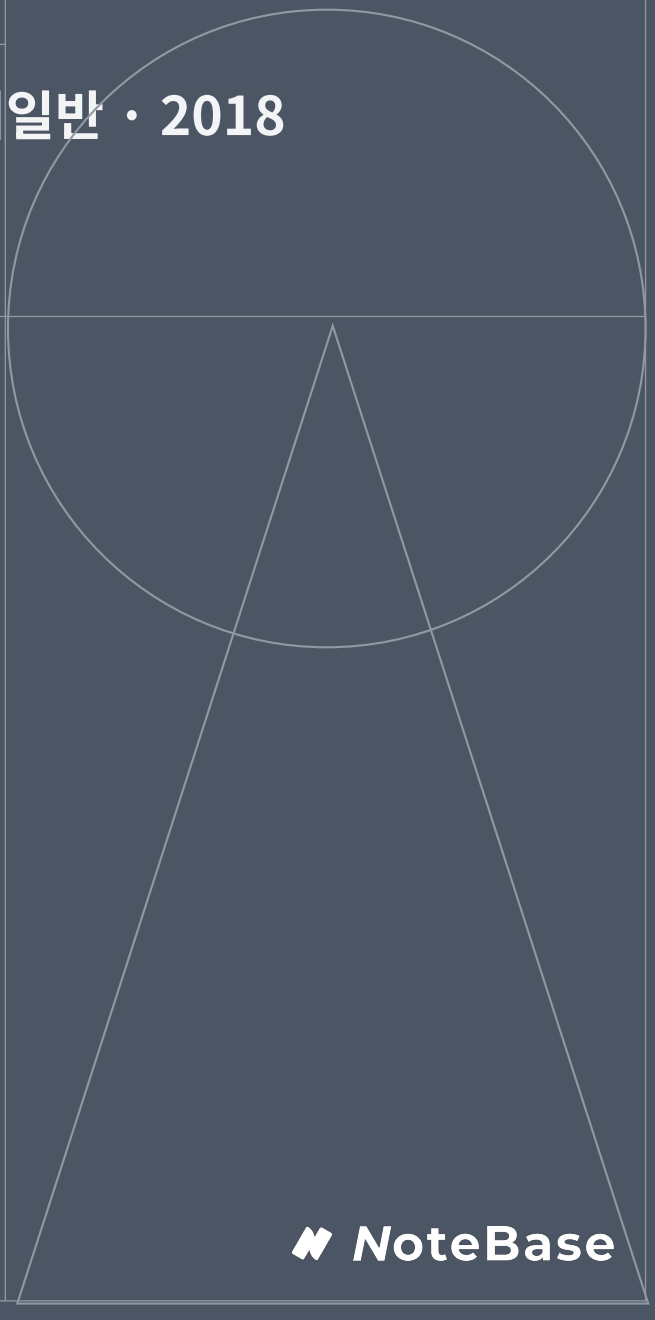
기출OX 해설집

공무원/국회직/9급 · 컴퓨터일반 · 2018



NoteBase

기출OX 해설집

공무원/국회직/9급 · 컴퓨터일반 · 2018

공무원/국회직/9급 · 컴퓨터일반 · 2018

1. 데이터베이스에서 트랜잭션에 대한 설명 중 다음의 특성이 의미하는 것으로 옳은 것은?

> 각 트랜잭션은 다른 트랜잭션과 독립적으로 수행되는 것처럼 보여야 하며, 다른 트랜잭션에 영향을 미치지 않는다.

① 고립성(Isolation)
② 일관성(Consistency)
③ 원자성(Atomicity)
④ 지속성(Durability)
⑤ 투명성(Transparency)

정답 1번

출제 포인트

이 문제는 트랜잭션의 ACID 특성 중 여러 트랜잭션이 동시에 실행될 때 서로 간섭하지 않도록 하는 성질을 묻는다.

핵심 개념

원자성(Atomicity)은 트랜잭션의 모든 연산이 전부 수행되거나 전혀 수행되지 않아야 한다는 성질이다.
일관성(Consistency)은 트랜잭션 수행 전후에 데이터베이스가 정해진 무결성 제약을 만족하는 유효한 상태를 유지해야 한다는 성질이다.
고립성(Isolation)은 동시에 수행되는 각 트랜잭션이 다른 트랜잭션의 중간 처리에 간섭하거나 영향을 받지 않는 것처럼 실행되어야 한다는 성질이다.
지속성(Durability)은 성공적으로 완료된 트랜잭션의 결과가 장애가 발생하더라도 영구적으로 보존되어야 한다는 성질이다.
핵심 대비는 원자성은 '전부 또는 전무', 일관성은 '규칙을 만족하는 상태 유지', 고립성은 '트랜잭션 간 간섭 방지', 지속성은 '완료 결과의 영구 보존'이다.

NoteBase

선지별 해설

① O — 고립성(Isolation)은 각 트랜잭션이 다른 트랜잭션과 독립적으로 수행되는 것처럼 보이게 하고, 실행 중인 트랜잭션들이 서로 간섭하지 않도록 하는 성질이다. 제시문의 설명과 정확히 일치한다.
② X — 일관성(Consistency)은 트랜잭션 간 독립성을 뜻하지 않는다. 틀린 표현은 '일관성'이며, 올바르게는 데이터베이스가 트랜잭션 수행 전후에 무결성 제약을 만족하는 일관된 상태를 유지하는 성질이다.
③ X — 원자성(Atomicity)은 다른 트랜잭션의 영향을 차단하는 성질이 아니다. 틀린 표현은 '원자성'이며, 올바르게는 트랜잭션의 연산이 모두 수행되거나 하나도 수행되지 않는 전부 또는 전무의 성질이다.
④ X — 지속성(Durability)은 트랜잭션 간 독립성을 뜻하지 않는다. 틀린 표현은 '지속성'이며, 올바르게는 커밋된 트랜잭션의 결과가 장애 후에도 영구적으로 보존되는 성질이다.
⑤ X — 투명성(Transparency)은 제시된 설명에 해당하지 않으며 ACID의 구성 요소도 아니다. 틀린 표현은 '투명성'이며, 올바르게는 여러 트랜잭션이 서로 영향을 미치지 않고 독립적으로 실행되는 것처럼 보이는 고립성(Isolation)이다.

오답 포인트

'독립적으로 수행된다'는 표현을 데이터베이스 상태가 일관되게 유지된다는 의미로 오해하기 쉽다. 그러나 이 문항의 결정적 단서는 '다른 트랜잭션에 영향을 미치지 않는다'이며, 이는 트랜잭션 사이의 간섭을 차단하는 고립성을 뜻한다.

다시 나오면 이렇게 판단

문장에서 무엇을 보호하는지 확인한다. 처리의 전부 또는 전무는 원자성, 무결성 규칙과 유효한 상태는 일관성, 동시 실행과 상호 간섭 방지는 고립성, 커밋 결과의 영구 보존은 지속성으로 판단한다.

암기 포인트

ACID는 원자성·일관성·고립성·지속성이며, '서로 간섭하지 않음'은 고립성이다.

NoteBase

2. 프로세스의 상태 변이에 대한 설명으로 옳지 않은 것은?

① 시간 할당량(time slice)을 사용하는 일반적 우선순위 기반 스케줄링에서 실행(running) 상태 프로세스의 시간 할당량이 모두 소진되었을 때, 우선순위가 높은 다른 준비 상태의 프로세스가 있다면 실행 중이던 프로세스는 커널(kernel)에 의해 스케줄링되기를 기다리는 준비(ready) 상태로 전이된다.

② 실행 상태의 프로세스가 동기식 입출력 요청을 하면, 일반적으로 해당 프로세스는 입출력이 완료될 때까지 CPU를 반납하고 대기(blocked 또는 waiting) 상태로 전이된다.

③ 대기 상태의 프로세스가 요청하였던 입출력이 완료되면, 해당 프로세스는 CPU 연산이 가능해지므로 바로 실행 상태로 전이된다.

④ 다중 처리기 시스템(multi-processor system)에서는 실행 상태의 프로세스가 여러 개 있을 수 있다.

⑤ 대기 상태의 프로세스들은 CPU 할당을 위한 스케줄링에서 제외된다.

정답 3번

출제 포인트

프로세스의 준비·실행·대기 상태가 어떤 사건에 의해 전이되는지와, CPU 스케줄러가 선택하는 대상이 무엇인지를 묻는 문제이다.

핵심 개념

준비(ready) 상태: CPU만 할당받으면 실행할 수 있어 스케줄링 대상이 되는 상태이다.
실행(running) 상태: CPU를 할당받아 명령을 수행하는 상태이다.
대기(blocked 또는 waiting) 상태: 입출력 완료와 같은 특정 사건을 기다리므로 즉시 CPU를 사용할 수 없고 스케줄링 대상에서 제외되는 상태이다.
대표 전이는 준비 → 실행(디스패치), 실행 → 준비(선점 또는 시간 할당량 소진), 실행 → 대기(입출력 요청), 대기 → 준비(기다리던 사건 완료)이다.

NoteBase

선지별 해설

① O — 시간 할당량이 소진되면 실행 중이던 프로세스는 CPU를 계속 점유할 수 없다. 더 높은 우선순위의 준비 상태 프로세스가 선택될 수 있으므로 기존 프로세스는 실행 상태에서 준비 상태로 전이되어 다시 스케줄링을 기다린다.
② O — 실행 중인 프로세스가 동기식 입출력을 요청하면 입출력이 끝날 때까지 연산을 계속할 수 없다. 따라서 CPU를 반납하고 실행 상태에서 대기 상태로 전이된다.
③ X — 틀린 표현은 '바로 실행 상태로 전이된다'이다. 입출력이 완료되면 프로세스는 대기 상태에서 준비 상태로 전이되어 CPU 스케줄링을 기다리고, 이후 디스패치되어야 실행 상태가 된다.
④ O — 다중 처리기 시스템은 여러 CPU 또는 처리 코어가 동시에 프로세스를 실행할 수 있으므로 실행 상태의 프로세스가 여러 개 존재할 수 있다.
⑤ O — 대기 상태의 프로세스는 입출력 완료 등 특정 사건을 기다리는 중이어서 CPU를 받아도 즉시 실행할 수 없다. 따라서 CPU 할당을 위한 준비 큐의 스케줄링 대상에서 제외된다.

오답 포인트

③은 입출력이 끝나면 실행 가능해진다는 사실을 곧바로 CPU를 할당받는다는 뜻으로 바꾼 함정이다. 실행 가능해진 프로세스는 먼저 준비 상태가 되며, 실제 실행 여부는 스케줄러의 선택과 디스패치에 달려 있다.

다시 나오면 이렇게 판단

CPU를 기다리면 준비 상태, 입출력이나 사건을 기다리면 대기 상태로 구분한다. 입출력 완료는 대기 → 준비이고, 준비 → 실행에는 반드시 스케줄링과 디스패치가 필요하다.

암기 포인트

입출력 요청은 실행 → 대기, 입출력 완료는 대기 → 준비, CPU 선택은 준비 → 실행이다.

NoteBase

3. OSI(Open System Interconnection) 7계층에 대한 설명으로 옳지 않은 것은?

① 7계층에서 물리적 전송 매체와 직접적으로 연관이 있는 계층은 물리 계층이다.
② 수신 측에서 패킷을 수신하게 되면, 상위 계층에서 하위 계층 순으로 처리된다.
③ 전송 계층(transport layer)의 대표적인 프로토콜로는 TCP, UDP 등이 있다.
④ 네트워크 계층에서는 라우팅 및 패킷 전송의 기능을 수행한다.
⑤ 응용 계층에서는 전자 사서함, 파일 전송 등의 서비스를 제공한다.

정답 2번

출제 포인트

OSI 7계층의 계층별 역할과 수신 측에서 데이터가 처리되는 방향을 묻는 문제이다.

핵심 개념

OSI 7계층은 네트워크 통신 기능을 7개 계층으로 나눈 참조 모델이다. 송신 측에서는 응용 계층에서 물리 계층 방향으로 캡슐화하고, 수신 측에서는 물리 계층에서 응용 계층 방향으로 역캡슐화한다.
계층별 핵심 역할은 물리 계층의 비트 전송, 네트워크 계층의 라우팅과 패킷 전달, 전송 계층의 종단 간 전송, 응용 계층의 사용자 응용 서비스 제공이다.

선지별 해설

① O — 물리 계층은 케이블, 광섬유, 전파 등 물리적 전송 매체를 통해 비트를 전달하므로 전송 매체와 직접 관련된다.
② X — 틀린 표현은 '상위 계층에서 하위 계층 순으로 처리된다'이다. 수신 측에서는 물리 계층부터 시작하여 하위 계층에서 상위 계층 순으로 역캡슐화된다.
③ O — TCP와 UDP는 전송 계층의 대표적인 프로토콜로, 종단 간 데이터 전송을 담당한다.
④ O — 네트워크 계층은 목적지까지의 경로를 선택하는 라우팅과 패킷 전달 기능을 수행한다.
⑤ O — 응용 계층은 전자우편과 파일 전송처럼 사용자 응용 프로그램이 이용하는 네트워크 서비스를 제공한다.

오답 포인트

②는 송신 측의 처리 방향과 수신 측의 처리 방향을 뒤바꾼 선지이다. 송신은 상위 계층에서 하위 계층으로, 수신은 하위 계층에서 상위 계층으로 진행된다.

NoteBase

다시 나오면 이렇게 판단

송신·캡슐화는 7계층에서 1계층 방향이고, 수신·역캡슐화는 1계층에서 7계층 방향이라는 기준으로 판단한다. 계층별 대표 기능도 물리 계층은 매체와 비트, 네트워크 계층은 라우팅, 전송 계층은 TCP·UDP, 응용 계층은 사용자 서비스로 연결한다.

암기 포인트

송신은 위에서 아래로, 수신은 아래에서 위로 처리된다.

NoteBase

4. 다음 정렬 알고리즘의 수행시간을 Big-O 표기법으로 나타냈을 때 최악의 경우의 수행시간이 같은 것들로만 나열된 것은?

선택 정렬(selection sort), 합병 정렬(merge sort), 삽입 정렬(insertion sort), 퀵 정렬(quick sort), 힙 정렬(heap sort)

① 합병 정렬, 퀵 정렬, 힙 정렬
② 힙 정렬, 선택 정렬, 퀵 정렬
③ 합병 정렬, 선택 정렬, 삽입 정렬
④ 합병 정렬, 힙 정렬, 삽입 정렬
⑤ 선택 정렬, 삽입 정렬, 퀵 정렬

정답 5번

출제 포인트

정렬 알고리즘별 최악의 경우 수행시간을 비교하여 같은 Big-O 시간복잡도를 갖는 알고리즘끼리 묶을 수 있는지를 묻는 문제이다.

핵심 개념

선택 정렬: 매 단계에서 최솟값을 찾아 배치하는 정렬이며, 최악의 경우 수행시간은 $O(n^2)$이다.
삽입 정렬: 원소를 앞쪽의 정렬된 부분에 알맞게 삽입하는 정렬이며, 최악의 경우 수행시간은 $O(n^2)$이다.
퀵 정렬: 피벗을 기준으로 분할하여 정렬하며, 분할이 한쪽으로 치우치는 최악의 경우 수행시간은 $O(n^2)$이다.
합병 정렬: 배열을 나눈 뒤 정렬하면서 합치는 정렬이며, 최악의 경우 수행시간은 $O(n\log n)$이다.
힙 정렬: 힙을 구성하고 최댓값 또는 최솟값을 반복해 꺼내는 정렬이며, 최악의 경우 수행시간은 $O(n\log n)$이다.
따라서 선택·삽입·퀵 정렬은 $O(n^2)$, 합병·힙 정렬은 $O(n\log n)$으로 구분된다.

NoteBase

선지별 해설

① X

합병 정렬과 힙 정렬은 $O(n\log n)$이지만 퀵 정렬은 최악의 경우 $O(n^2)$이다. 틀린 부분은 '합병 정렬, 퀵 정렬, 힙 정렬이 모두 같다'는 묶음이며, 올바르게는 합병 정렬과 힙 정렬만 최악의 경우 수행시간이 같다.

② X

선택 정렬과 퀵 정렬은 $O(n^2)$이지만 힙 정렬은 $O(n\log n)$이다. 틀린 부분은 힙 정렬을 선택·퀵 정렬과 같은 묶음에 넣은 것이며, 올바르게는 힙 정렬 대신 삽입 정렬을 넣어야 세 알고리즘이 모두 $O(n^2)$이 된다.

③ X

선택 정렬과 삽입 정렬은 $O(n^2)$이지만 합병 정렬은 $O(n\log n)$이다. 틀린 부분은 합병 정렬을 선택·삽입 정렬과 같은 묶음에 넣은 것이며, 올바르게는 합병 정렬 대신 퀵 정렬을 넣어야 한다.

④ X

합병 정렬과 힙 정렬은 $O(n\log n)$이지만 삽입 정렬은 $O(n^2)$이다. 틀린 부분은 삽입 정렬을 합병·힙 정렬과 같은 묶음에 넣은 것이며, 올바르게는 합병 정렬과 힙 정렬만 서로 같은 최악 수행시간을 갖는다.

⑤ O

선택 정렬, 삽입 정렬, 퀵 정렬은 모두 최악의 경우 수행시간이 $O(n^2)$이므로 같은 것들로만 나열되어 있다.

오답 포인트

퀵 정렬의 평균 수행시간 $O(n\log n)$을 최악 수행시간으로 착각하기 쉽다. 이 문항은 평균이 아니라 '최악의 경우'를 묻기 때문에 퀵 정렬을 $O(n^2)$ 그룹에 넣어야 한다.

다시 나오면 이렇게 판단

먼저 문제에서 평균·최선·최악 중 어느 경우를 묻는지 확인한다. 최악의 경우라면 선택·삽입·퀵 정렬을 $O(n^2)$, 합병·힙 정렬을 $O(n\log n)$으로 나눈 뒤 선지의 모든 알고리즘이 한 그룹에 속하는지 판단한다.

암기 포인트

최악 기준으로 '선·삽·퀵은 제곱, 합·힙은 엔로그엔'으로 기억한다.

NoteBase

5. 관계형 데이터베이스(relational database)에 대한 설명으로 옳은 것을 <보기>에서 모두 고르면?

<보 기>

ㄱ. 스키마 변환 시 정보의 무손실, 자료 중복의 감소, 관련된 구조 간의 통합의 원칙을 준수하여야 한다.

ㄴ. 관계대수(relational algebra)의 연산에서 피연산자는 모두 릴레이션이지만 연산결과는 릴레이션이 아니다.

ㄷ. 릴레이션에 연산을 수행 시 삽입이상(insertion anomaly), 삭제이상(deletion anomaly), 갱신이상(update anomaly)이 발생할 수 있다.

ㄹ. 튜플을 구성하는 속성 사이에 존재하는 종속관계를 고려하지 않고 하나의 릴레이션으로 표현하여 이상(anomaly)을 해결할 수 있다.

ㅁ. 릴레이션이 여러 속성을 표현할 때 이를 작게 분해(decomposition)하는 과정을 정규화(normalization)라고 한다.

ㅂ. 릴레이션들은 관계대수(relational algebra)로 조작이 가능하다.

① ㄱ, ㄴ, ㄷ
② ㄴ, ㄷ, ㄹ
③ ㄴ, ㄹ, ㅁ
④ ㄷ, ㄹ, ㅂ
⑤ ㄷ, ㅁ, ㅂ

정답 5번

출제 포인트

관계형 데이터베이스의 폐쇄성, 이상 현상, 함수 종속성과 정규화의 관계를 구별하여 옳은 설명의 조합을 찾는 문제이다.

핵심 개념

관계대수: 릴레이션을 입력으로 받아 연산 결과도 릴레이션으로 만드는 절차적 질의 연산 체계이다. 이를 폐쇄성이라고 한다.
이상 현상: 정규화가 충분하지 않은 릴레이션에서 데이터의 삽입·삭제·갱신 시 불필요한 제약이나 정보 손실, 불일치가 생기는 현상이다.
정규화: 속성 사이의 함수 종속성을 바탕으로 릴레이션을 더 작은 릴레이션으로 분해하여 중복과 이상 현상을 줄이는 과정이다.
보기 판단은 ㄱ O, ㄴ X, ㄷ O, ㄹ X, ㅁ O, ㅂ O이다.

NoteBase

선지별 해설

① X — ㄱ은 무손실 분해와 중복 감소 등을 고려한다는 취지에서 옳고 ㄷ도 옳지만, ㄴ이 틀리다. 관계대수는 피연산자뿐 아니라 연산 결과도 릴레이션이다. 따라서 옳은 조합은 ㄱ, ㄷ처럼 고쳐 판단해야 한다.
② X — ㄷ은 옳지만 ㄴ과 ㄹ이 틀리다. ㄴ은 '연산 결과도 릴레이션이다'로, ㄹ은 '속성 사이의 종속관계를 고려하여 릴레이션을 분해하면 이상 현상을 줄일 수 있다'로 고쳐야 한다.
③ X — ㅁ은 옳지만 ㄴ과 ㄹ이 틀리다. 관계대수의 결과는 릴레이션이며, 종속관계를 무시한 하나의 릴레이션은 이상 현상을 해결하는 것이 아니라 유발할 수 있다. 따라서 ㄴ, ㄹ, ㅁ은 옳은 조합이 아니다.
④ X — ㄷ과 ㅂ은 옳지만 ㄹ이 틀리다. ㄹ의 '종속관계를 고려하지 않고 하나의 릴레이션으로 표현하여 이상을 해결한다'는 부분은 '종속관계를 고려해 적절히 분해하여 이상을 줄인다'로 고쳐야 한다.
⑤ O — ㄷ은 비정규화된 릴레이션의 삽입·삭제·갱신 과정에서 이상 현상이 발생할 수 있다는 설명이고, ㅁ은 릴레이션을 작은 릴레이션으로 분해하는 정규화의 설명이며, ㅂ은 릴레이션을 관계대수로 조작할 수 있다는 설명이므로 모두 옳다.

오답 포인트

ㄴ의 함정은 관계대수의 폐쇄성을 놓치게 하는 데 있다. 관계대수는 입력과 출력이 모두 릴레이션이다. ㄹ은 정규화의 원리를 정반대로 서술했다. 함수 종속성을 무시한 통합이 아니라 종속성을 고려한 분해가 이상 현상을 줄인다.

다시 나오면 이렇게 판단

관계대수 문장은 '입력도 릴레이션, 출력도 릴레이션'인지 확인한다. 이상 현상과 정규화 문장은 '함수 종속성 고려 → 릴레이션 분해 → 중복과 이상 감소'의 흐름에 맞는지 확인한다. 이 기준으로 ㄴ과 ㄹ을 제외하면 ㄷ, ㅁ, ㅂ이 남는다.

암기 포인트

관계대수는 릴레이션에서 릴레이션으로, 정규화는 종속성을 보고 분해하여 이상을 줄이는 과정이다.

NoteBase

6. 후위표기법(postfix notation)으로 된 다음 식의 전위표기법(prefix notation)으로 옳은 것은?

ABC+D/-AE+BF*/+

① + - A/+BCD/+AE*BF
② - +A/BC+D/+AE*BF
③ +-A/+BCD/+*AEBF
④ +A-/+BCD/+AE*BF
⑤ - +/A+BCD/+*AEBF

정답 1번

출제 포인트

후위표기식을 왼쪽부터 읽으며 피연산자를 쌓고, 연산자를 만나면 오른쪽 피연산자와 왼쪽 피연산자를 차례로 꺼내 하나의 전위표기식으로 결합하는 과정을 묻는 문제이다.

올바른 순서

1. 후위식 ABC+D/-를 처리하면 -A/+BCD가 된다. 이는 $A - ((B + C)/D)$를 뜻한다.
2. 이어지는 AE+BF*/를 처리하면 /+AE*BF가 된다. 이는 $(A + E)/(B * F)$를 뜻한다.
3. 마지막 연산자 +가 두 부분식을 결합하므로 전체 전위표기식은 +-A/+BCD/+AE*BF가 된다.

왜 이 순서인가: 후위표기식에서 연산자는 바로 앞에서 완성된 두 피연산자 또는 부분식을 결합하므로, 각 부분식을 먼저 만든 뒤 마지막 +를 전체 식의 맨 앞에 놓아야 한다.

선지별 해설

① O — +-A/+BCD/+AE*BF는 왼쪽 부분식 -A/+BCD와 오른쪽 부분식 /+AE*BF를 마지막 +로 결합한 정확한 전위표기식이다.

② X — 맨 앞 연산자가 -이므로 전체 식을 뺄셈으로 묶는다. 그러나 주어진 후위식의 마지막 연산자는 +이므로 전체 연산도 덧셈이어야 한다. 틀린 표현 -+A/BC+D/+AE*BF는 +-A/+BCD/+AE*BF로 고쳐야 한다.

③ X — 오른쪽 부분의 /+*AEBF는 A와 E를 먼저 더하고 B와 F를 먼저 곱하는 구조가 아니다. 틀린 표현 /+*AEBF는 /+AE*BF로 고쳐야 하므로 전체 식은 +-A/+BCD/+AE*BF가 된다.

④ X — 맨 앞의 +A-는 A를 최종 덧셈의 왼쪽 피연산자로 분리하여, A가 먼저 뺄셈의 왼쪽 피연산자가 되어야 하는 구조를 깨뜨린다. 틀린 표현 +A-/+BCD/+AE*BF는 +-A/+BCD/+AE*BF로 고쳐야 한다.

⑤ X — 맨 앞 연산자가 -이므로 마지막 +가 전체 식을 결합한다는 후위표기식의 구조와 맞지 않고, 왼쪽 부분식의 연산 순서도 달라진다. 틀린 표현 -+/A+BCD/+*AEBF는 +-A/+BCD/+AE*BF로 고쳐야 한다.

NoteBase

오답 포인트

후위식의 연산자 순서를 그대로 앞쪽으로 옮기는 것이 함정이다. 특히 마지막 +는 가장 늦게 계산되는 최상위 연산자이므로 전위표기식에서는 가장 먼저 와야 하며, AE+와 BF*도 각각 하나의 부분식으로 완성한 뒤 /로 결합해야 한다.

다시 나오면 이렇게 판단

후위표기식을 왼쪽부터 스캔하면서 피연산자는 스택에 넣고, 연산자를 만나면 오른쪽 항과 왼쪽 항을 차례로 꺼내 '연산자+왼쪽 항+오른쪽 항'의 문자열로 다시 넣는다. 마지막에 스택에 남은 하나가 전위표기식이다.

암기 포인트

후위식의 마지막 연산자는 전위식의 첫 연산자이다. 연산자를 만나면 오른쪽 항을 먼저 꺼내고 왼쪽 항을 나중에 꺼낸다.

NoteBase

7. 다음 카르노 맵에 해당하는 논리식 F로 옳은 것은?

BC \ A	0	1
00	0	1
01	0	0
11	1	0
10	1	0

① F = AB + AB′C
② F = A′B′ + A′B′C′
③ F = AB′ + A′BC
④ F = A′B + AB′C′
⑤ F = AB + BC

정답 4번

출제 포인트

카르노 맵에서 값이 1인 인접 셀을 묶어 최소 논리식을 구하는 문제이다. 인접 묶음 안에서 값이 변하는 변수는 소거하고, 값이 고정된 변수만 곱항에 남긴다.

조건 정리

값이 1인 셀은 $A=0,\ BC=11$, $A=0,\ BC=10$, $A=1,\ BC=00$이다.
행은 그레이 코드 순서인 00, 01, 11, 10으로 배치되므로 $BC=11$과 $BC=10$은 서로 인접한다.

사용할 공식

카르노 맵의 인접한 1들을 2의 거듭제곱 개수로 묶고, 묶음 안에서 변하지 않는 변수만 남겨 곱항을 만든 뒤 각 곱항을 논리합한다.

$F = A'B + AB'C'$

계산 과정

1. $A=0$ 열의 $BC=11$, $BC=10$ 두 셀을 묶는다. 이 묶음에서는 $A=0$과 $B=1$이 고정되고 C만 변하므로 $A'B$가 된다.
2. $A=1,\ BC=00$인 1은 다른 1과 묶을 수 없다. 따라서 세 변수를 모두 남겨 $AB'C'$가 된다.
3. 두 곱항을 논리합하면 $F = A'B + AB'C'$이다.

NoteBase

선지 판단

① X — AB는 $A=1, B=1$인 셀들을 1로 만들지만 해당 셀들은 맵에서 0이다. 또한 $AB'C$는 $A=1, B=0, C=1$인 0의 셀을 포함한다. 틀린 식 $AB + AB'C \rightarrow$ 올바르게는 $A'B + AB'C'$이다.
② X — $A'B' + A'B'C'$에서 두 번째 항은 첫 번째 항에 흡수되어 $A'B'$가 되며, 이는 $A=0, B=0$인 0의 셀들을 1로 만든다. 틀린 식 $A'B' + A'B'C' \rightarrow$ 올바르게는 $A'B + AB'C'$이다.
③ X — AB'는 $A=1, B=0, C=1$인 0의 셀까지 포함하고, $A'BC$는 $A=0, B=1, C=0$인 1의 셀을 빠뜨린다. 틀린 식 $AB' + A'BC \rightarrow$ 올바르게는 $A'B + AB'C'$이다.
④ O — $A'B$는 왼쪽 열의 인접한 두 1을 나타내고, $AB'C'$는 오른쪽 열의 고립된 1을 나타낸다. 따라서 맵의 모든 1만 정확히 포함한다.
⑤ X — AB와 BC는 모두 맵에서 0인 $A=1, B=1$ 영역을 포함하며, 실제 1인 $A=1, B=0, C=0$도 나타내지 못한다. 틀린 식 $AB + BC \rightarrow$ 올바르게는 $A'B + AB'C'$이다.

오답 포인트

행의 11과 10이 인접한다는 점을 놓치면 $A'B$를 $A'BC$처럼 불필요하게 길게 쓸 수 있다. 또한 고립된 셀 $A=1, BC=00$에서는 B와 C가 모두 0이므로 $AB'C'$에서 두 변수에 모두 보수가 붙는다.

다시 나오면 이렇게 판단

먼저 1의 위치를 확인하고 가능한 가장 큰 인접 묶음을 만든다. 각 묶음에서 값이 변하는 변수는 지우고 고정된 변수만 남긴 뒤, 완성된 식이 1인 셀을 모두 포함하면서 0인 셀은 포함하지 않는지 검산한다.

암기 포인트

카르노 맵은 '변하는 변수는 소거, 고정된 변수는 유지'가 핵심이다. 0으로 고정된 변수에는 프라임을 붙인다.

NoteBase

8. 가상 메모리 시스템에서 메모리 부족 시의 페이지 교체 기법(page replacement algorithm)들에 대한 설명으로 옳지 않은 것은?

① LRU(Least Recently Used) 기법은 메모리에 적재된 페이지 중 가장 오랫동안 참조되지 않았던 페이지를 교체하는 기법이다.

② LRU 기법은 실제 그 구현 오버헤드가 커서, 일반적으로 오버헤드를 줄인 여러 유형의 LRU 근사 알고리즘(LRU approximation algorithm)들이 사용되는 것이 보통이다.

③ LRU 기법은 물리적 페이지의 개수를 확장했음에도 페이지 폴트가 늘어나는 경우가 발생할 수도 있는데, 이를 Belady's anomaly라 한다.

④ 이론적으로는 최적의 페이지 교체 기법은 메모리에 적재된 페이지들 중에서 앞으로 가장 오랫동안 참조되지 않을 페이지를 교체하는 것이다.

⑤ FIFO(First In First Out) 기법은 메모리에 적재된 페이지들 중 가장 먼저 메모리에 적재된 페이지를 교체하는 방법이다.

정답 3번

출제 포인트

페이지 교체 기법인 LRU, 최적(Optimal), FIFO의 교체 기준과 Belady의 이상 현상 발생 여부를 구별하는 문제이다.

핵심 개념

LRU는 과거에 가장 오랫동안 참조되지 않은 페이지를 교체하며, 최근 사용 이력을 유지해야 하므로 정확한 구현의 오버헤드가 크다.
최적 페이지 교체는 미래에 가장 오랫동안 참조되지 않을 페이지를 교체하며, 이론적으로 최소 페이지 폴트 수를 제공한다.
FIFO는 메모리에 가장 먼저 들어온 페이지를 교체한다.
Belady의 이상 현상은 페이지 프레임을 늘렸는데도 페이지 폴트가 증가하는 현상이다. FIFO에서는 발생할 수 있지만 스택 알고리즘인 LRU와 최적 기법에서는 발생하지 않는다.

NoteBase

선지별 해설

① O — LRU는 현재 메모리에 적재된 페이지 중 가장 오랫동안 참조되지 않은 페이지를 교체한다.
② O — 정확한 LRU를 구현하려면 각 페이지의 최근 참조 순서나 시점을 계속 관리해야 하므로 오버헤드가 크다. 따라서 참조 비트 등을 이용한 LRU 근사 알고리즘이 흔히 사용된다.
③ X — 틀린 표현은 'LRU 기법에서 프레임 수를 늘려도 페이지 폴트가 증가할 수 있다'는 부분이다. LRU는 프레임 수가 늘어날 때 기존 프레임의 페이지 집합이 포함되는 스택 알고리즘이므로 Belady의 이상 현상이 발생하지 않는다. 올바르게는 'FIFO 기법에서는 프레임 수를 늘려도 페이지 폴트가 증가하는 Belady의 이상 현상이 발생할 수 있다'이다.
④ O — 최적 페이지 교체 기법은 현재 적재된 페이지 중 앞으로 가장 늦게 참조되거나 다시 참조되지 않을 페이지를 교체한다.
⑤ O — FIFO는 메모리에 가장 먼저 적재된 페이지, 즉 적재된 지 가장 오래된 페이지를 교체한다.

오답 포인트

③은 Belady의 이상 현상이라는 용어 자체는 맞지만 그 발생 대상을 LRU로 바꾼 선지이다. 프레임 증가에도 페이지 폴트가 늘어날 수 있는 대표 기법은 FIFO이며, LRU에는 해당하지 않는다.

다시 나오면 이렇게 판단

교체 기준을 시간 방향으로 구분한다. FIFO는 가장 먼저 들어온 페이지, LRU는 과거에 가장 오래 사용되지 않은 페이지, 최적 기법은 미래에 가장 오래 사용되지 않을 페이지를 고른다. Belady의 이상 현상은 FIFO와 연결하고 LRU·최적 기법에서는 발생하지 않는다고 판단한다.

암기 포인트

FIFO는 입장 순서, LRU는 과거 사용 기록, Optimal은 미래 참조를 본다. Belady의 이상 현상은 FIFO에서는 가능하지만 LRU에서는 불가능하다.

NoteBase

9. 요즘 집안에서 스마트폰 및 컴퓨터의 인터넷 접속을 위하여 공유기의 사용이 일반화되어 있다. 공유기에 접속하는 컴퓨터 디바이스에는 사설 IP가 주로 할당되는데 이때 IP주소의 자동 할당을 위해 사용되는 프로토콜로 옳은 것은?

① ARP
② DHCP
③ TCP
④ UDP
⑤ ICMP

정답 2번

출제 포인트

공유기 내부의 컴퓨터나 스마트폰에 사설 IP 주소와 네트워크 설정을 자동으로 할당하는 프로토콜을 구별하는 문제이다.

핵심 개념

DHCP(Dynamic Host Configuration Protocol)는 네트워크에 접속한 장치에 IP 주소를 일정 기간 임대하고, 서브넷 마스크·기본 게이트웨이·DNS 서버 등의 설정도 함께 제공하는 프로토콜이다.
ARP는 IP 주소에 대응하는 MAC 주소를 찾고, TCP와 UDP는 데이터를 전달하는 전송 계층 프로토콜이며, ICMP는 네트워크 오류 보고와 상태 확인에 사용된다.

선지별 해설

① X — ARP는 같은 네트워크에서 특정 IPv4 주소에 대응하는 MAC 주소를 알아내는 프로토콜이다. 틀린 표현은 'IP주소의 자동 할당에 사용된다'는 취지이며, 올바르게는 'IP 주소를 MAC 주소로 대응시키는 데 사용된다'이다.
② O — DHCP는 공유기에 접속한 장치에 사용 가능한 사설 IP 주소를 자동으로 임대하고 필요한 네트워크 설정을 제공한다.
③ X — TCP는 연결을 설정한 뒤 순서와 신뢰성을 보장하며 데이터를 전달하는 전송 계층 프로토콜이다. 틀린 표현은 'IP주소의 자동 할당에 사용된다'는 취지이며, 올바르게는 '신뢰성 있는 연결형 데이터 전송에 사용된다'이다.
④ X — UDP는 연결 설정 없이 데이터그램을 전달하는 전송 계층 프로토콜이다. DHCP가 메시지를 주고받을 때 UDP를 이용하지만, IP 주소를 자동으로 할당하는 프로토콜 자체는 DHCP이다. 틀린 표현은 'UDP가 IP 주소를 자동 할당한다'는 취지이며, 올바르게는 'UDP는 DHCP 메시지의 전송에 이용된다'이다.
⑤ X — ICMP는 IP 통신 중 오류를 알리거나 네트워크 상태를 진단하는 데 사용되는 제어 메시지 프로토콜이다. 틀린 표현은 'IP주소의 자동 할당에 사용된다'는 취지이며, 올바르게는 '오류 보고와 통신 상태 확인에 사용된다'이다.

NoteBase

오답 포인트

④ UDP는 DHCP가 사용하는 전송 수단이므로 정답처럼 보일 수 있다. 그러나 'IP 주소를 자동으로 할당하는 기능'의 주체는 DHCP이고, UDP는 DHCP 메시지를 운반할 뿐이다.

다시 나오면 이렇게 판단

문제에서 'IP 주소 자동 할당·임대·네트워크 설정 제공'이 나오면 DHCP로 판단한다. 'IP와 MAC의 대응'은 ARP, '신뢰성 있는 연결형 전송'은 TCP, '비연결형 전송'은 UDP, '오류 보고·진단'은 ICMP로 구별한다.

암기 포인트

자동 IP는 DHCP, IP-MAC 대응은 ARP, 전송은 TCP·UDP, 오류·진단은 ICMP이다.

NoteBase

10. 다음의 C 프로그램 실행 결과로 출력되는 a, b, c 값으로 옳은 것은?

```
#include <stdio.h>
int foo(int a, int *b)
{
        int c;
        *b = a + 1;
        c = a - 1;
        return c;
}
void main()
{
    int a = 5;
    int b = 3;
    int c = 0;
    b = foo(a, &c);
    c = foo(b, &a);
    printf("a=%d b=%d c=%d\n", a, b, c);
}
```

① 5, 4, 3
② 4, 3, 2
③ 6, 3, 4
④ 4, 3, 5
⑤ 5, 3, 0

정답 1번

출제 포인트

이 문제는 함수의 값 전달, 포인터를 통한 원본 변수 변경, 반환값 대입이 결합된 C 프로그램의 실행 흐름을 묻는다.

NoteBase

올바른 순서

1. main에서 처음에는 a=5, b=3, c=0이다.
2. b=foo(a, &c)를 실행하면 foo의 매개변수 a에는 5가 복사되고, 포인터 b는 main의 c를 가리킨다. 따라서 *b=a+1에 의해 main의 c가 6이 되고, 함수의 지역변수 c에는 a-1인 4가 저장되어 반환된다. 이 반환값이 main의 b에 대입되므로 a=5, b=4, c=6이 된다.
3. c=foo(b, &a)를 실행하면 foo의 매개변수 a에는 현재 main의 b 값인 4가 복사되고, 포인터 b는 main의 a를 가리킨다. 따라서 *b=a+1에 의해 main의 a가 5가 되고, 함수의 지역변수 c에는 a-1인 3이 저장되어 반환된다. 이 반환값이 main의 c에 대입되므로 최종적으로 a=5, b=4, c=3이 된다.

왜 이 순서인가: 각 함수 호출 안의 포인터 대입이 먼저 원본 변수를 바꾸고, 함수가 반환된 뒤 그 반환값이 대입문의 왼쪽 변수에 저장되기 때문이다.

선지별 해설

① O — 두 호출을 순서대로 추적한 최종값은 a=5, b=4, c=3이다.

② X — b와 c를 각각 3, 2로 계산한 부분이 틀렸다. 올바르게는 첫 번째 호출의 반환값 4가 b에 저장되고 두 번째 호출의 반환값 3이 c에 저장되므로 a=5, b=4, c=3이다.

③ X — a=6, b=3, c=4라는 결과는 첫 번째 호출에서 *b=a+1로 바뀌는 대상과 반환값의 저장 대상을 혼동한 것이다. 올바르게는 &c가 전달되므로 첫 호출에서 c가 6이 되고 반환값 4가 b에 저장되며, 두 번째 호출까지 마치면 a=5, b=4, c=3이다.

④ X — a=4, b=3, c=5라는 결과는 두 번째 호출의 값 매개변수와 포인터가 가리키는 원본 변수를 혼동한 것이다. 올바르게는 두 번째 호출에서 함수의 매개변수 a는 4이고 포인터 b는 main의 a를 가리키므로 main의 a는 5, 반환값은 3이 되어 최종값은 a=5, b=4, c=3이다.

⑤ X — a=5, b=3, c=0은 함수 호출 전의 초기값이므로 실행 결과가 아니다. 올바르게는 두 함수 호출의 포인터 변경과 반환값 대입을 모두 반영하여 a=5, b=4, c=3이다.

오답 포인트

핵심 함정은 foo 안의 매개변수 이름 a, b와 main의 변수 a, b를 같은 변수로 보는 데 있다. foo의 정수형 a는 값이 복사된 지역 매개변수이고, 포인터형 b만 호출할 때 전달된 주소의 원본 변수를 바꾼다. 또한 함수 안의 지역변수 c와 main의 c도 서로 다른 변수이다.

다시 나오면 이렇게 판단

함수 호출마다 먼저 값 매개변수에 복사되는 값과 포인터가 가리키는 원본 변수를 적고, 함수 내부 문장을 위에서 아래로 실행한 뒤, 마지막에 반환값을 호출문의 왼쪽 변수에 대입하여 상태를 갱신한다.

암기 포인트

값 매개변수는 복사본, 포인터 역참조는 원본 변경, return 값은 호출문의 왼쪽 변수에 저장된다.

NoteBase

11. 데이터베이스에서 사용하는 뷰(view)에 대한 설명으로 옳은 것을 <보기>에서 모두 고르면?

<보 기>

ㄱ. 뷰의 정의는 변경할 수 없다.
ㄴ. 뷰는 삽입, 갱신, 삭제 연산에 제약이 있다.
ㄷ. 뷰 위에 다른 뷰를 정의할 수 없다.
ㄹ. 뷰가 정의된 테이블이 삭제되더라도 뷰는 남는다.
ㅁ. 뷰는 물리적으로 구현되는 테이블이다.

① ㄱ, ㄴ
② ㄱ, ㅁ
③ ㄴ, ㄷ
④ ㄷ, ㄹ
⑤ ㄹ, ㅁ

정답 1번

출제 포인트

뷰의 정의·갱신 제약·중첩 정의·기본 테이블 삭제 시의 존속 여부·물리적 저장 여부를 구분하는 문제이다.

핵심 개념

뷰는 하나 이상의 기본 테이블이나 다른 뷰에서 유도되는 가상 테이블이다. 일반적인 뷰는 질의 정의를 저장하며, 실제 데이터는 기본 테이블에 저장된다.

ㄱ X: 뷰의 정의는 변경할 수 있다. 기존 뷰를 삭제한 뒤 다시 만들거나 DBMS가 지원하는 뷰 변경·대체 기능을 사용할 수 있다.

ㄴ O: 뷰는 구성 방식에 따라 삽입·갱신·삭제가 제한된다. 조인, 집계, 그룹화 등이 포함된 뷰는 일반적으로 갱신이 제한된다.

ㄷ X: 뷰를 이용하여 다른 뷰를 정의할 수 있다.

ㄹ O: 뷰가 참조하는 기본 테이블을 삭제해도 뷰 정의 자체는 남을 수 있다. 다만 참조 대상이 없어져 그 뷰는 정상적으로 사용할 수 없게 된다.

ㅁ X: 일반적인 뷰는 물리적으로 데이터를 저장한 테이블이 아니라 논리적으로 구성된 가상 테이블이다.

NoteBase

선지별 해설

① O — ㄴ과 ㄹ은 모두 옳다. 뷰에는 갱신 연산의 제약이 있으며, 기본 테이블이 삭제되어도 뷰 정의는 남을 수 있다.
② X — ㄱ의 '정의는 변경할 수 없다'가 틀리고 ㅁ의 '물리적으로 구현되는 테이블'도 틀리다. 올바른 조합은 ㄴ, ㄹ이다.
③ X — ㄷ의 '뷰 위에 다른 뷰를 정의할 수 없다'가 틀리다. 뷰를 기반으로 다른 뷰를 정의할 수 있으므로 올바른 조합은 ㄴ, ㄹ이다.
④ X — ㄷ은 틀린 설명이다. 뷰 위에 다른 뷰를 정의할 수 있으므로 올바른 조합은 ㄴ, ㄹ이다.
⑤ X — ㅁ의 '물리적으로 구현되는 테이블'이 틀리다. 일반적인 뷰는 논리적인 가상 테이블이므로 올바른 조합은 ㄴ, ㄹ이다.

오답 포인트

ㄹ의 '뷰는 남는다'를 '계속 정상적으로 조회할 수 있다'는 뜻으로 오해하기 쉽다. 기본 테이블이 삭제되면 뷰 정의는 남을 수 있지만 참조 대상이 없으므로 사용할 수 없다. 또한 일반 뷰와 조회 결과를 실제로 저장하는 물리적 뷰를 혼동하면 ㅁ을 옳다고 판단하기 쉽다.

다시 나오면 이렇게 판단

뷰 문제에서는 먼저 '정의가 저장되는가, 데이터가 저장되는가'를 구분한다. 일반 뷰는 질의 정의를 저장하는 가상 테이블이고, 다른 뷰를 기반으로 정의할 수 있으며, 복잡한 뷰일수록 삽입·갱신·삭제가 제한된다고 판단한다.

암기 포인트

뷰는 가상 테이블, 중첩 정의 가능, 갱신은 제한될 수 있음, 기본 테이블 삭제 시 정의는 남아도 사용은 불가능하다.

NoteBase

12. 프로세스(process)나 스레드(thread)들이 공유 자원에 하나 이상의 수정(write 또는 modify) 연산을 포함하는 동시 접근을 할 때 그 접근 부분들을 임계 구역(critical section)이라 한다. 이를 보호하기 위한 병행 프로세스 동기화 기법으로 옳은 것은?

① 인터럽트(interrupt)
② 선점 스케줄링(preemptive scheduling)
③ 문맥교환(context switching)
④ 상호배제(mutual exclusion)
⑤ 교착상태(deadlock)

정답 4번

출제 포인트

이 문제는 공유 자원을 수정하는 임계 구역을 보호하는 동기화 원리가 무엇인지 묻는 문제이다.

핵심 개념

임계 구역은 여러 프로세스나 스레드가 공유 자원에 접근하는 코드 중 동시에 실행되면 데이터 불일치가 발생할 수 있는 부분이다.
상호 배제는 한 시점에 하나의 프로세스나 스레드만 임계 구역에 들어가게 하여 경쟁 상태를 방지하는 원리이다. 뮤텍스, 세마포어 등의 동기화 도구가 이를 구현한다.

선지별 해설

① X — 인터럽트는 하드웨어나 소프트웨어 사건을 CPU에 알리고 현재 실행 흐름을 잠시 중단시키는 메커니즘이다. '인터럽트가 임계 구역을 보호한다'는 표현이 틀렸으며, 올바르게는 상호 배제 기법이 임계 구역의 동시 진입을 막는다.
② X — 선점 스케줄링은 운영체제가 실행 중인 프로세스로부터 CPU를 회수해 다른 프로세스에 할당할 수 있는 스케줄링 방식이다. '선점 스케줄링이 공유 자원 접근을 보호한다'는 표현이 틀렸으며, 올바르게는 뮤텍스나 세마포어 같은 동기화 도구가 공유 자원 접근을 조정한다.
③ X — 문맥교환은 실행 중인 프로세스나 스레드의 상태를 저장하고 다음 실행 대상의 상태를 복원하는 작업이다. '문맥교환이 임계 구역의 동시 접근을 차단한다'는 표현이 틀렸으며, 올바르게는 상호 배제가 동시에 하나만 임계 구역에 진입하도록 보장한다.
④ O — 상호 배제는 공유 자원을 다루는 임계 구역에 한 시점에 하나의 프로세스나 스레드만 진입하도록 제한하므로 임계 구역을 보호하는 동기화 원리이다.
⑤ X — 교착상태는 둘 이상의 프로세스나 스레드가 서로 상대방이 가진 자원을 기다리며 영원히 진행하지 못하는 상태이다. '교착상태가 임계 구역을 보호하는 기법'이라는 표현이 틀렸으며, 올바르게는 교착상태는 동기화 과정에서 발생할 수 있어 예방하거나 회피해야 하는 문제이다.

NoteBase

오답 포인트

인터럽트, 선점 스케줄링, 문맥교환은 모두 실행 흐름의 전환과 관련되어 있어 동기화 기법처럼 보일 수 있다. 그러나 임계 구역 보호의 핵심은 실행을 단순히 바꾸는 것이 아니라 동시에 둘 이상이 진입하지 못하게 하는 상호 배제이다. 교착상태는 보호 기법이 아니라 잘못된 자원 대기로 발생하는 문제이다.

다시 나오면 이렇게 판단

문제에서 '공유 자원', '임계 구역', '동시 접근 방지'를 제시하면 한 번에 하나만 접근하게 하는 상호 배제를 먼저 판단한다. CPU 실행 순서를 바꾸는 기능과 공유 자원 접근을 제한하는 기능을 구분한다.

암기 포인트

임계 구역 보호는 상호 배제, CPU 전환은 스케줄링·문맥교환, 멈춰 기다리는 문제는 교착상태이다.

NoteBase

13. 배열 int array[10][200]를 행우선순서(row major order)로 저장하는 경우의 원소 array[7][12]의 시작주소는 몇 번지인가?
(단, 배열 array의 시작주소는 10840h로, int의 크기는 4바이트로 가정한다. 배열 첨자는 0부터 시작하며 숫자에 붙은 h는 16진수 표기를 의미한다.)

① 10804h
② 11E50h
③ 16488h
④ 108BFh
⑤ 10A3Ch

정답 2번

출제 포인트

행우선순서로 저장된 2차원 배열에서 특정 원소까지의 선형 인덱스를 구하고, 원소 크기를 곱해 바이트 오프셋과 시작주소를 계산하는 문제이다.

조건 정리

주어진 값: 배열은 int array[10][200], 시작주소는 10840h, int의 크기는 4*byte*이다.
구해야 할 값: array[7][12]의 시작주소이다.
주의할 조건: 행과 열의 첨자는 0부터 시작하며, 행우선순서에서는 한 행의 200개 원소가 연속으로 저장된다.

사용할 공식

행우선순서에서 array[i][j]의 주소는 다음과 같다.
주소=시작주소+((i×열의 수)+j)×원소 크기
행우선순서는 앞선 i개 행의 모든 원소 뒤에 현재 행의 j번째 원소가 놓이므로 이 공식을 사용한다.

계산 과정

1. array[7][12] 앞에는 $7 \times 200 + 12 = 1{,}412$개의 원소가 있다.
2. 바이트 오프셋은 $1{,}412 \times 4 = 5{,}648$이고, 이를 16진수로 바꾸면 1610h이다.
3. 시작주소에 오프셋을 더하면 $10840h + 1610h = 11E50h$이다.

NoteBase

선지 판단

① X — 10804h는 시작주소 10840h보다도 작고, 계산된 오프셋 1610h를 더한 주소가 아니다. 올바르게는 11E50h이다.
② O — 10840h에 array[7][12]의 바이트 오프셋 1610h를 더한 결과가 11E50h이다.
③ X — 16488h는 $10840h + 1610h$의 합과 일치하지 않는다. 올바르게는 11E50h이다.
④ X — 108BFh는 시작주소와의 차가 7Fh에 불과하여 필요한 바이트 오프셋 1610h를 반영하지 못한다. 올바르게는 11E50h이다.
⑤ X — 10A3Ch는 시작주소와의 차가 1FCh로, 필요한 바이트 오프셋 1610h와 다르다. 올바르게는 11E50h이다.

오답 포인트

첨자가 0부터 시작한다는 이유로 7과 12에서 다시 1을 빼면 안 된다. 또한 7행을 건너뛸 때는 행의 개수인 10이 아니라 한 행의 원소 수인 200을 곱해야 하며, 원소 개수로 구한 오프셋에 int의 크기인 4*byte*를 곱한 뒤 시작주소에 더해야 한다.

다시 나오면 이렇게 판단

행우선 2차원 배열의 주소 문제는 '행 첨자 × 열의 수 + 열 첨자'로 선형 인덱스를 구한 뒤, 원소 크기를 곱하고 시작주소에 더하는 순서로 판단한다.

암기 포인트

행우선 주소 = 시작주소 + (행 첨자 × 열의 수 + 열 첨자) × 원소 크기이다.

NoteBase

14. 다음은 마트에서 판매 기록을 저장한 트랜잭션(transaction) 데이터이다. 규칙 “기저귀” -> “맥주”의 신뢰도(confidence) 값으로 옳은 것은?

식별자	품목
1	맥주, 땅콩, 기저귀
2	맥주, 커피, 기저귀
3	맥주, 기저귀, 계란
4	땅콩, 계란, 우유
5	땅콩, 커피, 기저귀, 우유

① 0.33
② 0.5
③ 0.75
④ 0.8
⑤ 1.00

정답 3번

출제 포인트

이 문제는 연관 규칙 ‘기저귀 → 맥주’의 신뢰도를 구하는 문제이다. 신뢰도는 기저귀를 산 거래 중 맥주도 함께 산 거래의 비율이다.

조건 정리

기저귀가 포함된 거래는 식별자 1, 2, 3, 5의 4건이다.
기저귀와 맥주가 함께 포함된 거래는 식별자 1, 2, 3의 3건이다.
구해야 할 값은 규칙 ‘기저귀 → 맥주’의 신뢰도이다.

사용할 공식

$\text{confidence}(A \rightarrow B) = \frac{\text{count}(A \cap B)}{\text{count}(A)}$
신뢰도는 선행항 A가 발생한 거래를 조건으로 후행항 B도 발생한 비율이므로, 분모에는 기저귀가 포함된 거래 수를 넣는다.

계산 과정

1. 기저귀가 포함된 거래 수는 4건이다.
2. 기저귀와 맥주가 함께 포함된 거래 수는 3건이다.
3. 따라서 신뢰도는 $\frac{3}{4} = 0.75$이다.

NoteBase

선지 판단

① X — 0.33은 계산 결과와 다르다. '기저귀 → 맥주'의 신뢰도는 0.75이다.
② X — 0.5는 계산 결과와 다르다. 기저귀 거래 4건 중 맥주 동시 거래가 3건이므로 올바른 값은 0.75이다.
③ O — 3/4 = 0.75이므로 계산 결과와 일치한다.
④ X — 0.8은 전체 5건 중 기저귀가 포함된 4건의 비율이다. 이는 신뢰도가 아니며, 올바른 신뢰도는 0.75이다.
⑤ X — 1.00이 되려면 기저귀가 포함된 모든 거래에 맥주도 있어야 한다. 식별자 5에는 맥주가 없으므로 올바른 값은 0.75이다.

오답 포인트

가장 큰 함정은 분모를 전체 거래 5건으로 잡는 것이다. 신뢰도는 '기저귀를 산 경우'라는 조건 아래 맥주도 산 비율이므로 분모는 기저귀 거래 4건이다. 또한 식별자 5에는 기저귀는 있지만 맥주가 없어 신뢰도가 1.00이 되지 않는다.

다시 나오면 이렇게 판단

규칙 $A \rightarrow B$의 신뢰도는 먼저 A가 포함된 거래만 모은 뒤, 그중 A와 B가 함께 포함된 거래의 비율을 계산한다. 규칙의 방향이 바뀌면 분모도 바뀐다.

암기 포인트

신뢰도는 '선행항 거래 중 동시 발생 거래의 비율'이다. 즉, $A \rightarrow B$에서는 A가 분모이다.

NoteBase

15. 사용자가 WWW(World Wide Web)을 사용하여 임의의 사이트에 접속하는 과정의 설명으로 옳지 않은 것은?

① 사용자의 웹브라우저에서 DNS 서버로 DNS 질의 메시지를 보낸다.
② DNS 간의 질의는 재귀형(recursive) 또는 반복형(iterative)으로 연계된다.
③ 사용자의 웹브라우저는 DNS 서버로부터 사이트의 웹 서버 MAC 주소를 획득한다.
④ HTTP 요청 메시지를 보내기 전에 TCP 3-방향 핸드셰이크를 수행한다.
⑤ 웹 서버가 보낸 HTTP 응답 메시지의 바디로부터 HTML을 추출하여 웹 페이지를 보여준다.

정답 3번

출제 포인트

웹 사이트 접속 과정에서 DNS, TCP, HTTP가 각각 담당하는 기능과 IP 주소·MAC 주소의 차이를 묻는 문제이다.

핵심 개념

DNS는 도메인 이름을 웹 서버의 IP 주소로 변환하는 이름 해석 시스템이다. TCP는 HTTP 데이터를 안정적으로 주고받기 위한 연결을 설정하며, 일반적인 TCP 기반 HTTP에서는 통신 전에 3-방향 핸드셰이크를 수행한다. HTTP는 웹브라우저와 웹 서버 사이에서 요청과 응답을 교환하는 응용 계층 프로토콜이다. MAC 주소는 같은 데이터 링크 구간에서 프레임을 전달할 때 사용하는 주소이므로, DNS가 원격 웹 서버의 MAC 주소를 알려 주는 것은 아니다.

NoteBase

선지별 해설

① O — 사용자가 도메인 이름으로 접속하면 웹브라우저 또는 운영체제의 이름 해석 기능이 DNS 서버에 질의하여 해당 도메인의 IP 주소를 확인한다.

② O — DNS 질의 방식에는 질의를 받은 서버가 최종 결과까지 찾아 주는 재귀적 질의와, 다음에 질의할 서버 정보를 돌려주는 반복적 질의가 있다.

③ X — 틀린 표현은 '웹 서버 MAC 주소를 획득한다'이다. DNS로 얻는 것은 사이트 웹 서버의 IP 주소이다. MAC 주소가 필요할 때에는 같은 네트워크 구간의 다음 홉 장치 주소를 ARP 또는 NDP 등으로 확인하므로, 올바르게는 '웹브라우저는 DNS 서버로부터 웹 서버의 IP 주소를 획득한다'이다.

④ O — 일반적인 TCP 기반 HTTP 통신에서는 HTTP 요청을 보내기 전에 클라이언트와 웹 서버가 TCP 3-방향 핸드셰이크로 연결을 설정한다.

⑤ O — 웹브라우저는 웹 서버가 보낸 HTTP 응답의 바디에 담긴 HTML을 해석하고, 관련 자원을 처리하여 웹 페이지를 화면에 표시한다.

오답 포인트

③은 네트워크 통신에 IP 주소와 MAC 주소가 모두 쓰인다는 점을 이용해 둘의 역할을 바꿔 놓은 선지이다. DNS는 도메인 이름과 IP 주소를 대응시키며, 라우터를 거치는 원격 통신에서 목적지 웹 서버의 MAC 주소를 DNS로 받아 사용하는 것은 아니다.

다시 나오면 이렇게 판단

웹 접속 과정을 'DNS로 IP 주소 확인 → TCP 연결 설정 → HTTP 요청·응답 → 브라우저의 HTML 해석'으로 구분한다. DNS의 결과를 MAC 주소라고 설명하면 틀린 것으로 판단한다.

암기 포인트

DNS는 도메인에서 IP로, MAC 주소 확인은 같은 링크 구간에서 이루어진다.

NoteBase

16. 1과 1000 사이의 정수로 구성된 이진탐색트리(binary search tree)에서 숫자 573을 탐색하는 경우 다음 중 비교경로로 옳지 않은 것은?

① 2, 173, 241, 856, 301, 489, 710, 516, 573
② 7, 816, 68, 714, 121, 561, 278, 395, 573
③ 981, 825, 693, 38, 137, 608, 224, 461, 573
④ 926, 139, 884, 278, 734, 319, 662, 481, 573
⑤ 14, 970, 831, 765, 111, 249, 318, 473, 573

정답 2번

출제 포인트

이 문제는 이진 탐색 트리에서 목표값 573을 찾을 때 각 비교 결과가 다음 노드의 허용 범위를 어떻게 좁히는지 묻는 문제이다.

올바른 순서

1. 현재 노드와 573을 비교한다.
2. 현재 노드가 573보다 크면 왼쪽 서브트리로 이동하고 그 노드를 새로운 상한으로 정한다.
3. 현재 노드가 573보다 작으면 오른쪽 서브트리로 이동하고 그 노드를 새로운 하한으로 정한다.
4. 이후의 모든 비교값이 누적된 하한과 상한 사이에 있는지 확인하며 573에 도달한다.

왜 이 순서: 앞선 비교에서 정해진 하한과 상한이 다음에 방문할 수 있는 서브트리의 범위를 결정하기 때문이다.

선지별 해설

① O — 2, 173, 241은 차례로 하한을 높이고, 856은 상한을 정한다. 이후 301, 489, 710, 516이 모두 당시의 하한과 상한 안에 있으므로 573에 도달할 수 있는 경로이다.
② X — 561은 573보다 작으므로 다음 탐색은 561의 오른쪽 서브트리에서 이루어져야 한다. 따라서 다음 값은 561보다 커야 하는데 278이 제시되어 이진 탐색 규칙에 어긋난다. 틀린 부분은 '561 다음에 278을 비교한다'는 순서이다. 올바르게는 561 다음에 561보다 크고 당시 상한인 714보다 작은 값이 와야 하며, 예를 들어 7, 816, 68, 714, 121, 561, 600, 580, 573은 가능한 경로이다.
③ O — 981, 825, 693은 상한을 낮추고, 38과 137은 하한을 높인다. 이어서 608, 224, 461도 각각 갱신된 범위 안에 있으므로 573까지 탐색할 수 있다.
④ O — 926을 상한으로 정한 뒤 139를 하한으로 정하고, 이후 884, 278, 734, 319, 662, 481이 번갈아 상한과 하한을 좁힌다. 모든 값이 당시 허용 범위 안에 있으므로 가능한 경로이다.
⑤ O — 14를 하한으로 정하고 970, 831, 765로 상한을 낮춘 뒤 111, 249, 318, 473으로 하한을 높인다. 범위 위반 없이 573에 도달하므로 가능한 경로이다.

NoteBase

오답 포인트

각 숫자가 단순히 573보다 큰지 작은지만 보면 ②의 오류를 놓치기 쉽다. 핵심 함정은 561에서 오른쪽으로 이동한 뒤에는 이전 하한 121이 아니라 561이 새 하한이 된다는 점이다. 따라서 278과 395는 573보다 작더라도 561보다 작아서 더 이상 방문할 수 없다.

다시 나오면 이렇게 판단

처음에는 하한과 상한을 각각 1과 1000으로 두고, 목표값보다 작은 노드를 만나면 하한을 그 값으로, 큰 노드를 만나면 상한을 그 값으로 갱신한다. 다음 값이 매 단계의 열린 구간 '하한보다 크고 상한보다 작음'을 벗어나는 순간 그 경로는 불가능하다고 판단한다.

암기 포인트

작은 노드는 하한을 올리고, 큰 노드는 상한을 내린다. 다음 비교값은 항상 그 사이에 있어야 한다.

NoteBase

17. 유니버설 게이트 집합(universal gate set)은 그 구성 원소만으로 어떤 형태의 디지털 시스템도 구현할 수 있는 기능적으로 완전한 게이트들의 집합이다. 다음 중 유니버설 게이트 집합으로 옳지 않은 것은?

① { NAND }
② { OR, DECODER }
③ { MULTIPLEXER }
④ { OR, NOT }
⑤ { XOR }

정답 5번

출제 포인트

유니버설 게이트 집합이 임의의 불 함수를 구현할 수 있는 기능적으로 완전한 구성인지 판단하는 문제이다.

핵심 개념

유니버설 게이트 집합은 그 구성 원소만으로 NOT, AND, OR을 비롯한 모든 불 함수를 구현할 수 있는 집합이다. NAND는 단독으로 기능적으로 완전하고, 디코더와 OR의 조합은 최소항의 합으로 함수를 구현하며, 멀티플렉서는 입력과 상수값을 적절히 연결해 임의의 불 함수를 구성할 수 있다. OR와 NOT의 조합도 드모르간 법칙으로 AND를 만들 수 있다. 반면 XOR만으로 만든 함수는 입력과 상수의 배타적 논리합 형태에 한정되어 AND와 같은 함수를 구현할 수 없다.

선지별 해설

① O — {NAND}는 NAND만으로 NOT, AND, OR을 모두 만들 수 있으므로 유니버설 게이트 집합이다.
② O — {OR, DECODER}에서 디코더는 입력 조합에 대응하는 최소항을 만들고 OR는 필요한 최소항을 합하므로 임의의 불 함수를 구현할 수 있다.
③ O — {MULTIPLEXER}는 선택선과 데이터 입력에 변수 또는 상수 0, 1을 연결하여 NOT, AND, OR 및 임의의 불 함수를 구현할 수 있으므로 유니버설 게이트 집합이다.
④ O — {OR, NOT}는 드모르간 법칙에 따라 AND를 구성할 수 있으므로 기능적으로 완전하다.
⑤ X — 틀린 표현은 '{XOR}가 유니버설 게이트 집합이다'라는 판단이다. 올바르게는 XOR만으로는 AND와 같은 비선형 불 함수를 구현할 수 없으므로 {XOR}는 유니버설 게이트 집합이 아니다.

오답 포인트

③의 멀티플렉서와 ②의 디코더가 기본 게이트 이름이 아니라는 이유만으로 제외하기 쉽다. 유니버설 여부는 장치의 이름이 아니라 그 구성만으로 모든 불 함수를 합성할 수 있는지로 판단한다. 또한 XOR가 여러 논리 연산에 쓰인다는 사실과 기능적으로 완전하다는 사실을 혼동하면 ⑤를 놓치게 된다.

NoteBase

다시 나오면 이렇게 판단

각 집합만으로 NOT과 AND 또는 OR를 모두 만들 수 있는지 확인한다. NAND 단독, NOR 단독, OR와 NOT, AND와 NOT은 기능적으로 완전하며, 디코더와 OR 및 멀티플렉서도 임의의 불 함수를 구현할 수 있다. XOR만 있는 집합은 AND를 만들 수 없으므로 기능적으로 완전하지 않다.

암기 포인트

NAND·NOR는 단독 완전, MUX도 완전, XOR 단독은 불완전이다.

NoteBase

18. 다음 C 프로그램의 실행 결과로 옳은 것은?

```
#include <stdio.h>

int recursion(int n)
{
        if (n < 5) return 1;
        else    if (n % 5 == 1) return n + recursion(n - 1);
        else    recursion(n - 1);
}

int main()
{
        int n    = recursion(16);
        printf("%d",    n);

        return    0;
}
```

① 34
② 33
③ 31
④ 29
⑤ 28

정답 1번

출제 포인트

재귀 호출이 종료 조건에 도달할 때까지 인수를 1씩 줄이고, n 5 = 1인 단계의 값만 더하여 출력값을 구하는 문제이다.

조건 정리

주어진 값: recursion(16)을 호출한다.
종료 조건: $n < 5$이면 1을 반환한다.
덧셈 조건: n 5=1이면 출제 의도상 n을 재귀 호출의 결과에 더한다.
주의할 조건: 5로 나눈 나머지가 1이 아닌 단계는 인수만 1 줄여 다음 호출로 넘어간다.

NoteBase

사용할 공식

출제 의도에 따른 점화식은 다음과 같다.

$R(n) = 1 \quad (n < 5)$
R(n)=n+R(n-1)(n≥ 5, n 5=1)
R(n)=R(n-1)(n≥ 5, n 5≠ 1)

이 점화식을 쓰는 이유는 재귀 호출 과정에서 조건을 만족하는 단계만 현재의 n을 누적하기 때문이다.

계산 과정

1. 16에서 16 5=1이므로 16을 더한다.
2. 이후 15부터 인수를 1씩 줄이다가 11과 6에서 다시 나머지가 1이므로 각각 11과 6을 더한다.
3. 1에서는 먼저 $n < 5$ 조건이 성립하므로 1을 반환하고 재귀가 종료된다.
4. 따라서 출력값은 16 + 11 + 6 + 1 = 34이다.

선지 판단

① O — 계산 결과가 34이므로 프로그램의 의도된 출력값과 일치한다.
② X — 33은 종료 조건에서 반환되는 1을 누락한 값이다. ‘33’이 아니라 올바른 출력값은 34이다.
③ X — 31은 누적되는 16, 11, 6과 종료값 1의 합과 일치하지 않는다. ‘31’이 아니라 올바른 출력값은 34이다.
④ X — 29는 조건을 만족하는 단계의 누적합과 일치하지 않는다. ‘29’가 아니라 올바른 출력값은 34이다.
⑤ X — 28은 16, 11, 1만 더하고 6을 빠뜨린 경우에 나오는 값이다. ‘28’이 아니라 올바른 출력값은 34이다.

오답 포인트

종료 조건을 먼저 검사하므로 $n = 1$에서는 나머지 조건을 다시 적용하지 않고 1만 반환한다. 또한 15, 14처럼 나머지가 1이 아닌 수는 더하지 않으며, 16·11·6만 누적한다. 다만 제시된 C 코드의 마지막 else에는 return이 빠져 있다. 표준 C에서는 값을 반환해야 하는 함수가 값 없이 끝나면 반환값이 정의되지 않으므로, 34는 해당 문장을 return recursion(n - 1);로 보는 출제 의도에 따른 답이다.

다시 나오면 이렇게 판단

재귀 함수에서는 종료 조건을 먼저 확정한 뒤, 각 호출 단계에서 실제로 누적되는 값만 적는다. 이 문제에서는 16부터 5까지 중 5로 나눈 나머지가 1인 16, 11, 6을 고르고 마지막 반환값 1을 더한다.

암기 포인트

재귀 계산은 ‘종료값 1 + 조건을 만족하는 16·11·6’으로 정리되며 합은 34이다.

NoteBase

19. 인터넷에서 주로 사용하는 TCP와 UDP에 대한 설명으로 옳지 않은 것은?

① TCP에는 네트워크 상 트래픽이 폭주하는 현상을 방지하는 혼잡 제어(congestion control) 기능이 포함되어 있다.
② UDP는 비신뢰적이고 비연결형인 서비스를 제공한다.
③ TCP는 패킷 손실이 발생했을 때, 패킷 재전송 기능을 수행한다.
④ UDP 패킷 헤더에는 출발지 포트 번호, 목적지 포트 번호, 길이, 체크섬 등의 정보를 담고 있다.
⑤ TCP는 신뢰적인 데이터 전달은 보장하지만, 수신 프로세스에게 데이터가 올바른 순서로 전달되는 것은 보장하지 못한다.

정답 5번

출제 포인트

이 문제는 TCP와 UDP의 연결 방식, 신뢰성, 순서 보장, 혼잡 제어 및 헤더 구조의 차이를 묻는 문제이다.

핵심 개념

TCP는 연결형 전송 프로토콜로, 확인 응답·재전송·순서 제어·흐름 제어·혼잡 제어를 이용하여 신뢰성 있고 순서가 보장된 바이트 스트림을 제공한다.
UDP는 비연결형 전송 프로토콜로, 재전송이나 순서 제어 없이 데이터그램을 전달하므로 신뢰성과 도착 순서를 보장하지 않는다.
TCP는 신뢰성과 순서를 중시하고, UDP는 단순한 헤더와 적은 처리 부담을 중시한다.

선지별 해설

① O — TCP에는 네트워크 혼잡을 완화하기 위한 혼잡 제어 기능이 포함되어 있다.
② O — UDP는 연결 설정을 하지 않는 비연결형이며, 확인 응답과 재전송을 수행하지 않아 비신뢰적 서비스를 제공한다.
③ O — TCP는 세그먼트 손실이 감지되면 확인 응답과 타이머 등을 바탕으로 해당 데이터를 재전송한다.
④ O — UDP 헤더는 출발지 포트 번호, 목적지 포트 번호, 길이, 체크섬의 네 필드로 구성된다.
⑤ X — 틀린 표현은 '데이터가 올바른 순서로 전달되는 것은 보장하지 못한다'이다. TCP는 순서 번호와 확인 응답을 이용하므로 수신 프로세스에 데이터를 올바른 순서로 제공한다. 올바르게는 'TCP는 신뢰성 있는 전달과 올바른 순서의 전달을 모두 보장한다'이다.

오답 포인트

⑤는 TCP가 신뢰적이라는 사실은 맞게 제시한 뒤 순서 보장만 부정하여 혼동을 유도한다. 순서 보장은 TCP 신뢰성의 핵심 기능이며, 순서와 신뢰성을 보장하지 않는 쪽은 UDP이다.

NoteBase

다시 나오면 이렇게 판단

‘연결·확인 응답·재전송·순서 보장·흐름 제어·혼잡 제어’가 나오면 TCP로 판단하고, ‘비연결·비신뢰·순서 미보장·단순한 헤더’가 나오면 UDP로 판단한다.

암기 포인트

TCP는 연결·신뢰·순서·재전송·혼잡 제어, UDP는 비연결·비신뢰·순서 미보장이다.

NoteBase

20. 다음 중 회선 교환 방식(circuit switching)의 특성으로 옳지 않은 것은?

① 데이터를 주고 받기 전에 종단 간 연결(end to end connnection) 과정이 필요하다.
② 연결이 수립되면, 해당 회선을 독점적으로 사용한다.
③ 회선 교환 방식의 대표적인 예로는 전화망이 있다.
④ 패킷 교환 방식(packet switching)에 비해 더 많은 동시 접속자를 수용할 수 있다.
⑤ 패킷 교환 방식에 비해 안정적인 전송속도를 보장할 수 있다.

정답 4번

출제 포인트

회선 교환 방식이 통신 전에 전용 경로를 설정하고 그 경로의 자원을 독점하는 방식임을 알고, 패킷 교환 방식과 비교할 수 있는지를 묻는 문제이다.

핵심 개념

회선 교환: 송신자와 수신자 사이에 종단 간 전용 통신 경로를 먼저 설정한 뒤 데이터를 연속적으로 전송하는 방식이다. 연결된 동안 회선 자원을 독점하므로 전송률과 지연이 비교적 안정적이지만, 통신하지 않는 순간에도 자원이 예약되어 이용 효율이 낮다.
패킷 교환: 데이터를 패킷으로 나누어 여러 사용자가 통신망의 경로와 자원을 공유하는 방식이다. 회선 교환보다 자원 이용 효율이 높아 더 많은 동시 접속자를 수용할 수 있지만, 혼잡 상태에 따라 지연과 전송률이 달라질 수 있다.

선지별 해설

① O — 회선 교환은 데이터를 주고받기 전에 송신자와 수신자 사이의 종단 간 연결을 먼저 설정해야 한다.
② O — 연결이 수립되면 통신이 끝날 때까지 설정된 회선 자원을 해당 연결이 독점적으로 사용한다.
③ O — 전통적인 전화망은 통화 전에 전용 회선을 설정하여 사용하는 회선 교환 방식의 대표적인 예이다.
④ X — 틀린 표현은 '패킷 교환 방식에 비해 더 많은 동시 접속자를 수용할 수 있다'이다. 회선 교환은 각 연결에 회선 자원을 전용으로 할당하므로 패킷 교환보다 동시 접속자 수용 능력이 작다. 올바르게는 '패킷 교환 방식이 회선 교환 방식보다 더 많은 동시 접속자를 수용할 수 있다'이다.
⑤ O — 연결마다 전용 경로와 대역폭이 확보되므로 패킷 교환에 비해 안정적인 전송속도를 보장하기 쉽다.

오답 포인트

④는 전용 회선이 있다는 점을 동시 접속자 수용 능력이 크다는 뜻으로 바꾼 함정이다. 전용 자원은 한 연결의 품질을 안정시키지만, 여러 사용자가 자원을 공유하지 못하게 하므로 전체 동시 수용 능력은 오히려 작아진다.

NoteBase

다시 나오면 이렇게 판단

‘연결 설정·전용 경로·자원 독점·안정적 전송률’은 회선 교환, ‘패킷 분할·자원 공유·높은 이용 효율·가변 지연’은 패킷 교환으로 구분한다. 동시 접속자 수는 자원을 공유하는 패킷 교환 쪽이 더 많다고 판단한다.

암기 포인트

회선 교환은 전용이라 안정적이지만 수용 인원은 적고, 패킷 교환은 공유라 효율적이고 수용 인원은 많다.

NoteBase