**Part-I: Theoretical Understanding of RNN, LSTM, & Encoder-Decoder**

Task 1: Conceptual Questions:

**Q1. What is the difference between RNN and LSTM?**

RNN struggles to remember long-term dependencies because of the vanishing gradient problem but LSTM is designed to remember information for longer periods using special units called memory cells and gates like input, forget, and output gates.

**Q2. What is the vanishing gradient problem, and how does LSTM solve it?**

The vanishing gradient problem occurs when gradients become extremely small during backpropagation, making it hard for the network to learn long-term dependencies. LSTM solves it by using special gates input, forget, and output that control the flow of information. These gates help the model remember important data for a long time and prevent it from forgetting.

**Q3. Explain the purpose of the Encoder-Decoder architecture.**

The Encoder-Decoder architecture is used in tasks like language translation. The encoder reads the input sentence and converts it into a fixed summary called the context vector. This context vector holds the meaning of the whole input. The decoder then uses this to generate the output sentence step by step.

**Q4. In a sequence-to-sequence model, what are the roles of the encoder and decoder?**

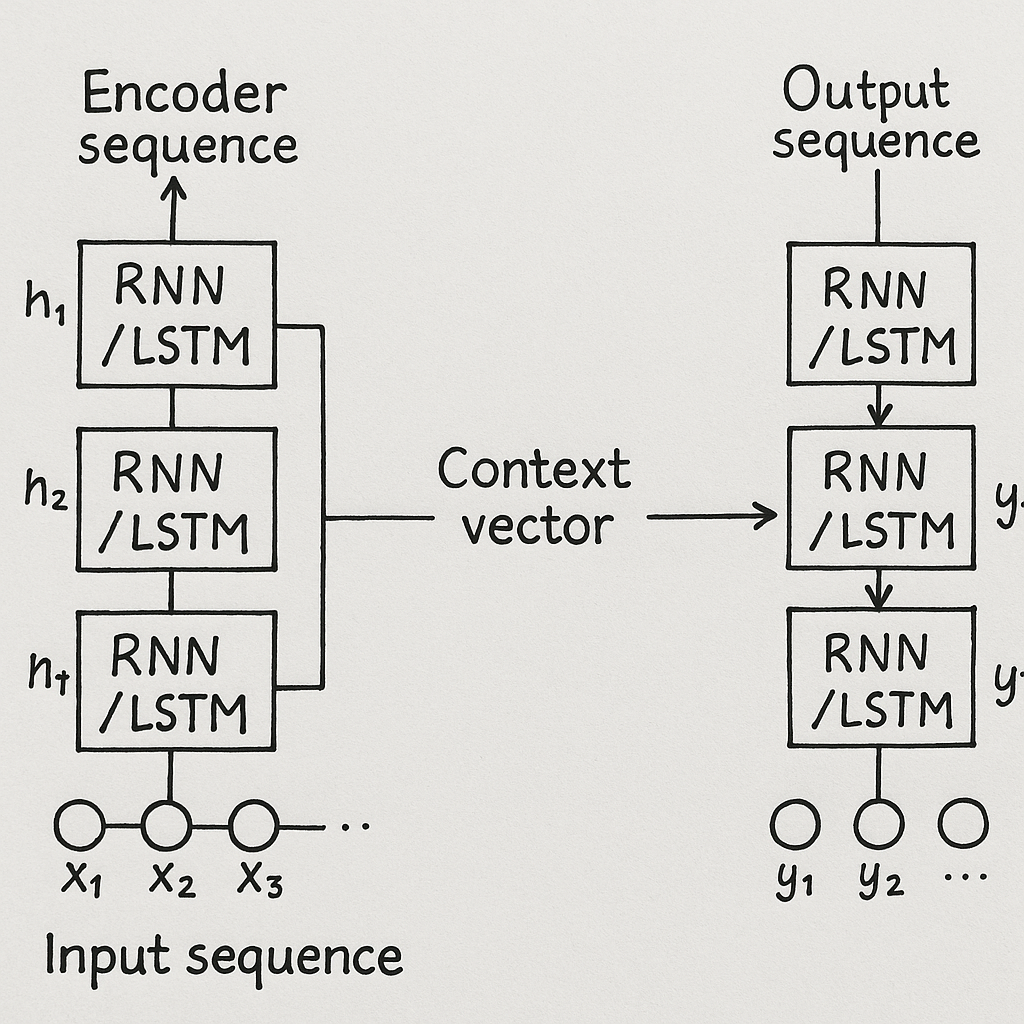
In a sequence-to-sequence model, the encoder reads the input sequence and understands its meaning. It then converts this information into a context vector. The decoder takes this context vector as input. It uses it to generate the output sequence one step at a time.

**Q5. How is attention different from a basic encoder-decoder model?**

In a basic encoder-decoder model, the decoder uses only the final context vector from the encoder. This can miss important details from long input sequences. Attention is different because it looks at all the encoder outputs while generating each word. It helps the decoder focus on the most relevant parts of the input at each step.

**Task 2: Sequence-to-Sequence Data Flow:**

The data flow in an encoder-decoder model



**Input sequence:** The original data or sentence given to the model, like a line in English for translation. The bottom left part labelled as x₁, x₂, x₃ as input.

**Hidden states:** These are memory values created by the encoder at each step to remember the input information. Labelled as h₁, h₂, ..., hₜ on the left beside each RNN/LSTM box.

**Context vector:** A summary of the whole input sequence that helps the decoder understand what to generate. The arrow in the center going from encoder to decoder labelled context vector

**Output sequence**: The final sentence or result produced by the decoder, like a translation in another language. The bottom right part labelled as y₁, y₂, ..., yₜ.