7	**One** mark for each correct data transmission method. • Serial • Simplex Any **three** from: Serial • Data will need to travel long distances • One wire is used // bits are sent in order // sent one bit at a time • Less chance of data being skewed // less chance of error • Transmission speed would be adequate Simplex • Data **only** needs to be sent in **one direction** • Data does not need to be **sent back** from the **speakers** to the microphone

4(a)(i)	**Two** from (one mark for serial, one mark for full-duplex): **serial** • Serial is suitable as data may be travelling a long distance (over 5 metres). • Serial is suitable to make sure that bits don't arrive skewed/arrives in order. • Serial is suitable as there is less chance of error. • Serial has sufficient transmission speed as not much data being sent. **full-duplex** • Full-duplex is suitable as data needs to go between customer and kitchen computer // data needs to go in both directions // so notification of errors can be sent back to customer.
4(a)(ii)	The data may be transmitted faster.
4(b)(i)	Any **five** from: • The checksum is **calculated** from/using the **data**. • … using an algorithm // by example. • The (checksum) value is transmitted with the data. • The (checksum) value is recalculated **after transmission**/**by recipient**. • If the **checksum values** do not match, an error is detected // If the **checksum values** match, no error is detected … • … request for data to be resent if there is an error.
4(b)(ii)	Any **one** from: • Parity byte/block check • Echo check
4(c)(i)	D
4(c)(ii)	Any **Two** from: • The packets **arrive** in the wrong order. • Each packet could take a different route. • Some packets may take longer than others to be transmitted. • Some packets may not arrive/be corrupted and need to be requested again.

3(a)	Any **two** from: • It is easier to debug. • Less likely to make errors. • The program is machine independent/portable
3(b)(i)	**One** mark for each correct term in the correct place. • whole code • executing • all • line by line • error A compiler translates the **whole code** at once before **executing** it. A compiler produces an error report that displays **all** errors. An interpreter translates and executes the code **line by line**. An interpreter stops execution when an **error** is found and continues once it is corrected.
3(b)(ii)	**One** mark for each correct function. **One** mark for each correct matching role description. Examples: • Code editor • Allows the programmer to write/change the program. • Run-time environment • Allows the user to run the code and see the output. • Error **diagnostics** • Features that can be used to find errors in the code • Auto-completion • A programmer starts to type a command word, and the IDE suggests **option** for completing it. • Auto-correction • If a programmer **misspells** a command word it is changed to the correct spelling • Prettyprint • The **command words**/**identifiers** are given different colours
3(c)(i)	Any **one** from: • Inkjet • Laser

3(c)(ii)	**One** mark from: • Serial • Parallel **One** mark from: • Half-duplex • Full-duplex • Simplex Any **four** from (for descriptions matching transmission types given): **serial** • Serial would send bits in order // serial uses only one wire. • … so won't be skewed // less likely to have errors • Serial transmission speed would be adequate. **parallel** • Parallel would transmit data faster. • … as multiple bits are sent at the **same time** // … as multiple wires are used. • For parallel, chance of **skewing/errors** would be low as **short distance** transmission only required. **half/full duplex** • To allow data to be sent in both directions • … so any interrupts/notifications for errors can be sent back to the computer. **simplex** • Data **only** needs to be sent one direction // Data transmission doesn't need to be two-way. • … as the printer may not need to send errors back to the computer

3(c)(iii)	• A parity bit is added to each **byte.** • … to make the number of 1s/0s even // that will be 1 if the number of 1s/0s is odd // that will be 0 if the number of 1s/0s is even. • The number of 1s/0s in each byte is counted **after transmission.** • If any bytes have an odd number of 1s/0s an error is detected