Questions on Network Topologies:

1. **Star Topology:**
    - Can you explain how the **star topology** is set up in Packet Tracer?
    - What are the advantages and disadvantages of using a star topology?
    - In a star topology, what happens if the central hub or switch fails?
2. **Bus Topology:**
    - How would you create a **bus topology** in Packet Tracer?
    - What are the limitations of a bus topology in real-world networking?
    - How does signal collision occur in a bus topology, and how is it managed?
3. **Ring Topology:**
    - Can you demonstrate how a **ring topology** works in Packet Tracer?
    - What is a key difference between the ring topology and the star topology in terms of data flow?
    - How does a ring topology handle data transmission failures?
4. **Mesh Topology:**
    - Show how to set up a **mesh topology** using Packet Tracer.
    - What are the main benefits of using a mesh topology in a network?
    - How does mesh topology enhance network reliability?
5. **Hybrid Topology:**
    - Can you give an example of a **hybrid topology** in Packet Tracer?
    - Why would a business choose to implement a hybrid topology rather than a single topology?

Questions on Transmission Media:

1. **Wired Transmission Media:**
    - Can you demonstrate the use of **twisted-pair cables** in Packet Tracer?
    - What are the key differences between **Cat5, Cat6,** and **Cat7 cables**?
    - How do **coaxial cables** compare to twisted-pair cables in terms of bandwidth and interference?
2. **Fiber Optic Cables:**
    - How would you set up a network using **fiber optic cables** in Packet Tracer?
    - What are the main advantages of fiber optics over copper cables?
    - Can you explain how **single-mode** and **multi-mode** fiber optic cables differ?

3. **Wireless Transmission:**
   - How would you configure a **wireless network** in Packet Tracer?
   - What are the advantages of using wireless media over wired media in a LAN?
   - How do wireless signals suffer from interference, and how can you reduce it?
4. **Bluetooth and Infrared:**
   - Can you explain how **Bluetooth** is simulated in Packet Tracer?
   - How does **infrared transmission** differ from other wireless methods in terms of usage and range?
5. **Comparison:**
   - Can you compare the strengths and weaknesses of wired and wireless transmission media in terms of cost, speed, and reliability?
   - How would you decide which transmission media is suitable for a particular network scenario?

General/Follow-up Questions:

- Why is it important to consider network topology when designing a network?
- How do different types of transmission media affect network performance?
- Can you discuss some real-world examples where certain topologies or transmission media would be most appropriate?
- How does Packet Tracer help in visualizing and simulating these topologies and media?

Questions on WAN Setup:

1. **Basic WAN Configuration:**
   - Can you explain how you set up a **WAN** in Packet Tracer that connects both wired and wireless LANs?
   - What devices are necessary to create a WAN that connects multiple LANs?
   - How do you configure routers in Packet Tracer for communication between LAN1 and LAN2?
2. **Router and Switch Configuration:**
   - How did you configure the **router interfaces** to connect the wired LAN (LAN1) and wireless LAN (LAN2)?
   - What commands did you use to configure the **routing tables** on the router to ensure packet transfer between the two LANs?

- Can you explain the difference between **static routing** and **dynamic routing** in this context? Which one did you use?

3. **Subnetting and IP Addressing:**
   - How did you assign IP addresses to the devices in both LAN1 (wired) and LAN2 (wireless)?
   - What subnetting scheme did you use to divide the network into different segments for LAN1 and LAN2?
   - Why is it necessary to use subnetting when setting up a WAN?

Questions on Wired LAN (LAN1) Setup:

1. **Wired LAN Configuration:**
   - How did you configure the **wired LAN** (LAN1) in Packet Tracer?
   - What type of cables did you use for connecting the devices in LAN1?
   - Why did you choose the specific network topology (e.g., star, bus) for the wired LAN?
2. **Switch Configuration in LAN1:**
   - How did you configure the **switch** in LAN1 to ensure proper communication among wired devices?
   - Can you explain the role of **VLANs** in the wired LAN setup, if used?
3. **Testing Connectivity in LAN1:**
   - How did you test the communication between devices within LAN1?
   - What Packet Tracer tools did you use to verify connectivity within the wired LAN?

Questions on Wireless LAN (LAN2) Setup:

1. **Wireless LAN Configuration:**
   - How did you configure the **wireless LAN** (LAN2) in Packet Tracer?
   - Which **wireless access points** did you use, and how were they configured to ensure connectivity?
   - What security measures did you apply to the wireless LAN (e.g., WPA2, MAC filtering)?
2. **Device Connectivity in LAN2:**
   - How did you ensure that devices within the wireless LAN are properly connected and configured to communicate with each other?
   - How did you assign IP addresses to the devices in the wireless LAN, and how are they different from the wired LAN?
3. **Testing Connectivity in LAN2:**

- How did you verify that devices in LAN2 can communicate with each other?
- What Packet Tracer simulation tools did you use to check the wireless signal strength and device connections?

Questions on Packet Transfer Between LAN1 and LAN2:

1. **Demonstrating Packet Transfer:**
   - Can you demonstrate the transfer of a packet from a device in **LAN1 (wired)** to a device in **LAN2 (wireless)** using Packet Tracer?
   - What steps did you follow to ensure that the packet reaches its destination in the wireless LAN?
2. **Troubleshooting Packet Transfer:**
   - What tools in Packet Tracer did you use to verify the successful transfer of the packet between the two LANs?
   - If the packet transfer fails, what troubleshooting steps would you take to resolve the issue?
3. **Routing and Switching:**
   - How does the router in the WAN handle the packet transfer between the wired LAN and the wireless LAN?
   - What is the role of the **switch** and **access point** in ensuring that the packet is successfully delivered to the wireless LAN?
4. **Protocols and Packet Inspection:**
   - Which network protocols (e.g., TCP/IP, ICMP) are involved in the transfer of the packet between LAN1 and LAN2?
   - How can you inspect the packet details in Packet Tracer to see the source and destination IP addresses?

General and Advanced Questions:

1. **WAN Architecture:**
   - Can you explain the role of WAN in connecting multiple LANs and how it differs from a LAN-to-LAN connection?
   - What are the challenges of managing a WAN with both wired and wireless networks in a real-world scenario?
2. **Security Considerations:**
   - How would you ensure security during the packet transfer between wired and wireless networks?
   - What additional security configurations would you recommend for this setup?
3. **Scaling the Network:**

- How would you modify the WAN to include additional LANs in different geographical locations?
- How would adding more wireless access points or switches affect the overall network performance?

General Questions on Error Detection and Correction:

1. **Conceptual Questions:**
   - What is the purpose of **error detection** and **error correction** in data communication?
   - Can you explain the difference between **error detection** and **error correction**?
   - Why is it important to apply error detection and correction techniques in communication systems?
2. **ASCII Codes:**
   - Why are 7-bit or 8-bit **ASCII codes** commonly used in error detection and correction algorithms?
   - How are characters represented in the **ASCII** encoding scheme?
   - What is the difference between **7-bit** and **8-bit** ASCII codes, and why would you use one over the other?

Questions on Hamming Codes:

1. **Basic Hamming Code Concepts:**
   - What is the principle behind **Hamming Codes** for error detection and correction?
   - How does a **Hamming Code** detect and correct single-bit errors?
   - Can you explain the significance of **parity bits** in Hamming Codes?
2. **Hamming Code Calculation:**
   - How do you calculate the number of **parity bits** needed for a given ASCII code?
   - Can you demonstrate how to insert parity bits in an 8-bit ASCII code using Hamming Code?
   - How do you detect and correct an error in a received ASCII code using Hamming Code?
3. **Program-Specific Questions (Hamming Code):**
   - Can you explain the steps your program follows to encode an ASCII character using Hamming Code?
   - How does your program detect an error in the transmitted code, and how does it locate the position of the error?

- If an error is found, how does your program correct the error in the received ASCII code?

4. **Advanced Questions:**
   - What types of errors can **Hamming Codes** correct, and what are their limitations?
   - How would you modify the Hamming Code algorithm if you needed to detect and correct **double-bit errors**?

Questions on Cyclic Redundancy Check (CRC):

1. **Basic CRC Concepts:**
   - What is the role of **Cyclic Redundancy Check (CRC)** in error detection?
   - Can you explain the mathematical principle behind CRC, especially how **polynomials** are used?
   - Why is CRC considered more reliable for **error detection** than simpler methods like parity checks?
2. **CRC Calculation:**
   - How do you generate a **CRC code** for a given 7/8-bit ASCII message?
   - Can you explain how the **CRC generator polynomial** is used to divide the data bits to produce the CRC checksum?
   - How is the remainder from the division process used to detect errors in the received message?
3. **Program-Specific Questions (CRC):**
   - How does your program generate the **CRC code** for a given ASCII message?
   - Can you demonstrate how your program detects an error in the received ASCII code using CRC?
   - What steps does your program follow if an error is detected in the received code?
4. **Advanced CRC Questions:**
   - What are the common **CRC polynomials** used in network communications, and how do they differ?
   - How does CRC differ from Hamming Code in terms of both **error detection** and **error correction**?
   - Can CRC correct errors, or is it purely an error detection method?

Comparison and Application Questions:

1. **Comparison Between Hamming Codes and CRC:**

- Can you compare **Hamming Codes** and **CRC** in terms of their efficiency in error detection and correction?
- In what situations would you prefer using **Hamming Codes** over **CRC**, and vice versa?

2. **Real-World Applications:**
   - Where are **Hamming Codes** and **CRC** typically used in real-world systems?
   - How would you decide which error detection and correction technique to use in a network or communication protocol?
   - Can you provide examples of how **Hamming Codes** or **CRC** are used in computer networks or data storage systems?

Program Debugging and Optimization:

1. **Debugging and Testing:**
   - How did you test your program to ensure that it correctly detects and corrects errors?
   - What were the most challenging parts of implementing Hamming Codes or CRC in your program?
2. **Optimization:**
   - How would you optimize your program for faster **error detection and correction**?
   - What could be the impact of large data sets on the performance of your error detection and correction program?

Basic Conceptual Questions:

1. **Sliding Window Protocol:**
   - What is the **Sliding Window Protocol**, and why is it used in data communication?
   - Can you explain the difference between **flow control** and **error control** in the Sliding Window Protocol?
   - How does the size of the sliding window affect the performance of a network protocol?
2. **Go-Back-N Protocol:**
   - What is the **Go-Back-N (GBN)** mode in the Sliding Window Protocol?
   - How does the sender manage **unacknowledged frames** in Go-Back-N?
   - What happens when a packet is lost or corrupted in Go-Back-N, and how does the protocol handle retransmission?
3. **Selective Repeat Protocol:**

- How does the **Selective Repeat (SR)** mode differ from Go-Back-N?
- What are the advantages of using Selective Repeat over Go-Back-N?
- How are **out-of-order frames** handled in Selective Repeat, and how does the receiver manage the buffer?

Programming-Specific Questions:

1. **General Program Flow:**
   - Can you explain the general flow of your program in simulating the Sliding Window Protocol in both **Go-Back-N** and **Selective Repeat** modes?
   - How did you simulate **peer-to-peer communication** in your program?
2. **Window Size:**
   - How did you implement the concept of a **sliding window** in your program?
   - How does the window size affect the performance of your program in both Go-Back-N and Selective Repeat?
   - Can you show how the window shifts in both protocols when acknowledgments are received?
3. **Timeout and Retransmissions:**
   - How did you implement **timeouts** in your Go-Back-N and Selective Repeat simulations?
   - What mechanism does your program use to retransmit lost or corrupted frames in Go-Back-N mode?
   - How does your program handle **individual retransmissions** in Selective Repeat?

Go-Back-N Specific Questions:

1. **GBN Sender and Receiver:**
   - How did you simulate the behavior of the **sender** and **receiver** in Go-Back-N mode?
   - Can you explain how your program handles the **ACKs** (acknowledgments) in Go-Back-N? What happens when an acknowledgment is lost?
   - How does the sender react when a packet or acknowledgment is lost in Go-Back-N?
2. **Retransmission in GBN:**
   - What triggers the **retransmission** of packets in Go-Back-N mode in your program?
   - How does the program handle the retransmission of packets after a **timeout** in Go-Back-N?

Selective Repeat Specific Questions:

1. **SR Sender and Receiver:**
   - How did you implement the sender and receiver logic for the **Selective Repeat** mode?
   - In Selective Repeat, how does the receiver handle **out-of-order frames**, and how did you implement this in your program?
   - How does the **receiver buffer** work in Selective Repeat, and how do you ensure that frames are delivered in the correct order?
2. **Acknowledgment Handling in SR:**
   - How does your program send **individual ACKs** for each correctly received frame in Selective Repeat?
   - What happens when an acknowledgment is lost in Selective Repeat, and how does the sender react?
3. **Timeout and Retransmission in SR:**
   - How did you implement **timeouts** and **retransmissions** in Selective Repeat mode?
   - What happens if a packet is lost and not retransmitted in Selective Repeat? How does the receiver behave in such cases?

Comparison Between Go-Back-N and Selective Repeat:

1. **Efficiency and Performance:**
   - Which protocol, Go-Back-N or Selective Repeat, is more efficient in terms of network utilization? Why?
   - What are the **trade-offs** between Go-Back-N and Selective Repeat in terms of **buffer size** and **bandwidth usage**?
   - How does **error rate** (packet loss) affect the performance of both Go-Back-N and Selective Repeat protocols?
2. **Handling Lost Packets:**
   - Can you explain how Go-Back-N handles **lost or corrupted packets** differently from Selective Repeat?
   - How does the **retransmission strategy** differ between Go-Back-N and Selective Repeat?
3. **Complexity and Implementation:**
   - Which protocol was more challenging to implement in your program, Go-Back-N or Selective Repeat? Why?
   - How does the **computational complexity** of Selective Repeat compare to Go-Back-N?

Advanced and Real-World Application Questions:

1. **Real-World Use:**
    - Where are **Go-Back-N** and **Selective Repeat** protocols used in real-world communication systems?
    - Why might **Selective Repeat** be preferred in some real-world applications over Go-Back-N?
2. **Scaling and Optimization:**
    - How would your program handle **larger window sizes** or **higher data rates**? Would you need to optimize it for efficiency?
    - What could be the impact of **network congestion** on the performance of Go-Back-N and Selective Repeat?
3. **Handling High Latency:**
    - How would **high latency** networks (such as satellite communication) affect the performance of Go-Back-N and Selective Repeat protocols?
    - How would you modify your program to improve performance in a high-latency environment?

Troubleshooting and Debugging:

1. **Debugging:**
    - What challenges did you face while debugging the **Go-Back-N** and **Selective Repeat** implementations?
    - How did you test your program to ensure that it correctly simulates the behavior of both protocols?
    - What tools or techniques did you use to simulate **packet loss**, **timeouts**, and **ACK loss** in your program?
2. **Common Issues:**
    - What are some common issues you encountered while implementing **sliding window mechanisms**, and how did you resolve them?
    - How did you ensure that the program handles **all possible edge cases**, such as multiple packet losses or multiple ACK losses?

Basic Conceptual Questions on Subnetting:

1. **What is Subnetting?**
    - Can you explain what **subnetting** is and why it is important in network design?

- How does subnetting help in reducing network congestion and improving performance?
- What are the main advantages of using subnetting in large networks?

2. **IP Addressing:**
   - Can you explain the structure of an **IP address** (IPv4)?
   - What is the difference between **Class A, B, and C** IP addresses in terms of default subnet masks?
   - What is the role of **network** and **host** portions in an IP address?
3. **Subnet Masks:**
   - What is a **subnet mask**, and how does it relate to an IP address?
   - How do you calculate the subnet mask for a given network based on the number of required subnets or hosts?
   - What is the significance of **CIDR notation** (e.g., /24) in subnetting?

Questions on Subnetting Calculations:

1. **Subnetting Basics:**
   - How do you determine the number of **subnets** and **hosts** per subnet when given a network address and subnet mask?
   - If you are given an IP address and a **subnet mask**, how do you find the **network address** and **broadcast address**?
   - Can you explain how you calculate the **first usable IP address** and the **last usable IP address** in a subnet?
2. **Example Calculation:**
   - If you have a network address of **192.168.1.0/24**, how would you divide it into **4 subnets**? What would be the new subnet mask?
   - Given an IP address of **172.16.5.33/16**, how would you calculate the **subnet mask** for creating 64 subnets?
   - If you are assigned **10.0.0.0/8**, how many hosts can you assign with a **/24** subnet mask?

Program-Specific Questions:

1. **Program Logic:**
   - Can you explain the logic of your program for calculating **subnets** and **subnet masks**?

- What inputs does your program require, and how does it process these inputs to generate subnets?
- How did you implement the conversion between **decimal** and **binary** for IP addresses and subnet masks?

2. **Subnet Mask Calculation:**
   - How does your program calculate the **subnet mask** based on the number of required subnets or hosts?
   - Can your program handle both **classful** and **classless** IP addressing schemes? If so, how does it differentiate between them?
   - How does the program display the **subnet mask** in both **dotted decimal** and **CIDR** notation?
3. **Generating Subnets:**
   - How does your program generate a list of **subnet addresses** and **broadcast addresses** for each subnet?
   - How does your program ensure that all subnets are calculated correctly without overlap?
   - Does your program display the **range of usable IP addresses** in each subnet? If yes, how?

Practical Application and Real-World Questions:

1. **Real-World Application:**
   - In what situations would you use **subnetting** in a real-world network?
   - Can you describe how subnetting is used in large organizations or **Internet Service Providers (ISPs)**?
   - What are the potential consequences of poor subnetting design in a network?
2. **Scaling and Optimization:**
   - How would you subnet a large network to accommodate both **internal** and **public-facing** services?
   - How would you handle **VLSM (Variable Length Subnet Mask)** in your program, and why is it important in real-world scenarios?
   - How does subnetting improve the **efficiency** and **security** of a network?

Advanced and Troubleshooting Questions:

1. **Subnetting Edge Cases:**

- How does your program handle edge cases, such as when there are too few or too many **subnets** or **hosts**?
- What happens if the input subnet size exceeds the total number of available addresses in the network?

2. **IPv6 Subnetting:**
    - Can your program be adapted to handle **IPv6 subnetting**? If so, what are the key differences compared to IPv4 subnetting?
    - How would you calculate subnets and subnet masks for an IPv6 network, given that it uses a much larger address space?
3. **Program Debugging:**
    - What were the most challenging parts of implementing subnetting logic in your program?
    - How did you test your program to ensure that it calculates **subnet masks** and **IP ranges** accurately?
    - What were some common mistakes or errors you encountered during the development of the program, and how did you resolve them?

Comparative Questions:

1. **Static vs Dynamic Subnetting:**
    - What is the difference between **static** and **dynamic** subnetting? How would your program handle both?
    - How does **DHCP** fit into the context of subnetting, and how does it automate IP address allocation?
2. **Subnetting vs Supernetting:**
    - Can you explain the difference between **subnetting** and **supernetting** (also known as route aggregation)?
    - In what situations would you use **supernetting**, and how does it differ from traditional subnetting?

Optimization and Future Enhancements:

1. **Optimizing the Program:**
    - How would you optimize your program to handle large networks with hundreds or thousands of subnets?
    - Can your program handle **network address translation (NAT)**, and if not, how would you extend it to support NAT?
2. **Future Enhancements:**

- What additional features or enhancements could you add to the program, such as visualizing the subnetting process or adding support for **IPv6**?
- How would you integrate the program with **real-time network tools** to automatically configure network devices based on the calculated subnets?

Basic Conceptual Questions:

1. **Routing Protocol Basics:**
    - What is the purpose of a **routing protocol** in a network?
    - Can you explain the difference between **routing** and **switching**?
    - What is the difference between **static routing** and **dynamic routing**?
2. **Link State Routing Protocol:**
    - What is the **Link State Routing Protocol**, and how does it work?
    - Can you explain the key steps in the **Link State** routing process, such as link-state advertisement (LSA) and path calculation using **Dijkstra's algorithm**?
    - How does a router maintain its **link-state database**, and how often does it update the network topology?
3. **Distance Vector Routing Protocol:**
    - What is the **Distance Vector Routing Protocol**, and how does it differ from Link State routing?
    - Can you explain how routers use the **Bellman-Ford algorithm** in Distance Vector routing?
    - What is meant by the term **split horizon**, and how does it prevent routing loops in Distance Vector protocols?

Algorithm-Specific Questions:

1. **Dijkstra's Algorithm (Link State):**
    - How does **Dijkstra's algorithm** find the shortest path in a network?
    - Can you explain the data structures used in your program to implement **Dijkstra's algorithm**?
    - How does Dijkstra's algorithm handle **changes in the network topology**, such as a link failure?
2. **Bellman-Ford Algorithm (Distance Vector):**
    - How does the **Bellman-Ford algorithm** calculate the shortest path in a network?

- Can you explain the process of **iterative distance updates** in the Distance Vector protocol?
- What are the limitations of the Bellman-Ford algorithm, particularly in detecting routing loops?

Program-Specific Questions:

1. **Program Design:**
   - Can you explain the overall structure of your program for implementing the routing protocol (Link State or Distance Vector)?
   - What input does your program take (e.g., network topology, link costs), and how does it process this input to calculate routes?
   - How does your program represent the **network graph**? Are you using adjacency matrices or adjacency lists?
2. **Link State Protocol (Program-Specific):**
   - How does your program handle the process of **link-state advertisements (LSAs)**?
   - Can you explain how the **Shortest Path Tree (SPT)** is built in your program using Dijkstra's algorithm?
   - How does your program react when a **link or router** goes down? How does it update the routing tables?
3. **Distance Vector Protocol (Program-Specific):**
   - How did you implement the **iterative distance updates** in the Distance Vector routing protocol?
   - How does your program ensure that **routing loops** are prevented, especially in the case of **count-to-infinity** problems?
   - What mechanisms did you include in the program for **route advertisement** and for detecting invalid routes?

Routing Table and Path Calculation:

1. **Routing Table Updates:**
   - How does your program build and update the **routing table** for each node in the network?
   - How does your program ensure that the **routing tables** are consistent across all nodes in the network?
   - Can you explain how your program selects the **best path** for a packet, based on metrics such as **hop count** or **link cost**?
2. **Path Calculation:**

- How does your program calculate the **shortest path** or **least-cost path** between two nodes?
- Can you explain how your program visualizes or outputs the calculated paths, and how it handles multiple possible paths?
- How does your program handle **dynamic changes** in the network, such as adding or removing links or nodes?

Error Handling and Efficiency:

1. **Error Handling:**
    - How does your program handle errors such as **unreachable destinations** or **link failures**?
    - What happens if your program encounters a **loop** in the network topology? How does it detect and resolve this?
    - How do you handle situations where the network becomes **disconnected**, and some nodes are isolated?
2. **Program Efficiency:**
    - How did you ensure that the routing algorithm is efficient in terms of both **time** and **space** complexity?
    - Can your program scale to handle large networks with many nodes and links? What optimizations did you implement for scalability?
    - How did you test your program to ensure it performs correctly under different network topologies and conditions?

Comparison and Protocol Characteristics:

1. **Comparison Between Link State and Distance Vector:**
    - What are the key differences between the **Link State** and **Distance Vector** routing protocols?
    - In what scenarios would you prefer to use **Link State routing** over Distance Vector, and vice versa?
    - Which protocol is more suitable for **large-scale networks** and why?
2. **Convergence and Stability:**
    - How does **convergence time** differ between the Link State and Distance Vector protocols?
    - What factors affect the **stability** of each protocol in dynamic or changing network environments?

- How do both protocols handle **routing loops**, and which is better at preventing them?

Advanced and Real-World Questions:

1. **Real-World Applications:**
    - Where are **Link State** and **Distance Vector** protocols used in real-world networks? Can you name some protocols that are based on these algorithms (e.g., OSPF, RIP)?
    - Why would an organization prefer to use **OSPF (Open Shortest Path First)** or **EIGRP** (Enhanced Interior Gateway Routing Protocol) over simpler Distance Vector protocols like RIP?
    - How do these routing protocols integrate with **interior gateway protocols (IGPs)** and **exterior gateway protocols (EGPs)**?
2. **Protocol Limitations:**
    - What are the limitations of **Distance Vector** and **Link State** protocols, especially in highly dynamic or large-scale networks?
    - How do advanced features like **hierarchical routing** and **area-based routing** improve the efficiency of Link State protocols like OSPF?
3. **Optimization and Improvements:**
    - How would you optimize your program to handle **frequent network changes** more efficiently?
    - Can you think of any additional features that could be added to your program to improve its performance or usability (e.g., load balancing, fault tolerance)?
    - How would you adapt your program to work with **real-world network devices**, such as routers and switches?

Troubleshooting and Debugging:

1. **Debugging Challenges:**
    - What challenges did you face while debugging the implementation of **Dijkstra's algorithm** or the **Bellman-Ford algorithm**?
    - How did you ensure that the routing calculations in your program are correct? Did you use any specific test cases or tools for verification?
    - What were the most common mistakes you encountered while developing the program, and how did you fix them?
2. **Testing and Validation:**
    - How did you test the **convergence** and **stability** of the routing protocol in your program?

- Did you simulate different network scenarios (e.g., link failures, topology changes) to validate the robustness of your implementation?
