Câu 1. Định nghĩa công nghệ Blockchain và các đặc điểm chính

Định nghĩa:

Blockchain là một sổ cái phân tán (distributed ledger) lưu trữ thông tin trong các khối (blocks) được liên kết với nhau bằng mật mã học, đảm bảo dữ liệu không thể bị sửa đổi sau khi ghi vào chuỗi.

Đặc điểm giúp Blockchain tin cậy và khác biệt so với cơ sở dữ liệu truyền thống:

Đặc điểm	Mô tả
Phân tán (Decentralized)	Dữ liệu không nằm ở một máy chủ trung tâm mà được chia sẻ giữa các nút (nodes).
Bất biến (Immutable)	Một khi dữ liệu được ghi vào khối, không thể bị sửa đổi nếu không có sự đồng thuận.
Minh bạch (Transparent)	Giao dịch công khai (trên Blockchain công khai) và có thể được kiểm tra bởi bất kỳ ai.
Bảo mật (Secure)	Dữ liệu được mã hóa bằng mật mã học (ví dụ: SHA-256).
Đồng thuận (Consensus)	Giao dịch chỉ được xác nhận khi có sự đồng thuận từ đa số các nút.

Câu 2. So sánh mô hình Client/Server với Peer-to-Peer trong Blockchain

Tiêu chí	Client/Server	Peer-to-Peer (P2P)
Cấu trúc	Tập trung: 1 server nhiều client	Phi tập trung, mọi node ngang hàng
Dễ triển khai	Dễ triển khai, dễ kiểm soát	Phức tạp hơn, khó quản lý
Tính sẵn sàng	Server lỗi → hệ thống ngừng	Một node lỗi không ảnh hưởng hệ thống
Bảo mật	Dễ bị tấn công DDoS vào server	Khó tấn công toàn bộ mạng
Ứng dụng Blockchain	Không phù hợp với bản chất phân tán	Cốt lõi của Blockchain là P2P

Kết luận: Mô hình P2P là nền tảng phù hợp nhất cho Blockchain do tính phân tán và không phụ thuộc vào trung tâm.

Câu 3. Cấu trúc của một khối (block) trong Blockchain

Một block gồm 2 phần chính:

1. Block Header (Phần đầu):

- Hash của khối trước: Liên kết chuỗi khối.
- Timestamp: Thời gian tạo khối.
- Nonce: Giá trị để tìm hash hợp lệ (trong PoW).
- Merkle Root: Hash tổng hợp của tất cả giao dịch.
- Version, Difficulty (mức độ khó),...

2. Block Body (Phần dữ liệu):

- Danh sách các giao dịch (transactions) đã xác nhận.

Vai trò:

- Header: đảm bảo tính toàn vẹn, liên kết giữa các khối.
- Body: chứa dữ liệu thực tế (giao dịch).

Câu 4. So sánh các giao thức đồng thuận: PoW, PoS, PBFT

Tiêu chí	PoW (Proof of Work)	PoS (Proof of Stake)	PBFT (Practical Byzantine Fault Tolerance)
Cách hoạt động	Cạnh tranh giải bài toán tính toán	Chọn ngẫu nhiên theo số lượng coin nắm giữ	Nút trao đổi để đạt đồng thuận
Tốn năng lượng	Rất cao	Thấp	Thấp
Tốc độ xử lý	Chậm	Nhanh hơn PoW	Nhanh nhất
Khả năng mở rộng	Thấp	Trung bình	Thấp với số node lớn
Ứng dụng	Bitcoin, Ethereum (cũ)	Ethereum (hiện tại), Cardano	Hyperledger, hệ thống doanh nghiệp
Ưu điểm	Bảo mật tốt, chống gian lận	Tiết kiệm năng lượng	Hiệu quả, nhanh
Nhược điểm	Tốn tài nguyên, chậm	Rủi ro tập trung coin	Không mở rộng tốt cho mạng lớn

Câu 5. Tính toàn vẹn dữ liệu và vai trò của SHA-256

- Toàn vẹn dữ liệu đạt được do:
    - Mỗi khối chứa hash của khối trước → thay đổi bất kỳ sẽ làm sai toàn bộ chuỗi.
    - Dữ liệu giao dịch mã hóa → phát hiện được mọi chỉnh sửa.
- SHA-256:
    - Là thuật toán băm tạo ra chuỗi 256-bit từ bất kỳ dữ liệu nào.
    - Không thể đảo ngược, và chỉ cần thay đổi nhỏ trong đầu vào → đầu ra thay đổi hoàn toàn.

Vai trò trong Blockchain:

- Băm nội dung khối → tạo mã định danh duy nhất.
- Dùng trong xác minh giao dịch, Merkle Tree, tạo Nonce hợp lệ (trong PoW).

Câu 6. Smart Contract là gì? Ứng dụng thực tế

Smart Contract là chương trình tự động thực thi trên Blockchain khi điều kiện được đáp ứng, không cần bên thứ ba.

Ứng dụng thực tế:

Lĩnh vực	Ví dụ
Tài chính (DeFi)	Giao dịch tự động, cho vay ngang hàng
Bảo hiểm	Tự động chi trả khi xảy ra sự kiện
Logistics	Theo dõi chuỗi cung ứng
Bất động sản	Hợp đồng mua bán không cần công chứng

Ví dụ: Khi khách hàng thanh toán tiền đặt cọc, smart contract tự động chuyển quyền sở hữu tài sản số mà không cần người trung gian.

Câu 7. Các bước triển khai Smart Contract bằng Web3.py

1. Cài đặt thư viện:
2. pip install web3
3. Kết nối tới mạng Ethereum:
4. from web3 import Web3
5. w3 = Web3(Web3.HTTPProvider("http://localhost:8545"))
6. Biên dịch hợp đồng (Solidity) → ABI và Bytecode (dùng solc hoặc remix).
7. Tải ABI và Bytecode vào Python.
8. Triển khai hợp đồng:
9. Contract = w3.eth.contract(abi=abi, bytecode=bytecode)
10. tx_hash = Contract.constructor().transact({'from': w3.eth.accounts[0]})
11. Chờ xác nhận giao dịch và lấy địa chỉ hợp đồng:
12. tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)
13. contract_instance = w3.eth.contract(address=tx_receipt.contractAddress, abi=abi)
14. Gọi hàm trong hợp đồng (đọc/gửi giao dịch).

Câu 8. Tích hợp Blockchain để xác minh nguồn gốc dữ liệu đầu vào trong Khoa học dữ liệu

Mục tiêu: đảm bảo dữ liệu đầu vào không bị chỉnh sửa hoặc làm giả.

Quy trình đề xuất:

1. Thu thập dữ liệu từ nguồn.
2. Tính băm (hash) dữ liệu đầu vào bằng SHA-256.
3. Ghi hash vào Blockchain kèm timestamp.
4. Mỗi lần truy cập/kiểm tra: so sánh hash hiện tại với hash đã ghi trên chuỗi → phát hiện chỉnh sửa.

Ứng dụng: truy xuất nguồn gốc dữ liệu y tế, tài chính, môi trường, chống sửa đổi sau khi thu thập.

Câu 9. Lỗi “invalid opcode” khi triển khai Smart Contract

Nguyên nhân có thể:

- Dùng phiên bản compiler sai so với hợp đồng.
- Viết code sai cú pháp Solidity hoặc logic không hợp lệ.
- Gọi hàm không tồn tại hoặc dùng biến chưa khởi tạo.
- Gửi sai tham số khi gọi hàm từ Web3.
- Lỗi quá giới hạn gas hoặc thiếu ether khi triển khai.

Cách khắc phục:

1. Kiểm tra kỹ mã Solidity: xác minh lại logic, phiên bản Solidity tương thích.
2. Kiểm tra ABI, Bytecode và cách gọi từ Web3.py.
3. Dùng Remix IDE để thử triển khai — dễ thấy lỗi cụ thể.
4. Kiểm tra logs trên ganache hoặc Etherscan (nếu trên testnet).
5. Đảm bảo tài khoản có đủ ETH để trả phí gas.