with the data

- ARQ
- … when an error is detected in a packet of data a request is automatically sent for the data to be resent

[2]

Q3)

(a) **(i)** 1 mark for correct check digit and 1 mark for showing the calculation

(4 × 1) + (2 × 2) + (4 × 3) + (1 × 4) + (5 × 5) + (0 × 6) + (8 × 7)

= 4 + 4 + 12 + 4 + 25 + 0 + 56 = 105

105/11 = 9 remainder 6

check digit is: **6** [2]

(ii) **1 mark**

- No/incorrect check digit

2 marks

- Total is 78
- 78/11 …
- … gives 7 remainder 1
- check digit should be 1 [3]

(b) **(i)** 1 mark for each correct parity bit

parity bit							
0	**1**	**1**	**0**	**0**	**1**	**1**	**0**

parity bit							
1	**0**	**0**	**0**	**0**	**0**	**0**	**1**

[2]

(ii) Any **one** from:

- an even number of digits are changed
- a transposition error(s) has occurred [1]

Q4)

(a) **1** mark for correct check digit and **1** mark for showing the calculation

$(4 \times 1) + (2 \times 2) + (4 \times 3) + (1 \times 4) + (5 \times 5) + (0 \times 6) + (8 \times 7)$

$= 4 + 4 + 12 + 4 + 25 + 0 + 56 = 105$

105/11 = 9 remainder 6

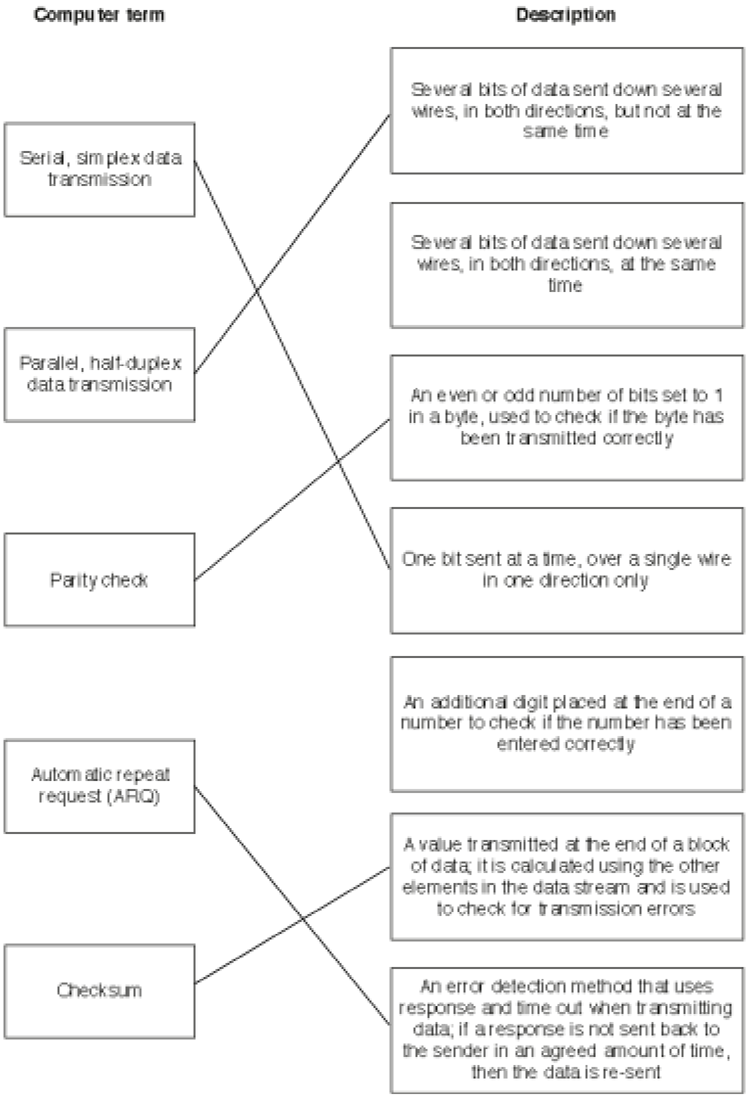
1 mark for any correct line of working

check digit is: **6** [2]

(b) **incorrect** check digit [1]

- check digit should be 1
- (3*1) + (2*2) + (4*3) + (0*4) + (0*5) + (4*6) + (5*7) // 3 + 4 + 12 + 0 + 0 + 24 + 35 // Total = 78
- 78/11 gives 7 remainder 1 [2]

Q5



Computer term

Serial, simplex data transmission

Parallel, half-duplex data transmission

Parity check

Automatic repeat request (ARQ)

Checksum

Description

Several bits of data sent down several wires, in both directions, but not at the same time

Several bits of data sent down several wires, in both directions, at the same time

An even or odd number of bits set to 1 in a byte, used to check if the byte has been transmitted correctly

One bit sent at a time, over a single wire in one direction only

An additional digit placed at the end of a number to check if the number has been entered correctly

A value transmitted at the end of a block of data; it is calculated using the other elements in the data stream and is used to check for transmission errors

An error detection method that uses response and time out when transmitting data; if a response is not sent back to the sender in an agreed amount of time, then the data is re-sent

[5]

Q6)

(d) **(i)** (byte) 5 [1]

(ii) (column) 4 [1]

(iii) corrected byte is: **1 0 0 1 1 1 1 1** [1]

(iv) that gives the value: **1 5 9**
(follow through applies) [1]

(v) Any **two** from:

- The byte would be transmitted without having 5 consecutive 1's
- The fault condition would not be recognised [2]

Q7)

(a) Intersection of Row 7 and column 4 circled [1]

(b)
- Row (byte number) 7 has an odd number of 1s (five 1s)
- Column (bit number) 4 has an odd number of 1s (five 1s)

[2]

Q8)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>1 mark per correct tick
<table>
<tr><th>Received byte</th><th>corrupted during transmission (✓)</th><th>not corrupted during transmission (✓)</th></tr>
<tr><td>10110100</td><td>✓</td><td></td></tr>
<tr><td>01101101</td><td></td><td>✓</td></tr>
<tr><td>10000001</td><td>✓</td><td></td></tr>
</table></td><td>3</td></tr>
<tr><td>(b)</td><td><b>Four</b> from:
∞ Uses acknowledgement and time out
∞ Check performed on received data // error is detected by e.g. parity check, check sum
∞ If error detected, request sent to resend data // negative acknowledgment is used
∞ If no acknowledgement is sent that data is received // positive acknowledgement is used
∞ Data is resent / Resend request repeated, till data is resent correctly …
∞ … or request times out // limit is reached</td><td>4</td></tr>
</table>

Q9)

	Two marks for each correct description **Parity Check** ∞ Checks a byte of data ∞ Check is performed when data is received ∞ A parity bit is added (to the parity byte) ∞ Counts / checks number of 1's // counts / checks to see if 1's are even // counts / checks to see if 1's are odd ∞ Can be even or odd ∞ If parity is incorrect, error is detected **Check digit** ∞ A digit that is calculated from the data // uses modulo to calculate digit // valid description of modulo ∞ It is appended / added to the data ∞ Digit is recalculated when data is entered ∞ Digits are compared to check for error **Checksum** ∞ A value is calculated from the data // Valid description of calculation ∞ It is transmitted with the data ∞ Value is recalculated after transmission ∞ Values are compared after transmission to check for error **Automatic Repeat reQuest** ∞ Uses acknowledgement / request and time-out ∞ Error control protocol ∞ Check performed on receiving data // error is detected by e.g. parity check, check sum ∞ If error detected, request is sent to resend data // negative acknowledgement is used ∞ Resend request is repeated till data is sent correctly / requests time out / limit is reached ∞ Send acknowledgement that data is received // positive acknowledgement is used ∞ If acknowledgement not received in set time data is resent	**8**

Q10)

Question	Answer	Marks
	1 mark for correct register, 3 marks for reason: – Register C Any **three** from: – Count the number of 1/0 bits (in each byte/register) – Two bytes/registers have an odd number of 1/0 bits // Two use odd parity – Odd parity must be the parity used – One byte/register has an even number of 1/0 bits // One uses even parity – One with an even number of one bits/even parity is incorrect // Register C should have odd parity	**4**

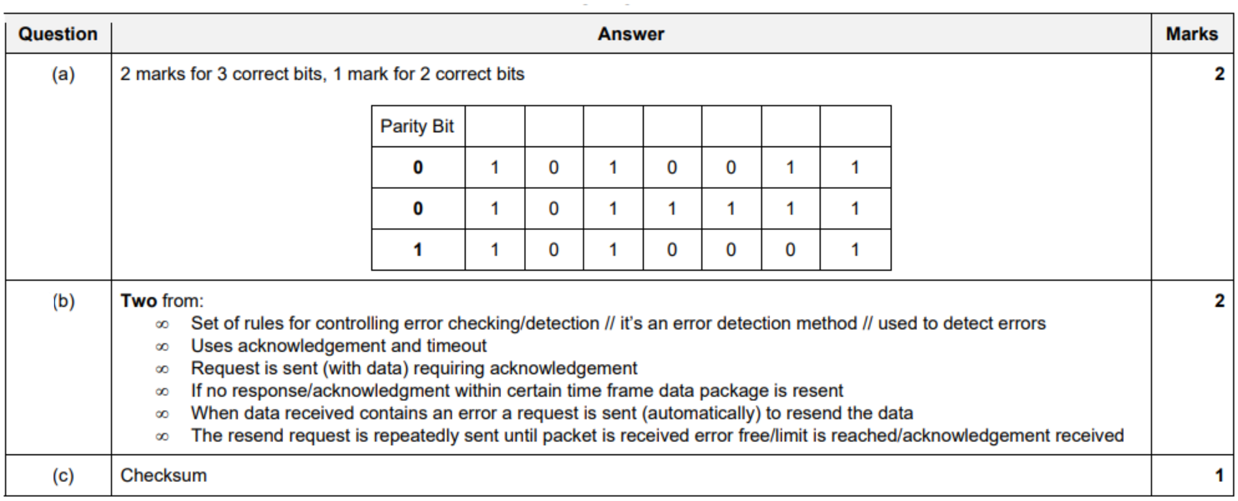
Q11)

Question	Answer	Marks
	1 mark for correct register, 3 marks for reason: – Register Y Any **three** from: – Count the number of 1/0 bits (in each byte/register) – Two bytes/registers have an odd number of 1/0 bits // Two have odd parity – Even parity must be the parity used – One byte/register has an even number of 1/0 bits // One uses even parity – The two with an odd number of one bits/odd parity are incorrect // Register X and Z should have even parity	**4**

Q12)

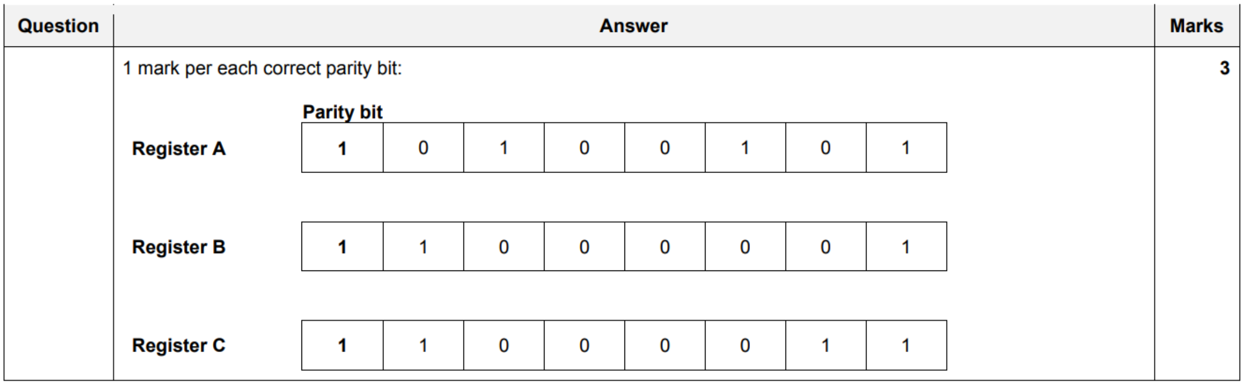
Question	Answer	Marks
(b)	Maximum of three marks per error detection method. 1 mark for naming the method, 2 marks for describing it. **Parity (check)** ∞ Odd or even parity can be used ∞ Bits are added together // 1 bits are counted ∞ Parity bit added (depending on parity set) ∞ Parity checked on receipt ∞ If parity bit is incorrect an error is detected **Checksum** ∞ Calculation performed on data (to get the checksum) ∞ Checksum sent with data ∞ Checksum recalculated after transmission ∞ Comparison made between checksum before and checksum after transmission ∞ Error detected if checksums are different **Automatic repeat request (ARQ)** ∞ Uses acknowledgement and timeout ∞ Request is sent (with data) requiring acknowledgement ∞ If no response/acknowledgment within certain time frame data package is resent ∞ When data received contains an error a request is sent (automatically) to resend the data ∞ The resend request is repeatedly sent until packet is received error free/limit is reached/acknowledgement received	**9**

Q13)



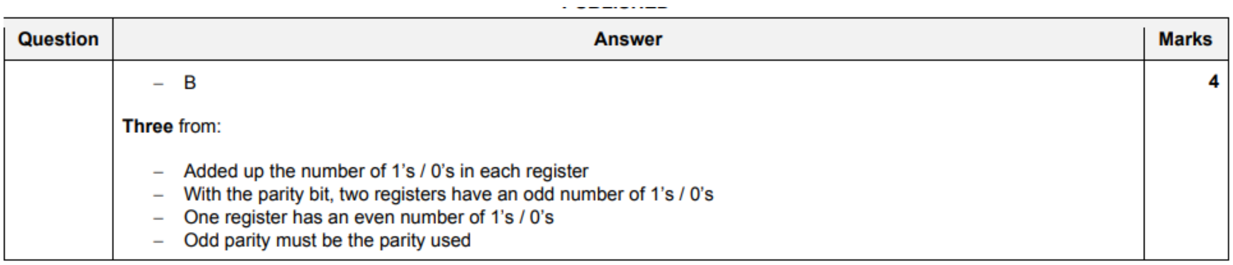
<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>2 marks for 3 correct bits, 1 mark for 2 correct bits
<table>
<tr><td>Parity Bit</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>0</b></td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td><b>0</b></td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td><b>1</b></td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
</table></td><td><b>2</b></td></tr>
<tr><td>(b)</td><td><b>Two</b> from:
∞ Set of rules for controlling error checking/detection // it's an error detection method // used to detect errors
∞ Uses acknowledgement and timeout
∞ Request is sent (with data) requiring acknowledgement
∞ If no response/acknowledgment within certain time frame data package is resent
∞ When data received contains an error a request is sent (automatically) to resend the data
∞ The resend request is repeatedly sent until packet is received error free/limit is reached/acknowledgement received</td><td><b>2</b></td></tr>
<tr><td>(c)</td><td>Checksum</td><td><b>1</b></td></tr>
</table>

Q14)



<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td>1 mark per each correct parity bit:
<table>
<tr><td></td><td><b>Parity bit</b></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td><b>Register A</b></td><td><b>1</b></td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td><b>Register B</b></td><td><b>1</b></td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td><b>Register C</b></td><td><b>1</b></td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
</table></td><td><b>3</b></td></tr>
</table>

Q15)



Question	Answer	Marks
	– B **Three** from: – Added up the number of 1's / 0's in each register – With the parity bit, two registers have an odd number of 1's / 0's – One register has an even number of 1's / 0's – Odd parity must be the parity used	**4**

Q16)

Question	Answer	Marks
(a)	**Four** from: ∞ Validation method ∞ Used to check data entry ∞ Digit is calculated from data // by example ∞ Digit is appended / added to data ∞ Digit is recalculated when data has been input ∞ Digits are compared ∞ If digits are different, error is detected // If digits match, no error is detected	**4**
(b)	**Six** from (maximum three marks per security method): ∞ Firewall … ∞ … Monitors the traffic ∞ … **Blocks** any traffic that doesn't meet the **criteria** / **rules** ∞ (Strong) password // biometric … ∞ … Data cannot be accessed without the use of the password / bio data ∞ … Prevent brute force attacks ∞ Encryption … ∞ … Data will be scrambled ∞ … **Key** is required to decrypt the data ∞ … If data is stolen it will be meaningless ∞ Physical security methods … ∞ … The physical security will need to be overcome ∞ … This can help deter theft of the data ∞ Antispyware … ∞ … will remove any spyware from system ∞ … will prevent data being relayed to a third party	**6**

Q17)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td></td><td><b>One</b> mark for each correct parity bit
<table>
<tr><th></th><th>Parity bit</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></tr>
<tr><td><b>Register A</b></td><td><b>0</b></td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td><b>Register B</b></td><td><b>0</b></td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td><b>Register C</b></td><td><b>0</b></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
</table>
</td><td><b>3</b></td></tr>
</table>

Q18)

Question	Answer	Marks
(a)	Any **two** from: – They both calculate a value from the data – They both append the calculated value to the data – They both recalculate the value – … They both report an error if they don't match	**2**
(b)	**One** mark for method, **three** marks for description: **Automatic Repeat reQuest** – Uses acknowledgement / request and time-out – Error control protocol – Check performed on receiving data // error is detected by e.g. parity check, check sum – If error detected, request is sent to resend data // negative acknowledgement is used – Resend request is repeated till data is sent correctly / requests timeout / limit is reached – Send acknowledgement that data is received // positive acknowledgement is used – If acknowledgement not received in set time data is resent **Parity Check** – A parity bit is added (to the parity byte) – Counts / checks number of 1's – Can be even or odd – If parity is incorrect, error is detected	**4**

Q19)

Question	Answer	Marks
(a)	– 1 – 0 – 0 – 0	**4**
(b)	**Two** from: – Checksum – Automatic repeat request // ARQ	**2**
(c)	Any **four** from: – Data is **input** with check digit – A calculation is performed on the (inputted) data // by example – The calculated digit is compared to a stored value – If it matches, the data entered is correct – If it does not match, the data entered is incorrect	**4**

Q20)

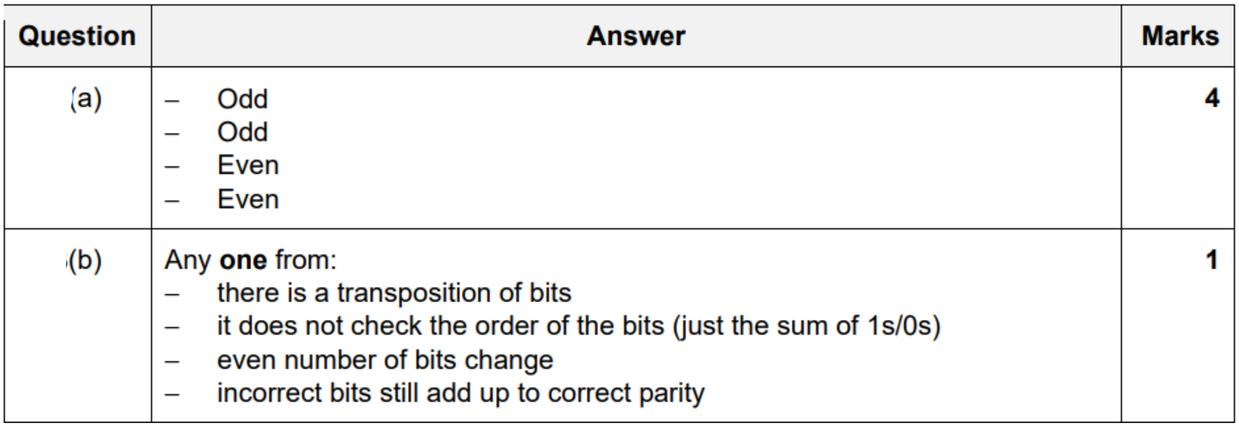
<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>**One** mark for each correct row:
<table>
<tr><th>8-bit binary value</th><th>Even (✓)</th><th>Odd (✓)</th></tr>
<tr><td>11111111</td><td>✓</td><td></td></tr>
<tr><td>01100110</td><td>✓</td><td></td></tr>
<tr><td>01111011</td><td>✓</td><td></td></tr>
<tr><td>10000000</td><td></td><td>✓</td></tr>
</table></td><td>**4**</td></tr>
<tr><td>(b)</td><td>Any **five** from:
– A value is calculated **from the data**
– The value is calculated **using an algorithm** // by example
– The value is appended to the data to be transmitted
– Value is recalculated after transmission
– Values are compared
– If the values match the data is correct // if the values do not match the data is incorrect</td><td>**5**</td></tr>
</table>

Q21)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td>**One** mark for each correct row:
<table>
<tr><th>8-bit binary value</th><th>Even (✓)</th><th>Odd (✓)</th></tr>
<tr><td>10000001</td><td>✓</td><td></td></tr>
<tr><td>10000010</td><td>✓</td><td></td></tr>
<tr><td>00101001</td><td></td><td>✓</td></tr>
<tr><td>00101000</td><td>✓</td><td></td></tr>
</table></td><td>**4**</td></tr>
<tr><td>(b)</td><td>Any **one** from:
– Transposition error
– When bits still add up to odd/even number
– **Even** number of incorrect bits</td><td>**1**</td></tr>
</table>

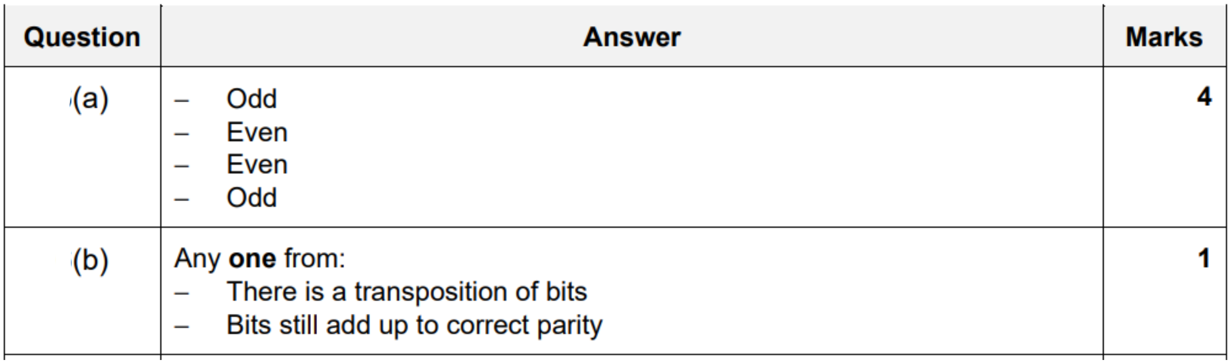
Question	Answer	Marks
(c)	Any **one** from: – ARQ – Checksum	**1**

Q22)



Question	Answer	Marks
(a)	– Odd – Odd – Even – Even	**4**
(b)	Any **one** from: – there is a transposition of bits – it does not check the order of the bits (just the sum of 1s/0s) – even number of bits change – incorrect bits still add up to correct parity	**1**

Q23)



Question	Answer	Marks
(a)	– Odd – Even – Even – Odd	**4**
(b)	Any **one** from: – There is a transposition of bits – Bits still add up to correct parity	**1**

Q24)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td><b>One</b> mark per each correct row.
<table>
<tr><th>Statement</th><th>Checksum (✓)</th><th>Check digit (✓)</th><th>Parity check (✓)</th></tr>
<tr><td>uses an additional bit to create an odd or even number of 1s</td><td></td><td></td><td>✓</td></tr>
<tr><td>checks for errors on data entry</td><td></td><td>✓</td><td></td></tr>
<tr><td>compares <b>two</b> calculated values to see if an error has occurred</td><td>✓</td><td>✓</td><td></td></tr>
<tr><td>will <b>not</b> detect transposition errors</td><td></td><td></td><td>✓</td></tr>
<tr><td>sends additional values when data is transmitted from one computer to another</td><td>✓</td><td></td><td>(✓)</td></tr>
</table>
</td><td>5</td></tr>
<tr><td>(b)</td><td>– ARQ</td><td>1</td></tr>
</table>

Q25)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>(a)</td><td><b>One</b> mark per each row
<table>
<tr><th>Statement</th><th>ARQ
(✓)</th><th>Check digit
(✓)</th><th>Checksum
(✓)</th></tr>
<tr><td>checks for errors on data entry</td><td></td><td>✓</td><td></td></tr>
<tr><td>uses a process of acknowledgement and timeout</td><td>✓</td><td></td><td></td></tr>
<tr><td>compares two calculated values to see if an error has occurred</td><td></td><td>✓</td><td>✓</td></tr>
<tr><td>may resend data until it is confirmed as received</td><td>✓</td><td></td><td></td></tr>
<tr><td>checks for errors in data after transmission from a computer to another</td><td></td><td></td><td>✓</td></tr>
</table>
</td><td><b>5</b></td></tr>
<tr><td>(b)</td><td>– Parity check</td><td><b>1</b></td></tr>
</table>

Q26)

Question	Answer	Marks
	– Even – Even – Odd – Even	**4**

Q27)

(b)	Any **four** from: – The number of 1 s/0 s are counted – A parity bit is added to each byte/7 bits before transmission – … to make the **sum** of the bits/1 or 0 s in each byte **odd** – After transmission, if the number is **odd** no error is detected – After transmission, if the number is **even** an error is detected	**4**
(c)	– Echo (check)	**1**

Q28)

Question	Answer	Marks
(a)	– Interference // crosstalk	**1**
(b)	– C	**1**
(c)	Any **five** from: – Timer is started when sending device transmits a data packet to receiver – Receiving device checks the data packet for errors – Once the receiving device knows the packet is error free it sends an acknowledgement back to the sending device … – … and the next packet is sent – If the sending device does not receive an acknowledgement before the timer ends … – … a timeout occurs – … the data packet is resent … – … until acknowledgement received // until max number of attempts reached	**5**

Q29)

(a)	Any **three** from: • Data could be lost • Data could be gained/added • Data could be changed • Bits could be **reassembled** in the wrong order • Interference could occur • Crosstalk could occur • Data collisions could occur • Data packets could time out/reach their hop count • **Network** could be infected with malware	**3**

Question	Answer	Marks
(b)(i)	Any **eight** from: • The 1s are counted (in each byte) • Each **byte** has a parity bit • If the number of 1s are odd the parity bit **is 0** (otherwise it **is 1**) • (The first packet of) data is sent and a timer is started • The **receiving** device counts the number of 1s (in each byte) • If the **number of 1s are odd/data meets odd parity** an acknowledgement is sent to say the data is error free … • … the sender then sends the next packet of data … • … and the timer is restarted • If the number of 1s is even an acknowledgement is not sent • If **no acknowledgement** is received **within a set timeframe/before timeout** … • … the data packet is resent	**8**
(b)(ii)	Any **two** from: • Echo check • Checksum • Even parity check • Negative ARQ	**2**

Q30)

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>-</td><td><b>One</b> mark for each correct method.
<table>
<tr><th>error detection method</th><th>statement</th></tr>
<tr><td>parity (check/bit/byte/block)</td><td>An odd or even process can be used.</td></tr>
<tr><td>checksum</td><td>A value is calculated from the data using an algorithm. This happens before and after the data is transmitted.</td></tr>
<tr><td>echo check</td><td>A copy of the data is sent back to the sender by the receiver.</td></tr>
<tr><td>automatic repeat query/request // ARQ</td><td>Acknowledgement and timeout are used.</td></tr>
<tr><td>check digit</td><td>A value is appended to data that has been calculated using the data. This value is checked on data entry.</td></tr>
</table>
</td><td><b>5</b></td></tr>
</table>