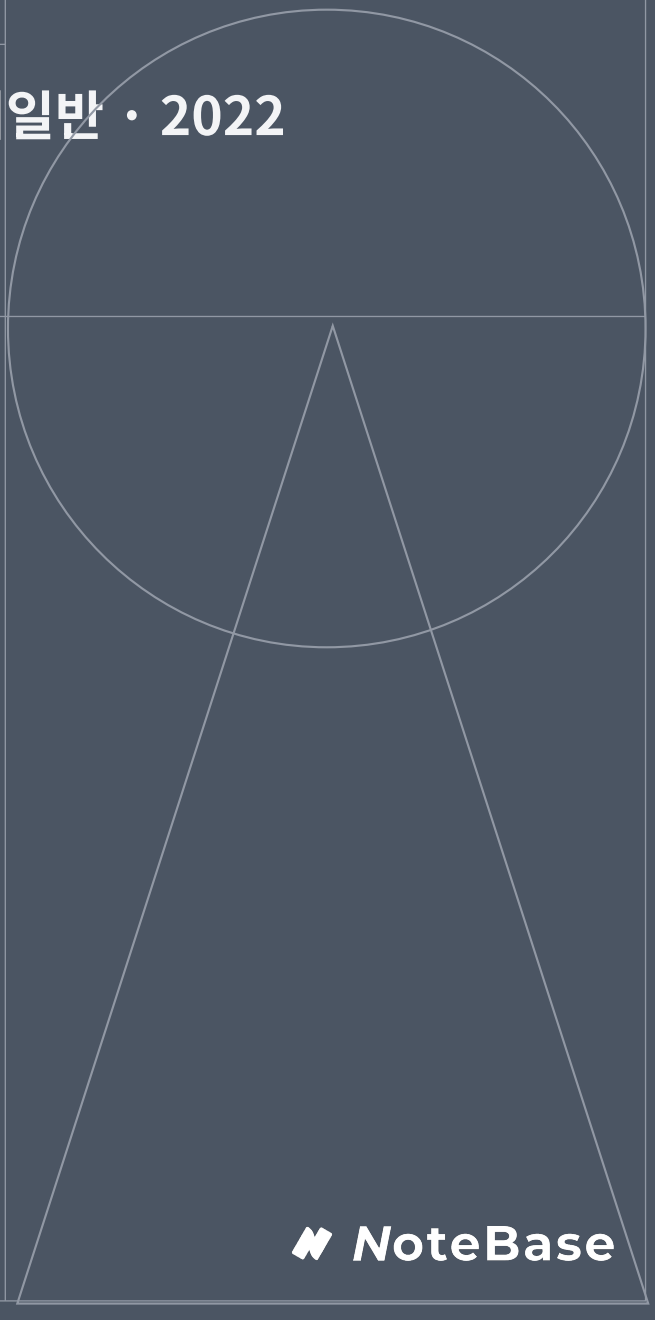
기출OX 해설집

공무원/국회직/9급 · 컴퓨터일반 · 2022



NoteBase

기출OX 해설집

공무원/국회직/9급 · 컴퓨터일반 · 2022

공무원/국회직/9급 · 컴퓨터일반 · 2022

1. OSI 7계층과 해당 프로토콜을 연결한 것으로 옳지 않은 것은?

① 응용 계층 - HTTP, SMTP
② 표현 계층 - MPEG, SSL
③ 전송 계층 - TCP, TELNET
④ 네트워크 계층 - IP, ICMP
⑤ 데이터링크 계층 - Ethernet, FDDI

정답 3번

출제 포인트

OSI 7계층의 기능과 대표 프로토콜을 올바르게 연결할 수 있는지를 묻는 문제이다.

핵심 개념

응용 계층은 사용자의 네트워크 서비스가 동작하는 계층으로 HTTP, SMTP, TELNET 등이 해당한다. 표현 계층은 데이터의 표현 형식 변환, 압축, 암호화를 담당하며 MPEG, SSL 등이 해당한다. 전송 계층은 종단 간 전송과 신뢰성·흐름 제어를 담당하며 TCP, UDP 등이 해당한다. 네트워크 계층은 논리 주소 지정과 경로 선택을 담당하며 IP, ICMP 등이 해당한다. 데이터링크 계층은 인접 노드 간 프레임 전송과 매체 접근 제어를 담당하며 Ethernet, FDDI 등이 해당한다.

선지별 해설

① O — HTTP와 SMTP는 사용자가 웹과 전자우편 서비스를 이용하도록 하는 응용 계층 프로토콜이다.
② O — MPEG는 데이터 압축·부호화와 관련되고 SSL은 암호화와 관련되므로 전통적인 OSI 분류에서 표현 계층에 해당한다.
③ X — TCP는 전송 계층이지만 TELNET은 원격 접속 서비스를 제공하는 응용 계층 프로토콜이다. 틀린 표현은 '전송 계층 - TCP, TELNET'이며, 올바르게는 '전송 계층 - TCP, UDP'이다.
④ O — IP는 논리 주소 지정과 패킷 전달을, ICMP는 네트워크 오류 및 제어 메시지 전달을 담당하므로 둘 다 네트워크 계층에 해당한다.
⑤ O — Ethernet과 FDDI는 프레임 전송과 매체 접근을 규정하는 데이터링크 계층 기술이다.

NoteBase

오답 포인트

③은 TCP와 TELNET이 모두 T로 시작하고 TCP/IP 환경에서 함께 사용된다는 점을 이용한 함정이다. TELNET은 TCP를 이용하지만, 자신은 응용 계층에 속한다. 사용하는 하위 계층 프로토콜과 해당 프로토콜 자체의 계층을 구분해야 한다.

다시 나오면 이렇게 판단

프로토콜의 용도를 먼저 본다. 웹·메일·원격 접속처럼 사용자 서비스를 제공하면 응용 계층, 종단 간 데이터 전달을 담당하면 전송 계층, 주소 지정과 경로 선택을 담당하면 네트워크 계층으로 판단한다.

암기 포인트

응용은 HTTP·SMTP·TELNET, 표현은 MPEG·SSL, 전송은 TCP·UDP, 네트워크는 IP·ICMP, 데이터링크는 Ethernet·FDDI이다.

NoteBase

2. 유닉스 운영체제에서 다음과 같은 파일의 접근권한을 세자리 숫자로 표기한 것으로 옳은 것은?

> 파일의 소유자는 읽고 쓰고 실행할 수 있지만 파일의 소유자를 제외한 사용자는 실행만 할 수 있다.

① 711
② 722
③ 744
④ 644
⑤ 611

정답 1번

출제 포인트

유닉스 파일 접근권한을 소유자·그룹·기타 사용자 순서의 세 자리 8진수로 변환할 수 있는지를 묻는 문제이다.

핵심 개념

유닉스 접근권한의 숫자는 읽기 r=4, 쓰기 w=2, 실행 x=1의 합으로 나타낸다. 세 자리 숫자는 차례로 소유자, 그룹, 기타 사용자의 권한이다. 따라서 읽기·쓰기·실행은 4+2+1=7이고, 실행만 가능한 권한은 1이다.

선지별 해설

① O — 711은 소유자에게 7(rwx), 그룹과 기타 사용자에게 각각 1(--x)을 부여한다. 제시된 조건과 정확히 일치한다.
② X — 722에서 뒤의 두 자리 2는 그룹과 기타 사용자에게 쓰기 권한만 부여하므로 '실행만 가능'이라는 조건에 어긋난다. 올바른 표기는 711이다.
③ X — 744에서 뒤의 두 자리 4는 그룹과 기타 사용자에게 읽기 권한만 부여하므로 '실행만 가능'이라는 조건에 어긋난다. 올바른 표기는 711이다.
④ X — 644의 첫 자리 6은 소유자에게 읽기와 쓰기만 허용하고 실행 권한을 주지 않으며, 뒤의 4도 비소유자에게 읽기 권한을 준다. 올바른 표기는 711이다.
⑤ X — 611의 첫 자리 6은 소유자에게 읽기와 쓰기만 허용하여 실행 권한이 빠진다. 소유자에게 읽기·쓰기·실행을 모두 주려면 첫 자리가 7이어야 하므로 올바른 표기는 711이다.

오답 포인트

각 숫자를 권한의 개수로 해석하거나 세 자리의 대상을 뒤섞기 쉽다. 특히 비소유자에게 실행만 허용한다는 표현은 읽기나 쓰기도 가능하다는 뜻이 아니므로 둘째와 셋째 자리는 각각 1이어야 한다.

NoteBase

다시 나오면 이렇게 판단

먼저 권한을 소유자·그룹·기타 사용자로 나눈 뒤 각 대상별로 r=4, w=2, x=1을 더한다. 이 문제에서는 소유자의 rwx가 7이고 그룹과 기타 사용자의 --x가 각각 1이므로 711이 된다.

암기 포인트

r=4, w=2, x=1이며 세 자리 순서는 소유자-그룹-기타이다.

NoteBase

3. 페이지 교체 알고리즘으로 LRU와 FIFO 알고리즘을 사용하고 페이지 참조의 순서는 다음과 같다. 이 경우, 할당된 프레임의 수가 3개일 때 각각의 알고리즘에서 발생하는 페이지 부재 횟수로 옳은 것은? (단, 초기에는 기억장치가 모두 비어 있다고 가정한다)

페이지 참조의 순서: 3, 1, 2, 4, 1, 4, 3, 2, 1, 2, 3

	LRU	FIFO
①	5번	6번
②	7번	6번
③	8번	6번
④	7번	7번
⑤	8번	7번

정답 4번

출제 포인트

프레임이 3개이고 처음에는 모두 비어 있을 때, 같은 페이지 참조열에 LRU와 FIFO를 각각 적용하여 페이지 부재 횟수를 계산하는 문제이다.

조건 정리

페이지 참조열: 3, 1, 2, 4, 1, 4, 3, 2, 1, 2, 3
할당 프레임 수: 3개
초기 상태: 모든 프레임이 비어 있으므로 처음 등장하는 페이지는 페이지 부재가 발생한다.
구해야 할 값: LRU와 FIFO에서 발생하는 페이지 부재의 총횟수이다.

사용할 공식

페이지 부재 횟수는 참조열을 차례로 처리하면서 해당 페이지가 프레임에 없을 때 기록한 F의 개수이다.
LRU는 페이지 부재가 발생하고 프레임이 가득 찼을 때 가장 오랫동안 사용되지 않은 페이지를 교체한다.
FIFO는 페이지 부재가 발생하고 프레임이 가득 찼을 때 가장 먼저 들어온 페이지를 교체한다. 페이지가 적중해도 FIFO의 적재 순서는 바뀌지 않는다.

NoteBase

계산 과정

LRU 추적: 3(F) → 1(F) → 2(F) → 4(F, 3 교체) → 1(H) → 4(H) → 3(F, 2 교체) → 2(F, 1 교체) → 1(F, 4 교체) → 2(H) → 3(H)이다. 따라서 LRU의 페이지 부재는 7회이다.
FIFO 추적: 3(F) → 1(F) → 2(F) → 4(F, 3 교체) → 1(H) → 4(H) → 3(F, 1 교체) → 2(H) → 1(F, 2 교체) → 2(F, 4 교체) → 3(H)이다. 따라서 FIFO의 페이지 부재는 7회이다.
여기서 F는 페이지 부재, H는 페이지 적중을 뜻한다.

선지 판단

① X — ‘LRU 5번, FIFO 6번’은 두 계산 결과와 모두 다르다. 올바르게는 LRU 7번, FIFO 7번이다.
② X — LRU 7번은 맞지만 ‘FIFO 6번’이 틀리다. FIFO에서는 3, 1, 2, 4, 3, 1, 2를 적재할 때 총 7회의 부재가 발생하므로, 올바르게는 LRU 7번, FIFO 7번이다.
③ X — ‘LRU 8번, FIFO 6번’은 두 계산 결과와 모두 다르다. LRU의 마지막 두 참조 2와 3은 이미 프레임에 있어 적중하며, FIFO 부재도 7회이므로 올바르게는 LRU 7번, FIFO 7번이다.
④ O — LRU와 FIFO의 페이지 부재 횟수는 각각 7회이므로 제시된 ‘LRU 7번, FIFO 7번’과 일치한다.
⑤ X — FIFO 7번은 맞지만 ‘LRU 8번’이 틀리다. LRU에서 페이지 부재가 발생하는 참조는 3, 1, 2, 4, 3, 2, 1의 7개이므로, 올바르게는 LRU 7번, FIFO 7번이다.

오답 포인트

LRU에서는 적중이 발생할 때도 최근 사용 시점이 갱신되지만, FIFO에서는 적중이 발생해도 들어온 순서가 갱신되지 않는다. 특히 참조열 중간의 1과 4의 적중을 FIFO 순서 변경으로 잘못 처리하면 FIFO를 6회로 세기 쉽고, LRU의 마지막 2와 3을 다시 부재로 세면 LRU를 8회로 잘못 계산하기 쉽다.

다시 나오면 이렇게 판단

참조열을 왼쪽부터 한 번씩 처리하며 프레임에 있으면 H, 없으면 F를 기록한다. 프레임이 가득 찬 뒤의 F에서만 교체 대상을 정하고, LRU는 최근 사용 시점, FIFO는 최초 적재 순서를 별도로 관리한 다음 F의 개수를 센다.

암기 포인트

LRU는 ‘마지막 사용 시점’, FIFO는 ‘처음 들어온 순서’를 기준으로 교체하며, 적중 시 갱신되는 것은 LRU의 사용 시점뿐이다.

NoteBase

4. 다음 중위표기식을 후위표기식으로 바꾼 것으로 옳은 것은?

$$a+(b-c)\times d\div(e-f)$$

① $ab+c-def-\div\times$
② $abc-d\times ef-\div+$
③ $abc-d\times ef\div-+$
④ $a+b-c\times d\div e-f$
⑤ $+a\div\times-bcd-ef$

정답 2번

출제 포인트

중위표기식에서 괄호와 연산자 우선순위를 반영해 각 연산자를 두 피연산자 뒤로 보내는 후위표기식 변환을 묻는 문제이다.

올바른 순서

1. 괄호 안의 *b* − *c*를 *bc* −로 바꾼다.
2. 그 결과와 *d*의 곱을 *bc* − *d* ×로 바꾼다.
3. 괄호 안의 *e* − *f*를 *ef* −로 바꾼다.
4. 두 결과의 나눗셈을 *bc* − *d* × *ef* − ÷로 배열한다.
5. 마지막으로 *a*와의 덧셈 연산자 +를 맨 뒤에 놓아 *abc* − *d* × *ef* − ÷ +를 얻는다.

왜 이 순서인가: 후위표기식에서는 각 연산자가 해당 피연산자들이 모두 나온 뒤에 위치하므로, 안쪽 괄호의 연산을 먼저 완성한 뒤 바깥 연산자를 붙여야 한다.

선지별 해설

① X — *ab* +는 *a* + *b*를 먼저 계산하게 하므로 원식의 최상위 덧셈 구조와 다르다. 틀린 부분 *ab* +는 올바르게 *a*를 먼저 두고 오른쪽 전체를 완성한 뒤 마지막에 +를 붙인 *abc* − *d* × *ef* − ÷ +로 고쳐야 한다.

② O — *bc* −와 *ef* −가 두 괄호를 각각 나타내고, 이어서 (*b* − *c*) × *d*, 그 결과를 (*e* − *f*)로 나눈 뒤 마지막에 *a*를 더하는 순서이므로 *abc* − *d* × *ef* − ÷ +가 된다.

③ X — *ef* ÷ −는 *e* ÷ *f*를 먼저 계산한 뒤 뺄셈을 하게 되어 *e* − *f*를 나타내지 못한다. 틀린 부분 *ef* ÷ −는 *ef* − ÷로 고쳐야 하며, 전체는 *abc* − *d* × *ef* − ÷ +이다.

④ X — *a* + *b* − *c* × *d* ÷ *e* − *f*는 연산자가 피연산자 사이에 있어 후위표기식이 아니라 괄호만 생략한 중위표기식 형태이다. 올바른 후위표기식은 *abc* − *d* × *ef* − ÷ +이다.

⑤ X — 맨 앞의 +처럼 연산자를 피연산자보다 먼저 놓았으므로 후위표기식 규칙에 어긋난다. 선행 연산자 배열을 피연산자 뒤로 옮겨 올바르게 쓰면 *abc* − *d* × *ef* − ÷ +이다.

NoteBase

오답 포인트

최상위 연산자인 덧셈을 먼저 적거나, $e - f$를 완성하기 전에 나눗셈 기호를 붙이는 것이 핵심 함정이다. 특히 ③은 $ef -$와 그 뒤의 $\div$ 순서를 바꾸어 괄호 구조를 훼손한다.

다시 나오면 이렇게 판단

가장 안쪽 괄호를 먼저 후위표기식으로 바꾸고, 그 결과를 하나의 피연산자처럼 취급해 바깥 연산자를 차례로 뒤에 붙인다. 식 전체에서 가장 마지막에 수행되는 최상위 연산자는 후위표기식에서도 맨 마지막에 온다.

암기 포인트

후위표기식은 '피연산자 두 개 뒤에 연산자'이며, 괄호 안부터 바꾸고 최상위 연산자는 맨 끝에 둔다.

NoteBase

5. 기계학습에 대한 설명으로 옳지 않은 것은?

① 회귀 문제는 지도 학습의 한 방법이다.
② 군집화 문제는 비지도 학습의 한 방법이다.
③ 강화 학습은 보상을 기반으로 학습하는 방법이다.
④ 인공신경망은 지도, 비지도, 강화 학습 모두에 이용될 수 있다.
⑤ 비지도 학습은 명시적 정답이 제공되지 않으므로 손실함수가 정의될 수 없다.

정답 5번

출제 포인트

지도 학습·비지도 학습·강화 학습의 구분과 비지도 학습에서도 학습 목적을 수치화한 손실함수를 정의할 수 있다는 점을 묻는 문제이다.

핵심 개념

지도 학습은 입력과 정답 레이블의 대응 관계를 학습하며, 회귀와 분류가 대표적이다.
비지도 학습은 정답 레이블 없이 데이터의 구조나 패턴을 찾으며, 군집화가 대표적이다.
강화 학습은 에이전트가 환경과 상호작용하면서 얻는 보상을 바탕으로 누적 보상을 크게 하는 행동 정책을 학습한다.
손실함수는 학습 목표와 현재 결과의 차이를 수치화한 함수이다. 정답 레이블이 없는 비지도 학습에서도 군집의 응집도나 데이터의 재구성 오차처럼 학습 목표에 맞는 목적함수 또는 손실함수를 정의할 수 있다.

선지별 해설

① O — 회귀는 정답이 연속적인 수치인 학습 데이터를 이용해 입력과 출력의 관계를 학습하는 대표적인 지도 학습 방법이다.
② O — 군집화는 정답 레이블 없이 유사한 데이터끼리 묶어 구조를 찾으므로 대표적인 비지도 학습 방법이다.
③ O — 강화 학습은 행동의 결과로 얻는 보상을 학습 신호로 삼아 더 큰 누적 보상을 얻는 정책을 학습한다.
④ O — 인공신경망은 특정 학습 유형에만 한정된 모델이 아니다. 지도 학습의 분류·회귀, 비지도 학습의 표현 학습, 강화 학습의 정책·가치 함수 근사 등에 이용될 수 있다.
⑤ X — 틀린 표현은 '명시적 정답이 제공되지 않으므로 손실함수가 정의될 수 없다'이다. 올바르게는 비지도 학습에는 정답 레이블이 제공되지 않지만, 군집의 응집도나 재구성 오차 등 학습 목적에 맞는 목적함수 또는 손실함수를 정의할 수 있다.

오답 포인트

⑤는 '정답 레이블이 없다'는 비지도 학습의 특징을 '최적화할 손실함수도 없다'로 과도하게 확대한다. 레이블의 유무와 목적함수·손실함수의 유무는 서로 다른 판단 기준이다.

NoteBase

다시 나오면 이렇게 판단

학습 유형은 학습 신호로 구분한다. 정답 레이블을 사용하면 지도 학습, 데이터 자체의 구조를 찾으면 비지도 학습, 환경의 보상을 이용하면 강화 학습이다. 손실함수의 존재 여부로 지도 학습과 비지도 학습을 구분하지 않는다.

암기 포인트

회귀·분류는 지도, 군집화는 비지도, 보상은 강화이다. 비지도는 정답 레이블이 없을 뿐 학습 목적까지 없는 것은 아니다.

NoteBase

6. 메모리 관리 방안에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. Worst-fit 할당은 가장 큰 공간에 프로세스를 배치하는 방식이다.

ㄴ. Best-fit 할당 방식은 요청하는 메모리 크기에 가장 일치하는 크기의 메모리 블록을 할당함으로써 외부 단편화를 최소화 할 수 있다.

ㄷ. First-fit 할당 방식은 탐색 시 요청한 메모리 크기를 할당할 수 있는 메모리 블록을 찾았을 때 바로 할당함으로써 탐색 시간을 줄일 수 있다.

ㄹ. Buddy 할당 방식은 메모리를 2의 거듭제곱 단위의 크기로 할당하여 내부 단편화를 최소화할 수 있다.

① ㄱ, ㄷ

② ㄱ, ㄹ

③ ㄴ, ㄹ

④ ㄱ, ㄴ, ㄷ

⑤ ㄴ, ㄷ, ㄹ

정답 1번

출제 포인트

이 문제는 가변 분할 메모리 할당 방식인 Worst-fit, Best-fit, First-fit과 Buddy 방식의 특징 및 단편화 유형을 구별하는 문제이다.

핵심 개념

Worst-fit은 들어갈 수 있는 빈 공간 중 가장 큰 곳에 할당하는 방식이다.
Best-fit은 들어갈 수 있는 빈 공간 중 요청 크기와 가장 가까운 작은 곳에 할당하지만, 작은 잔여 공간이 많이 생겨 외부 단편화를 근본적으로 최소화하지는 못한다.
First-fit은 앞에서부터 탐색하다가 처음 발견한 충분한 크기의 빈 공간에 즉시 할당하여 탐색 시간이 짧다.
Buddy 방식은 메모리를 2의 거듭제곱 크기로 분할하고 짝 블록을 병합하기 쉽게 만든 방식이다. 병합으로 외부 단편화를 줄일 수 있지만 요청보다 큰 블록이 배정되어 내부 단편화가 생길 수 있다.

NoteBase

선지별 해설

① O — ㄱ과 ㄷ은 모두 옳다. Worst-fit은 가장 큰 빈 공간을 선택하고, First-fit은 할당 가능한 첫 빈 공간을 찾으면 탐색을 멈춘다.
② X — ㄹ의 '내부 단편화를 최소화'가 틀렸고, 옳은 ㄷ이 빠져 있다. Buddy 방식은 2의 거듭제곱 크기로 할당하여 내부 단편화가 발생할 수 있으며, 올바른 조합은 ㄱ, ㄷ이다.
③ X — ㄴ과 ㄹ이 모두 틀렸다. Best-fit은 작은 외부 조각을 만들 수 있고, Buddy 방식은 내부 단편화가 발생할 수 있다. 올바른 조합은 ㄱ, ㄷ이다.
④ X — ㄴ의 '외부 단편화를 최소화'가 틀렸다. Best-fit은 남는 공간이 가장 작은 블록을 고르지만 작은 빈 공간을 많이 만들 수 있으므로, 올바른 조합은 ㄱ, ㄷ이다.
⑤ X — ㄴ의 '외부 단편화를 최소화'와 ㄹ의 '내부 단편화를 최소화'가 모두 틀렸다. 올바른 조합은 ㄱ, ㄷ이다.

오답 포인트

Best-fit이 남는 공간을 가장 작게 만든다는 사실을 '외부 단편화를 최소화한다'로 확대 해석하거나, Buddy 방식의 병합 용이성을 '내부 단편화 최소화'로 혼동하기 쉽다. Best-fit은 작은 외부 조각을 만들 수 있고, Buddy 방식은 블록 크기를 올려 배정하므로 내부 단편화가 생길 수 있다.

다시 나오면 이렇게 판단

빈 공간 선택 기준은 Worst-fit=가장 큰 공간, Best-fit=가장 작은 충분한 공간, First-fit=처음 발견한 충분한 공간으로 구분한다. Buddy 방식은 2의 거듭제곱 분할과 짝 블록 병합을 먼저 떠올리고, 요청 크기보다 큰 블록 배정에 따른 내부 단편화를 확인한다.

암기 포인트

Worst는 가장 크게, Best는 가장 딱 맞게, First는 가장 먼저 찾은 곳에 할당한다. Buddy는 2의 거듭제곱과 쉬운 병합, 대신 내부 단편화 가능성으로 기억한다.

NoteBase

7. 이진 트리에 대한 설명으로 옳지 않은 것은?

① 일반 트리와 다르게 공집합을 허용한다.
② 모든 노드에게 유일한 부모 노드가 존재한다.
③ 각 노드는 최대 2개의 자식 노드를 갖는다.
④ 어떤 노드에서 루트 노드에 이르는 경로는 유일하다.
⑤ n개의 노드를 가진 이진 트리는 n-1개의 간선을 갖는다.

정답 2번

출제 포인트

이 문제는 이진 트리의 부모·자식 관계, 경로, 간선 수에 관한 기본 성질을 묻는다. 특히 루트 노드는 부모가 없다는 예외를 구별해야 한다.

핵심 개념

이진 트리는 각 노드가 최대 2개의 자식 노드를 가지는 트리이며, 왼쪽 자식과 오른쪽 자식을 구분한다. 공집합도 이진 트리로 정의할 수 있다. 트리에서 루트를 제외한 각 노드는 정확히 하나의 부모를 가지며, 두 노드 사이의 단순 경로는 유일하다. 노드가 n개인 비어 있지 않은 트리의 간선 수는 n-1개이다.

선지별 해설

① O — 이진 트리는 재귀적으로 공집합도 허용하므로 옳다.
② X — '모든 노드'라는 표현이 틀렸다. 루트 노드는 부모가 없고, 루트를 제외한 모든 노드에만 유일한 부모 노드가 존재한다.
③ O — 이진 트리의 각 노드는 자식 노드를 최대 2개까지 가질 수 있으므로 옳다.
④ O — 이진 트리도 트리이므로 임의의 노드에서 루트 노드에 이르는 단순 경로는 유일하다.
⑤ O — 노드가 n개인 비어 있지 않은 트리는 종류와 관계없이 n-1개의 간선을 가지므로 이진 트리에도 적용된다.

오답 포인트

②에서 '유일한 부모'라는 성질 자체는 맞지만 적용 대상을 '모든 노드'로 확대한 것이 함정이다. 루트 노드는 트리의 시작점이므로 부모가 없다.

다시 나오면 이렇게 판단

이진 트리의 선지는 자식 수는 최대 2개, 루트의 부모는 없음, 루트 이외 노드의 부모는 하나, 두 노드 사이의 단순 경로는 하나, 간선 수는 노드 수보다 1개 적음의 순서로 확인한다.

NoteBase

암기 포인트

이진 트리는 자식 최대 2개, 루트는 부모 0개, 나머지는 부모 1개, 간선은 n-1개이다.

NoteBase

8. 캐시 메모리에 대한 설명으로 옳은 것은?

① 완전 연관 사상 캐시의 경우 충돌 실패가 발생한다.
② 캐시의 블록 크기를 키우면 시간적 지역성을 더 잘 활용하여 캐시 실패를 줄일 수 있다.
③ Write-through 정책은 쓰기 동작 발생 시 캐시만 갱신한다.
④ 직접 사상 캐시의 경우 캐시 블록 교체정책이 필요 없다.
⑤ 캐시 쓰기 성능향상을 위한 Write 버퍼는 Write-back 캐시에만 적용된다.

정답 4번

출제 포인트

캐시의 사상 방식과 블록 크기, 쓰기 정책, 교체정책, 쓰기 버퍼의 관계를 구분하는 문제이다.

핵심 개념

완전 연관 사상은 주기억장치의 블록을 캐시의 어느 라인에도 배치할 수 있어 충돌 실패가 발생하지 않는다. 직접 사상은 각 주기억장치 블록이 들어갈 캐시 라인이 하나로 정해져 있어 교체 대상을 선택하는 정책이 필요 없다. Write-through는 캐시와 주기억장치를 함께 갱신하고, Write-back은 우선 캐시만 갱신한 뒤 교체 시 변경된 블록을 주기억장치에 기록한다.

선지별 해설

① X — 틀린 표현은 '충돌 실패가 발생한다'이다. 완전 연관 사상에서는 한 블록을 캐시의 어느 라인에나 둘 수 있으므로 특정 라인에 여러 블록이 몰려 생기는 충돌 실패가 발생하지 않는다.
② X — 틀린 표현은 '시간적 지역성을 더 잘 활용하여 캐시 실패를 줄일 수 있다'이다. 블록 크기를 키워 인접한 데이터를 함께 가져오는 것은 공간적 지역성과 관련된다. 또한 블록이 지나치게 커지면 캐시 라인 수가 줄어들므로 전체 캐시 실패가 반드시 감소하는 것은 아니다.
③ X — 틀린 표현은 '캐시만 갱신한다'이다. Write-through 정책은 쓰기 동작 때 캐시와 주기억장치를 모두 갱신한다. 캐시만 먼저 갱신하고 나중에 주기억장치에 반영하는 것은 Write-back 정책이다.
④ O — 직접 사상에서는 각 주기억장치 블록이 들어갈 캐시 라인이 하나로 고정되므로 여러 후보 중 교체할 블록을 고르는 교체정책이 필요 없다.
⑤ X — 틀린 표현은 'Write-back 캐시에만 적용된다'이다. 쓰기 버퍼는 특히 Write-through에서 주기억장치 쓰기가 끝날 때까지 CPU가 기다리는 시간을 줄이는 데 사용되며, Write-back에만 한정되지 않는다.

NoteBase

오답 포인트

③의 '캐시만 갱신'은 Write-back의 특징을 Write-through에 바꿔 붙인 표현이다. ⑤는 Write-back과 쓰기 버퍼를 배타적으로 연결하지만, 쓰기 버퍼의 대표적 목적은 Write-through의 느린 주기억장치 쓰기를 완충하는 것이다.

다시 나오면 이렇게 판단

사상 방식은 배치 가능한 라인 수로 판단한다. 어느 라인이나 가능하면 완전 연관 사상, 한 라인으로 고정되면 직접 사상이다. 쓰기 정책은 주기억장치를 즉시 함께 갱신하면 Write-through, 교체 시 나중에 반영하면 Write-back으로 구분한다.

암기 포인트

완전 연관은 충돌 실패 없음, 직접 사상은 교체 선택 없음, Write-through는 캐시와 주기억장치 동시 갱신이다.

NoteBase

9. ㉠, ㉡에 들어갈 단어를 바르게 연결한 것은?

> 기존 현실 세계의 객체에서 불필요한 속성을 제거하고, 중요한 정보만 클래스로 표현하는 일종의 모델링 기법으로 객체지향 프로그래밍에서는 클래스를 통해서 ㉠ 을/를 지원하고 있다.
>
> 객체의 상세한 내용을 객체 외부에 철저히 숨기고 단순히 메시지만으로 객체와의 상호작용을 하게 하는 것을 ㉡ (이)라고 말한다.

	㉠	㉡
①	추상화	다형성
②	추상화	캡슐화
③	다형성	캡슐화
④	상속	다형성
⑤	캡슐화	추상화

정답 2번

출제 포인트

객체지향 프로그래밍의 핵심 개념인 추상화와 캡슐화의 정의를 구별하는 문제이다.

핵심 개념

추상화: 현실 세계의 객체에서 불필요한 세부 속성을 제외하고 중요한 특징만 뽑아 클래스 등으로 표현하는 것이다. 키워드는 '핵심만 모델링'이다.
캡슐화: 객체의 데이터와 기능을 하나로 묶고 내부 구현을 외부에서 직접 다루지 못하게 숨긴 뒤, 정해진 인터페이스를 통해 상호작용하게 하는 것이다. 키워드는 '정보 은닉과 인터페이스'이다.
상속: 기존 클래스의 속성과 기능을 새로운 클래스가 물려받아 재사용하거나 확장하는 것이다.
다형성: 같은 인터페이스나 메시지에 대해 객체의 실제 유형에 따라 서로 다르게 동작할 수 있는 성질이다.

NoteBase

선지별 해설

① X — ㉠의 '추상화'는 맞지만, ㉡의 '다형성'이 틀렸다. 객체의 내부 내용을 숨기고 외부에는 인터페이스만 제공하는 ㉡은 캡슐화이다. 따라서 '추상화-캡슐화'로 고쳐야 한다.
② O — 중요한 정보만 뽑아 클래스로 표현하는 ㉠은 추상화이고, 상세 내용을 숨긴 채 메시지로 상호작용하게 하는 ㉡은 캡슐화이다. 두 개념이 모두 바르게 연결되었다.
③ X — ㉡의 '캡슐화'는 맞지만, ㉠의 '다형성'이 틀렸다. 불필요한 속성을 제거하고 중요한 정보만 모델링하는 것은 추상화이다. 따라서 '추상화-캡슐화'로 고쳐야 한다.
④ X — ㉠의 '상속'과 ㉡의 '다형성'이 모두 틀렸다. ㉠은 추상화이고 ㉡은 캡슐화이므로 '추상화-캡슐화'로 고쳐야 한다.
⑤ X — 두 개념의 위치가 서로 뒤바뀌었다. ㉠의 '캡슐화'는 추상화로, ㉡의 '추상화'는 캡슐화로 고쳐야 한다.

오답 포인트

'속성을 제거한다'는 표현 때문에 내부 내용을 숨기는 캡슐화와 혼동하기 쉽다. ㉠은 현실 객체를 모델링할 때 불필요한 특징을 제외한다는 뜻이므로 추상화이고, ㉡은 객체 내부의 상세 내용을 외부에 숨긴다는 뜻이므로 캡슐화이다.

다시 나오면 이렇게 판단

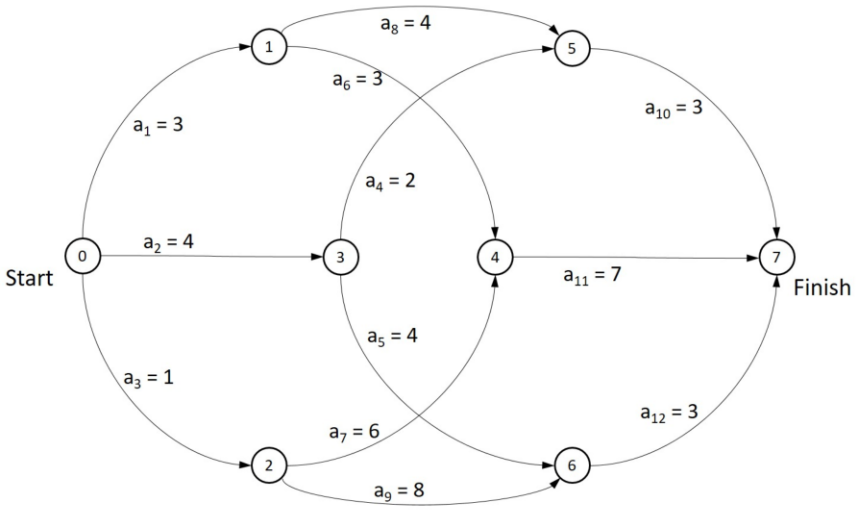
현실의 복잡한 대상에서 핵심 특징만 뽑아 모델을 만드는 설명이면 추상화로 판단한다. 객체 내부의 데이터나 구현을 숨기고 공개된 인터페이스로만 접근하게 하는 설명이면 캡슐화로 판단한다.

암기 포인트

추상화는 '핵심만 뽑기', 캡슐화는 '내부는 숨기고 창구만 열기'이다.

NoteBase

10. 다음 간선작업 네트워크에서 임계경로의 길이는?



① 10
② 11
③ 13
④ 14
⑤ 16

정답 4번

출제 포인트

간선작업 네트워크에서 각 결합점의 최조발생시간을 순방향으로 계산하여 시작점에서 종료점까지의 임계경로 길이를 구하는 문제이다.

조건 정리

시작 결합점 0의 최조발생시간은 0이다. 하나의 결합점으로 여러 작업이 들어오면 그 결합점의 최조발생시간은 각 선행 경로의 도착시간 중 최댓값이다. 구해야 할 값은 종료 결합점 7의 최조발생시간, 즉 전체 공사의 최소 완료시간이자 임계경로의 길이이다.

사용할 공식

결합점 j의 최조발생시간은 다음과 같다.

$$E(j) = \max_{(i,j)}\{E(i)+t_{ij}\}$$

여기서 t_{ij}는 결합점 i에서 j로 가는 작업의 소요시간이다. 모든 선행 작업이 끝나야 다음 결합점에 도달하므로 합계 중 최댓값을 사용한다.

NoteBase

계산 과정

1. 시작점에서 $E(1) = 3$, $E(3) = 4$, $E(2) = 1$이다.
2. 중간 결합점은 $E(5) = \max(3 + 4, 4 + 2) = 7$, $E(4) = \max(3 + 3, 1 + 6) = 7$, $E(6) = \max(4 + 4, 1 + 8) = 9$이다.
3. 종료점은 $E(7) = \max(7 + 3, 7 + 7, 9 + 3) = \max(10, 14, 12) = 14$이다.
4. 최댓값 14를 만드는 경로는 0→2→4→7이며, 길이는 $1 + 6 + 7 = 14$이다.

선지 판단

① X — 10은 경로 0→1→5→7의 길이 $3 + 4 + 3$일 뿐 전체 경로 중 최댓값이 아니다. ‘10’은 틀린 값이며 올바른 임계경로 길이는 14이다.
② X — 11은 경로 0→3→6→7의 길이 $4 + 4 + 3$일 뿐이다. ‘11’은 틀린 값이며 올바른 임계경로 길이는 14이다.
③ X — 13은 경로 0→1→4→7의 길이 $3 + 3 + 7$이다. 결합점 4에는 0→2→4가 7의 시각에 더 늦게 도착하므로 이를 기준으로 해야 한다. ‘13’은 틀린 값이며 올바른 임계경로 길이는 14이다.
④ O — 경로 0→2→4→7의 길이가 $1 + 6 + 7 = 14$이고, 종료점 7의 세 도착시간 10, 14, 12 중 최댓값이므로 맞다.
⑤ X — 16은 네트워크의 방향을 따라 연결되는 어떤 시작점-종료점 경로의 길이도 아니다. ‘16’은 틀린 값이며 올바른 임계경로 길이는 14이다.

오답 포인트

특정 시작점-종료점 경로 하나의 합만 구하거나, 합이 작은 경로를 선택하면 임계경로를 놓친다. 특히 결합점 4에서 0→1→4의 도착시간 6만 사용하면 13이 나오지만, 실제로는 0→2→4의 도착시간 7을 선택해야 한다.

다시 나오면 이렇게 판단

시작점의 시간을 0으로 두고 화살표 방향으로 진행하면서, 유입 간선이 하나이면 선행시간에 작업시간을 더하고 유입 간선이 여러 개이면 그 합들의 최댓값을 기록한다. 종료점에 기록된 값이 임계경로의 길이이다.

암기 포인트

분기에서는 각각 더하고, 합류점에서는 가장 늦은 도착시간인 최댓값을 선택한다. 이 문제의 임계경로는 0→2→4→7, 길이는 14이다.

NoteBase

11. 다음 gcd 함수에 대해 gcd(54, 30)을 호출하였을 때 반환되는 값은?

```
int gcd(int max, int min) {
    if (min == 0) return max;
    else return gcd(min, max % min);
}
```

① 6
② 0
③ 54
④ 30
⑤ 24

정답 1번

출제 포인트

이 문제는 나머지가 0이 될 때까지 두 수를 '작은 수와 나머지'로 바꾸어 호출하는 유클리드 호제법의 재귀 실행 과정을 묻는다.

조건 정리

주어진 호출: gcd(54, 30)
종료 조건: 두 번째 인수 min이 0이면 첫 번째 인수 max를 반환한다.
재귀 조건: min이 0이 아니면 gcd(min, max % min)을 호출한다.

사용할 공식

유클리드 호제법은 (a,b)=(b,a b)를 이용하며, $b = 0$이면 a가 최대공약수이다. 코드의 재귀 호출과 종료 조건이 이 원리를 그대로 구현하므로 이 공식을 사용한다.

계산 과정

1. 54 % 30 = 24이므로 gcd(54, 30)은 gcd(30, 24)를 호출한다.
2. 30 % 24 = 6이므로 gcd(30, 24)는 gcd(24, 6)을 호출한다.
3. 24 % 6 = 0이므로 gcd(24, 6)은 gcd(6, 0)을 호출한다.
4. 두 번째 인수가 0이므로 첫 번째 인수 6을 반환한다. 따라서 최초 호출의 반환값도 6이다.

NoteBase

선지 판단

① O — 재귀 호출이 gcd(6, 0)에 도달하고 종료 조건에 따라 6을 반환하므로 맞다.
② X — 0은 마지막 나머지일 뿐 반환값이 아니다. 올바른 반환값은 gcd(6, 0)의 첫 번째 인수인 6이다.
③ X — 54는 최초 호출의 첫 번째 인수이지만 두 번째 인수가 0이 아니므로 즉시 반환되지 않는다. 올바른 반환값은 6이다.
④ X — 30은 첫 재귀 호출의 첫 번째 인수이지만 그 호출에서도 두 번째 인수가 0이 아니다. 올바른 반환값은 6이다.
⑤ X — 24는 첫 번째 나눗셈에서 나온 중간 나머지일 뿐이다. 재귀를 종료 조건까지 계속하면 올바른 반환값은 6이다.

오답 포인트

마지막에 계산된 나머지 0을 답으로 고르거나, 중간 나머지 24에서 재귀 추적을 멈추기 쉽다. 이 함수는 min이 0인 호출에서 max를 반환하므로 마지막 0이 아닌 나머지 6이 답이다.

다시 나오면 이렇게 판단

각 호출을 gcd(max, min) → gcd(min, max % min) 형태로 차례로 적고, 두 번째 인수가 0이 된 호출의 첫 번째 인수를 반환값으로 판단한다.

암기 포인트

유클리드 호제법의 반환값은 마지막 나머지 0이 아니라 마지막 0이 아닌 나머지이다.

NoteBase

12. Wi-Fi 6(IEEE 802.11ax)에 대한 설명으로 옳지 않은 것은?

① 변조 방식으로 1024-QAM을 사용한다.
② CSMA/CA를 사용한다.
③ 주파수분할 방식으로 FDMA를 사용한다.
④ 보안 표준으로 WPA3를 사용한다.
⑤ 주파수대역 2.4GHz와 5GHz를 지원한다.

정답 3번

출제 포인트

Wi-Fi 6(IEEE 802.11ax)의 변조 방식, 매체 접근 방식, 다중 사용자 전송 방식, 보안 규격, 지원 주파수 대역을 구별하는 문제이다.

핵심 개념

Wi-Fi 6는 IEEE 802.11ax에 해당하는 무선 LAN 규격으로, 1024-QAM과 OFDMA를 이용해 전송 효율을 높이고 WPA3 및 2.4*GHz*·5*GHz* 대역을 지원한다.
CSMA/CA는 단말이 무선 매체의 사용 여부를 확인하고 충돌을 피하며 접근하는 방식이고, OFDMA는 하나의 채널을 여러 직교 부반송파 묶음으로 나누어 여러 사용자에게 동시에 할당하는 다중 접속 방식이다.

선지별 해설

① O — Wi-Fi 6는 높은 데이터 전송률을 위해 1024-QAM 변조를 지원한다.
② O — 무선 LAN의 매체 접근에는 충돌을 사전에 피하는 CSMA/CA가 사용된다.
③ X — 틀린 표현은 '주파수분할 방식으로 FDMA를 사용한다'이다. Wi-Fi 6의 핵심 다중 사용자 전송 방식은 FDMA가 아니라 OFDMA이다. OFDMA는 채널을 직교 부반송파 단위의 자원으로 나누어 여러 사용자에게 동시에 배분한다.
④ O — Wi-Fi 6는 강화된 무선 보안 규격인 WPA3를 지원한다.
⑤ O — Wi-Fi 6는 2.4*GHz*와 5*GHz* 주파수 대역을 모두 지원한다.

오답 포인트

③의 FDMA와 OFDMA를 같은 방식으로 보는 것이 함정이다. 둘 다 주파수 자원을 나누지만, Wi-Fi 6의 대표 기술은 직교 부반송파 자원을 사용자별로 유연하게 배분하는 OFDMA이다. 또한 CSMA/CA는 매체 접근 방식이고 OFDMA는 다중 사용자 전송 방식이므로 서로 대체되는 개념이 아니다.

NoteBase

다시 나오면 이렇게 판단

Wi-Fi 6의 특징을 묻는 문항에서는 '1024-QAM, OFDMA, CSMA/CA, WPA3, 2.4*GHz*·5*GHz*'를 기준으로 판단한다. 선지에 FDMA가 제시되면 OFDMA를 잘못 바꾼 표현인지 확인한다.

암기 포인트

Wi-Fi 6의 다중 사용자 전송은 FDMA가 아니라 OFDMA이다.

NoteBase

13. 관계형 데이터베이스에 대한 설명으로 옳은 것은?

① 한 릴레이션에는 똑같은 애트리뷰트가 포함될 수 없으며 포함된 애트리뷰트 사이에는 순서가 있다.
② 후보키가 되기 위해서는 유일성과 효율성을 둘 다 만족해야 한다.
③ 2개 이상의 후보키 중에서 기본키로 선택되지 않은 나머지 후보키를 슈퍼키라고 한다.
④ 참조 무결성을 위해 참조 대상이 존재하지 않는 외래키를 허용하지 않는다.
⑤ 후보키는 모든 튜플을 유일하게 식별할 수 있는 하나 또는 몇 개의 속성들의 부분 집합으로 최소성을 만족하지 않는다.

정답 4번

출제 포인트

관계형 데이터베이스에서 릴레이션의 속성, 후보키·기본키·대체키·슈퍼키, 참조 무결성의 의미를 구별하는 문제이다.

핵심 개념

릴레이션은 튜플의 집합이며, 속성과 튜플의 논리적 순서는 의미가 없다.
후보키는 각 튜플을 유일하게 식별하는 속성 집합으로 유일성과 최소성을 모두 만족한다. 후보키 중 기본키로 선택되지 않은 키는 대체키이고, 슈퍼키는 유일성은 만족하지만 최소성까지 반드시 만족할 필요는 없다.
참조 무결성은 외래키 값이 참조되는 릴레이션의 기본키 값과 일치하거나 널이어야 한다는 제약이다.

선지별 해설

① X — 한 릴레이션에서 속성 이름은 서로 달라야 하지만, 속성 사이에는 순서가 없다. 틀린 표현은 '포함된 애트리뷰트 사이에는 순서가 있다'이며, 올바르게는 '애트리뷰트 사이에는 순서가 없다'이다.
② X — 후보키의 조건은 유일성과 최소성이다. 틀린 표현은 '유일성과 효율성'이며, 올바르게는 '유일성과 최소성을 모두 만족해야 한다'이다.
③ X — 여러 후보키 가운데 기본키로 선택되지 않은 나머지 후보키는 대체키라고 한다. 틀린 표현은 '슈퍼키'이며, 올바르게는 '대체키'이다.
④ O — 참조 무결성에 따라 외래키는 참조 대상 릴레이션에 존재하는 기본키 값을 가리켜야 하므로, 참조 대상이 존재하지 않는 외래키 값은 허용되지 않는다.
⑤ X — 후보키는 모든 튜플을 유일하게 식별할 수 있는 하나 이상의 속성 집합이면서 최소성을 만족한다. 틀린 표현은 '최소성을 만족하지 않는다'이며, 올바르게는 '최소성을 만족한다'이다.

NoteBase

오답 포인트

③은 후보키도 유일성을 만족하므로 넓게 보면 슈퍼키에 포함된다는 점을 이용한 함정이다. 그러나 기본키로 선택되지 않은 후보키의 명칭을 묻는 문맥에서는 대체키가 정확하다. 또한 ②와 ⑤에서는 후보키의 두 조건인 유일성과 최소성을 효율성이나 '최소성 불충족'으로 바꾸었다.

다시 나오면 이렇게 판단

키 관련 선지는 먼저 유일성과 최소성을 확인한다. 유일성만 요구하면 슈퍼키, 유일성과 최소성을 모두 요구하면 후보키, 후보키 중 선택된 것은 기본키, 선택되지 않은 것은 대체키로 판단한다. 외래키는 참조 대상의 존재 여부를 확인한다.

암기 포인트

슈퍼키는 유일성, 후보키는 유일성+최소성, 후보키 중 선택은 기본키·미선택은 대체키이다.

NoteBase

14. IP주소 210.100.100.3이 속한 네트워크를 3개의 작은 서브 네트워크로 나누기 위해 서브넷 마스크를 255.255.255.192로 설정하였다. 이때 각 서브 네트워크의 호스트 개수는?

① 2^5-1
② 2^6
③ 2^6-1
④ 2^6-2
⑤ 2^7-1

정답 4번

출제 포인트

서브넷 마스크에서 호스트 비트 수를 구하고, 네트워크 주소와 브로드캐스트 주소를 제외한 사용 가능한 호스트 수를 계산하는 문제이다.

조건 정리

주어진 서브넷 마스크는 255.255.255.192이다. 마지막 옥텟 192는 이진수로 11000000이므로 프리픽스 길이는 /26이고, 호스트 비트는 32 − 26 = 6비트이다. 구해야 할 값은 각 서브네트워크에서 사용할 수 있는 호스트 주소의 개수이다.

사용할 공식

호스트 비트가 h개일 때 사용 가능한 호스트 수는 다음과 같다.

$2^h - 2$

각 서브네트워크의 전체 주소 중 네트워크 주소와 브로드캐스트 주소는 호스트에 할당할 수 없으므로 2개를 뺀다.

계산 과정

1. 255.255.255.192는 /26이므로 호스트 비트 수는 32 − 26 = 6이다.
2. 전체 주소 수는 $2^6 = 64$개이다.
3. 네트워크 주소와 브로드캐스트 주소를 제외하면 $2^6 - 2 = 62$개이다.

따라서 각 서브네트워크의 사용 가능한 호스트 수는 62개이다.

NoteBase

선지 판단

① X — $2^5 - 1$은 호스트 비트 수를 5비트로 잘못 보고 예약 주소도 1개만 제외한 식이다. 올바른 식은 $2^6 - 2$이다.
② X — 2^6은 전체 주소 수 64개이며 네트워크 주소와 브로드캐스트 주소를 제외하지 않았다. 사용 가능한 호스트 수는 $2^6 - 2$이다.
③ X — $2^6 - 1$은 예약 주소를 1개만 제외한 식이다. 네트워크 주소와 브로드캐스트 주소 2개를 제외하여 $2^6 - 2$로 계산한다.
④ O — 호스트 비트가 6개이고 예약 주소 2개를 제외하므로 $2^6 - 2 = 62$개가 맞다.
⑤ X — $2^7 - 1$은 호스트 비트 수와 제외할 예약 주소 수를 모두 잘못 적용한 식이다. 올바른 호스트 비트 수는 6개이고 식은 $2^6 - 2$이다.

오답 포인트

서브넷 하나의 전체 주소 수 64개와 실제 할당 가능한 호스트 수 62개를 혼동하기 쉽다. 또한 마지막 옥텟 192에서 1의 개수가 2개라는 점을 놓치면 호스트 비트 수를 잘못 정하게 된다.

다시 나오면 이렇게 판단

서브넷 마스크를 이진수로 바꾸어 프리픽스 길이를 확인하고, h=32-프리픽스 길이로 호스트 비트 수를 구한 뒤 $2^h - 2$를 적용한다.

암기 포인트

/26은 호스트 비트 6개, 전체 주소 64개, 사용 가능한 호스트 62개이다.

NoteBase

15. 원형 연결 리스트 CL에 x값을 갖는 노드 new를 pre가 가리키는 노드의 다음 노드로 삽입하고자 할 때, 빈 칸에 들어갈 의사코드로 옳은 것은?

```
begin insertMiddleNode(CL, pre, x)
  new ← getNode();  // 삽입할 노드를 할당받음
  new.data ← x;     // 새 노드의 데이터 필드에 x를 저장함
  if(CL == null) then  // CL이 공백 리스트인 경우
    CL ← new;        // CL이 노드 new를 첫번째 노드로 가리킴
    new.link ← new;  // 참조변수 new의 값을 노드 new가 가리키는
                     // 새 노드의 링크 필드(new.link)에 저장함
  else
    [                                    ]

end insertMiddleNode(CL, pre, x)
```

① pre.link ← new;
② new.link ← pre;
③ new.link ← pre.link;
new.link ← pre;
④ new.link ← pre.link;
pre ← new;
⑤ new.link ← pre.link;
pre.link ← new;

정답 5번

출제 포인트

원형 연결 리스트에서 새 노드를 중간에 삽입할 때 기존의 다음 노드를 잃지 않도록 링크를 갱신하는 순서를 묻는 문제이다.

올바른 순서

1. new.link ← pre.link — 새 노드가 원래 pre 다음에 있던 노드를 가리키게 한다.
2. pre.link ← new — pre가 새 노드를 다음 노드로 가리키게 한다.

왜 이 순서인가: pre.link를 먼저 바꾸면 원래 후속 노드의 주소를 잃을 수 있으므로, 그 주소를 new.link에 먼저 보존해야 한다.

NoteBase

선지별 해설

① X — pre.link ← new만 수행하면 새 노드가 원래 후속 노드를 가리키지 못한다. 틀린 부분은 new.link 설정이 빠진 것이며, 올바르게는 new.link ← pre.link를 먼저 수행한 뒤 pre.link ← new를 수행한다.

② X — new.link ← pre는 새 노드가 선행 노드 pre를 가리키게 하므로 삽입 방향이 잘못되고, pre.link도 new로 바뀌지 않는다. 올바르게는 new.link ← pre.link, pre.link ← new이다.

③ X — 첫 문장에서 원래 후속 노드를 new.link에 저장하지만, 다음 문장의 new.link ← pre가 그 값을 덮어쓴다. 또한 pre.link가 new를 가리키도록 바뀌지 않는다. 올바르게는 두 번째 문장이 pre.link ← new이어야 한다.

④ X — new.link ← pre.link는 맞지만 pre ← new는 참조변수 pre 자체만 새 노드로 바꾸며 기존 선행 노드의 link 필드를 수정하지 않는다. 올바르게는 두 번째 문장이 pre.link ← new이어야 한다.

⑤ O — new.link에 기존 후속 노드를 먼저 연결하고 pre.link를 new로 바꾸므로, pre → new → 기존 후속 노드의 연결이 완성된다.

오답 포인트

④의 pre ← new와 ⑤의 pre.link ← new를 혼동하기 쉽다. 전자는 지역 참조변수 pre의 값만 바꾸지만, 후자는 실제 선행 노드의 링크 필드를 변경한다. 또한 두 대입문의 순서를 바꾸면 기존 후속 노드의 주소를 잃을 수 있다.

다시 나오면 이렇게 판단

연결 리스트 삽입은 먼저 새 노드가 기존 후속 노드를 가리키게 하고, 그다음 선행 노드가 새 노드를 가리키게 하는 순서로 판단한다. 각 대입문의 왼쪽이 참조변수 자체인지 노드의 link 필드인지도 구분한다.

암기 포인트

새 노드의 뒤를 먼저 연결하고, 선행 노드의 앞을 나중에 연결한다: new.link ← pre.link, pre.link ← new.

NoteBase

16. 다음 동작에 해당하는 정렬 방법은?

```
정렬할 원소: 34, 27, 19, 50, 2, 21, 13
 1단계: 2 27 19 50 34 21 13
 2단계: 2 13 19 50 34 21 27
 3단계: 2 13 19 50 34 21 27
 4단계: 2 13 19 21 34 50 27
 5단계: 2 13 19 21 27 50 34
 6단계: 2 13 19 21 27 34 50
```

① 퀵 정렬
② 셸 정렬
③ 선택 정렬
④ 버블 정렬
⑤ 삽입 정렬

정답 3번

출제 포인트

각 단계에서 미정렬 구간의 최솟값을 골라 그 구간의 맨 앞 원소와 교환하는 선택 정렬의 동작을 식별하는 문제이다.

올바른 순서

1단계: 전체 구간의 최솟값 2를 첫 원소 34와 교환한다.
2단계: 나머지 구간의 최솟값 13을 두 번째 원소 27과 교환한다.
3단계: 나머지 구간의 최솟값 19가 이미 세 번째 자리에 있으므로 배열이 변하지 않는다.
4단계: 나머지 구간의 최솟값 21을 네 번째 원소 50과 교환한다.
5단계: 나머지 구간의 최솟값 27을 다섯 번째 원소 34와 교환한다.
6단계: 남은 두 원소 중 최솟값 34를 여섯 번째 원소 50과 교환하여 정렬을 마친다.
왜 이 순서: 한 단계가 끝날 때마다 미정렬 구간의 최솟값이 왼쪽 경계에 확정되므로 정렬된 구간이 왼쪽부터 한 칸씩 늘어난다.

NoteBase

선지별 해설

① X — 퀵 정렬은 피벗을 기준으로 작은 값과 큰 값을 분할한 뒤 각 부분을 재귀적으로 정렬한다. 제시된 과정처럼 매 단계 최솟값 하나를 골라 앞자리에 확정하지 않는다. 틀린 표현 '퀵 정렬'은 '선택 정렬'로 고쳐야 한다.
② X — 셸 정렬은 일정한 간격으로 떨어진 원소들을 부분 수열로 묶어 삽입 정렬하고, 간격을 줄여 가며 정렬한다. 제시된 과정에는 간격별 정렬이 없다. 틀린 표현 '셸 정렬'은 '선택 정렬'로 고쳐야 한다.
③ O — 1단계에서 2, 2단계에서 13, 3단계에서 19를 차례로 미정렬 구간의 맨 앞으로 확정한다. 이는 선택 정렬의 전형적인 과정이다.
④ X — 버블 정렬은 인접한 두 원소를 반복 비교·교환하며 한 번의 순회마다 큰 값이 오른쪽 끝으로 이동한다. 제시된 과정은 인접 원소의 연속 교환이 아니라 최솟값과 앞자리의 직접 교환이다. 틀린 표현 '버블 정렬'은 '선택 정렬'로 고쳐야 한다.
⑤ X — 삽입 정렬은 다음 원소를 이미 정렬된 왼쪽 구간의 알맞은 위치에 삽입하고 그 사이 원소들을 이동한다. 예를 들어 처음 두 원소 34와 27을 처리하면 27, 34로 시작해야 하므로 제시된 1단계와 다르다. 틀린 표현 '삽입 정렬'은 '선택 정렬'로 고쳐야 한다.

오답 포인트

3단계에서 배열이 변하지 않아 정렬이 멈춘 것처럼 보일 수 있다. 그러나 미정렬 구간의 최솟값 19가 이미 해당 자리에 있으면 선택 정렬은 교환 없이 다음 단계로 진행한다.

다시 나오면 이렇게 판단

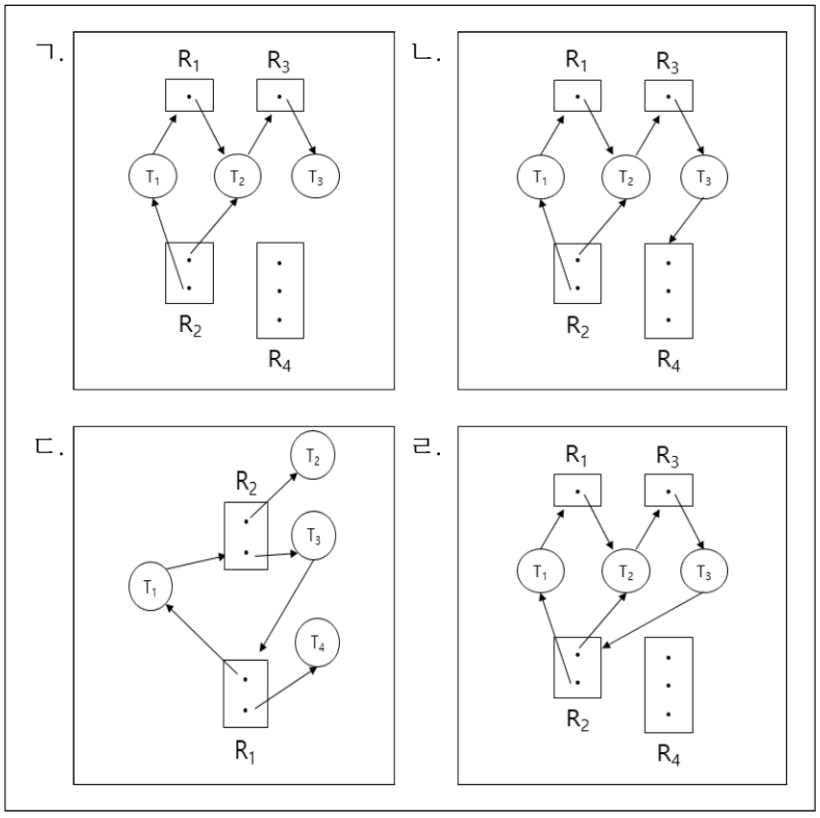
각 단계 뒤 왼쪽 자리가 최솟값부터 하나씩 확정되는지 확인한다. 미정렬 구간의 최솟값을 찾아 그 구간의 첫 원소와 교환한다면 선택 정렬이다.

암기 포인트

선택 정렬은 '남은 구간의 최솟값 선택 → 맨 앞과 교환 → 왼쪽 한 자리 확정'의 반복이다.

NoteBase

17. 다음 중 교착상태인 시스템 자원 할당 그래프에 해당하는 것만을 모두 고르면? (단, 자원 할당 그래프에서 T_i는 쓰레드(원형), R_j는 자원 유형(사각형), $T_i \rightarrow R_j$는 요청 간선, $R_j \rightarrow T_i$는 할당 간선이고 각 자원 유형 R_j는 한 개 이상의 인스턴스(사각형 내 점)를 가진다)



① ㄱ
② ㄴ
③ ㄹ
④ ㄱ, ㄴ
⑤ ㄷ, ㄹ

정답 3번

출제 포인트

자원 할당 그래프에서 요청 간선과 할당 간선을 따라 대기 관계를 찾고, 각 자원의 모든 인스턴스가 대기 중인 스레드들에 묶여 있어 누구도 진행할 수 없는지를 판단하는 문제이다. 인스턴스가 여러 개인 자원에서는 순환이 있다는 사실만으로 교착상태가 확정되지 않는다.

NoteBase

선지별 해설

① X — ㄱ에서는 T_1이 T_2가 가진 R_1을 기다리고, T_2는 T_3가 가진 R_3을 기다린다. 그러나 T_3은 다른 자원을 요청하지 않으므로 작업을 끝내고 R_3을 반납할 수 있다. 따라서 대기가 연쇄적으로 해소되며 교착상태가 아니다. 틀린 표현 'ㄱ'은 'ㄹ'로 고쳐야 한다.
② X — ㄴ에서는 T_3이 R_4를 요청하지만 R_4의 세 인스턴스는 할당되어 있지 않다. T_3이 자원을 받아 진행한 뒤 R_3을 반납할 수 있으므로 교착상태가 아니다. 틀린 표현 'ㄴ'은 'ㄹ'로 고쳐야 한다.
③ O — ㄹ에서는 T_1이 T_2가 가진 R_1을 기다리고, T_2는 T_3가 가진 R_3을 기다리며, T_3은 두 인스턴스 모두 T_1과 T_2에 할당된 R_2를 기다린다. 즉 $T_1 \rightarrow R_1 \rightarrow T_2 \rightarrow R_3 \rightarrow T_3 \rightarrow R_2 \rightarrow T_1$의 닫힌 대기가 형성되고 이를 풀 수 있는 가용 인스턴스도 없어 교착상태이다.
④ X — ㄱ과 ㄴ은 모두 대기를 끝낼 수 있는 스레드 또는 가용 자원이 있어 교착상태가 아니다. 따라서 'ㄱ, ㄴ'이 아니라 'ㄹ'만 해당한다.
⑤ X — ㄷ에는 $T_1 \rightarrow R_2 \rightarrow T_3 \rightarrow R_1 \rightarrow T_1$의 순환이 보이지만, R_2의 한 인스턴스를 가진 T_2와 R_1의 한 인스턴스를 가진 T_4는 다른 자원을 기다리지 않는다. 이들이 작업을 끝내 자원을 반납하면 순환이 해소되므로 ㄷ은 교착상태가 아니다. 따라서 'ㄷ, ㄹ'은 'ㄹ'로 고쳐야 한다.

오답 포인트

ㄷ의 순환만 보고 곧바로 교착상태라고 판단하기 쉽다. 자원 유형에 인스턴스가 하나뿐이면 순환이 교착상태를 뜻하지만, 여러 인스턴스가 있으면 순환 밖의 스레드가 자원을 반납하여 대기를 끊을 수 있는지까지 확인해야 한다.

다시 나오면 이렇게 판단

먼저 요청 간선은 스레드에서 자원으로, 할당 간선은 자원에서 스레드로 따라간다. 그다음 순환을 찾고, 순환에 포함된 자원의 모든 인스턴스가 묶여 있는지와 자원을 반납해 순환을 끊을 수 있는 스레드가 있는지를 함께 확인한다.

암기 포인트

단일 인스턴스는 순환이면 교착, 다중 인스턴스는 순환에 더해 탈출 가능한 인스턴스가 없는지까지 확인한다.

NoteBase

18. 큐(queue)를 사용하여 처리하기에 적절한 것만을 모두 고르면?

ㄱ. 미로 탐색 문제에서 가장 최근에 방문한 길들을 기억할 때
ㄴ. 키보드에서 입력된 키 값을 잠시 저장할 때
ㄷ. 그래프 너비 우선 탐색(BFS)을 할 때
ㄹ. 서브루틴 호출시 복귀 주소를 저장할 때

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

정답 2번

출제 포인트

큐와 스택의 처리 순서를 구분하고, 각 자료구조가 사용되는 사례를 판단하는 문제이다.

핵심 개념

큐(queue)는 먼저 들어온 데이터가 먼저 나가는 선입선출(FIFO) 자료구조이다. 입력 순서를 유지해 처리해야 하는 키보드 입력 버퍼와 가까운 정점부터 탐색하는 너비 우선 탐색(BFS)에 사용된다.
스택(stack)은 나중에 들어온 데이터가 먼저 나가는 후입선출(LIFO) 자료구조이다. 가장 최근의 경로나 상태로 되돌아가는 미로 탐색과 서브루틴의 복귀 주소 저장에 사용된다.

NoteBase

선지별 해설

① X
ㄴ은 큐의 사용 사례이지만 ㄱ은 스택의 사용 사례이므로 ㄱ, ㄴ을 모두 고른 조합은 틀리다. 틀린 표현은 ㄱ의 포함이며, 큐에 적절한 항목은 ㄴ, ㄷ이다.

② O
ㄴ은 입력된 키를 순서대로 처리하는 키보드 입력 버퍼의 사례이고, ㄷ은 먼저 발견한 정점부터 처리하는 BFS의 사례이므로 둘 다 큐에 적절하다.

③ X
ㄷ은 큐의 사용 사례이지만 ㄹ의 복귀 주소는 가장 최근 호출부터 꺼내야 하므로 스택에 저장한다. 틀린 표현은 ㄹ의 포함이며, 큐에 적절한 항목은 ㄴ, ㄷ이다.

④ X
ㄴ과 ㄷ은 큐의 사용 사례이지만 ㄱ은 최근에 방문한 길부터 되돌아가는 스택의 사용 사례이다. 틀린 표현은 ㄱ의 포함이며, 올바른 조합은 ㄴ, ㄷ이다.

⑤ X
ㄴ과 ㄷ은 큐의 사용 사례이지만 ㄹ은 가장 최근 서브루틴의 복귀 주소부터 사용하는 스택의 사례이다. 틀린 표현은 ㄹ의 포함이며, 올바른 조합은 ㄴ, ㄷ이다.

오답 포인트

ㄱ의 '가장 최근'과 ㄹ의 '복귀 주소'가 핵심 함정이다. 둘 다 가장 나중에 저장한 정보를 먼저 꺼내므로 큐가 아니라 스택에 해당한다.

다시 나오면 이렇게 판단

먼저 들어온 작업부터 순서대로 처리하면 큐, 가장 최근에 저장한 상태부터 되돌아가거나 꺼내면 스택으로 판단한다.

암기 포인트

큐는 FIFO·키보드 버퍼·BFS, 스택은 LIFO·되돌아가기·복귀 주소이다.

NoteBase

19. 소프트웨어 생명주기 모형에 대한 설명으로 옳지 않은 것은?

① 나선형 모형: 보헴(Boehm)이 제안한 것으로, 폭포수 모형과 프로토타입 모형의 장점에 위험분석 기능이 추가된 모델이다.

② V-모형: 개발 작업과 검증 작업 사이의 관계를 명백히 드러내 놓은 폭포수 모델의 변형으로 각 개발 단계의 테스트에 중점을 둔다.

③ 애자일 기법: 개발팀이 설계와 문서화보다는 소프트웨어 자체에 초점을 맞추도록 한다.

④ 익스트림 프로그래밍: 시험 개발과 검증에 사용자 참여를 유도하여 고객의 요구사항에 유연하게 대응하도록 한다.

⑤ UP 모형: 반복적이고 점진적인 소프트웨어 개발을 지원하며, 프로그램의 모든 요구사항을 초기에 완전히 파악하도록 요구한다.

정답 5번

출제 포인트

소프트웨어 생명주기 모형과 애자일 개발 방법론의 핵심 특징을 구별하는 문제이다. 특히 반복적·점진적 개발이 요구사항의 초기 완전 확정을 뜻하는지 판단해야 한다.

핵심 개념

나선형 모형은 위험 분석을 중심으로 계획·위험 분석·개발·평가를 반복하는 모형이다.
V-모형은 개발 단계와 이에 대응하는 테스트 단계를 V자 형태로 연결해 검증과 확인을 강조하는 모형이다.
애자일 기법은 문서보다 작동하는 소프트웨어와 변화에 대한 대응을 중시하는 반복적 개발 방식이다.
익스트림 프로그래밍(XP)은 고객 참여, 짧은 반복, 지속적 테스트와 피드백을 강조하는 애자일 방법론이다.
UP(Unified Process)는 유스케이스 중심·아키텍처 중심·반복적·점진적으로 개발하며, 반복 과정에서 요구사항을 구체화하는 프로세스이다.

NoteBase

선지별 해설

① O — 나선형 모형은 Boehm이 제안했으며, 폭포수 모형의 체계성과 프로토타이핑의 반복성을 결합하고 각 반복에서 위험 분석을 수행한다.
② O — V-모형은 폭포수 모형을 확장하여 요구분석·설계·구현의 각 개발 단계에 대응하는 인수·시스템·통합·단위 테스트를 연결한다.
③ O — 애자일 기법은 방대한 설계 문서 작성 자체보다 실제로 작동하는 소프트웨어의 지속적 제공에 더 높은 가치를 둔다. 다만 문서화를 전혀 하지 않는다는 뜻은 아니다.
④ O — 익스트림 프로그래밍은 고객의 지속적인 참여와 피드백, 테스트 중심 개발을 통해 변경되는 요구사항에 유연하게 대응한다.
⑤ X — 틀린 표현은 '프로그램의 모든 요구사항을 초기에 완전히 파악하도록 요구한다'이다. UP는 반복적·점진적으로 개발하면서 각 반복을 통해 요구사항을 발견하고 구체화하므로, 올바르게는 '핵심 요구사항을 바탕으로 시작하되 반복 과정에서 요구사항을 보완하고 상세화한다'이다.

출제자가 노린 함정

⑤ 앞부분의 '반복적이고 점진적인 개발'은 맞지만, 뒤의 '모든 요구사항을 초기에 완전히 파악'은 전통적인 순차 개발식 설명이다. 한 선지 안에서 맞는 특징과 반대되는 특징을 결합한 것이 함정이다.

다시 나오면 이렇게 판단

'반복적·점진적'이라는 표현이 나오면 요구사항과 결과물을 반복마다 보완한다는 뜻으로 판단한다. 따라서 '처음부터 모든 요구사항을 완전히 확정한다'는 절대적 표현이 함께 나오면 서로 모순되는지 확인한다.

암기 포인트

나선형은 위험 분석, V-모형은 개발 단계와 테스트의 대응, 애자일·XP·UP는 반복과 변화 대응이다. UP는 요구사항을 처음에 완성하는 것이 아니라 반복하며 구체화한다.

NoteBase

20. 데이터베이스 관리 시스템(DBMS)에 대한 설명으로 옳지 않은 것은?

① 하나의 데이터베이스 시스템에는 하나의 외부 스키마만 존재하며 하나의 외부 스키마를 여러 개의 응용 프로그램이나 여러 명의 사용자가 공용할 수도 있다.

② 개념 스키마는 개체 간의 관계와 제약 조건을 나타낸다.

③ 내부 스키마는 데이터베이스의 물리적 저장 형태를 기술한 것으로 하나만 존재한다.

④ 외부 스키마는 프로그래머나 사용자가 각각의 입장에서 필요로 하는 데이터베이스의 논리적 구조를 정의한 것이다.

⑤ 개념 스키마는 모든 응용 프로그램이나 사용자들이 필요로 하는 데이터를 종합한 조직 전체의 데이터베이스로 하나만 존재한다.

정답 1번

출제 포인트

이 문제는 DBMS의 3단계 스키마 구조에서 외부 스키마, 개념 스키마, 내부 스키마의 의미와 개수를 구별하는 문제이다.

핵심 개념

외부 스키마: 개별 사용자나 응용 프로그램의 관점에서 필요한 데이터의 논리적 구조를 정의한 사용자 뷰이다. 하나의 데이터베이스에 여러 개가 존재할 수 있다.
개념 스키마: 데이터베이스 전체의 논리적 구조와 개체 간 관계 및 제약 조건을 통합하여 정의한 조직 전체의 관점이다. 하나만 존재한다.
내부 스키마: 데이터의 저장 위치, 레코드 형식, 인덱스 등 물리적 저장 구조를 정의한 저장 관점이다. 하나만 존재한다.
개수의 핵심 대비는 외부 스키마는 여러 개, 개념 스키마와 내부 스키마는 각각 하나이다.

NoteBase

선지별 해설

① X - 틀린 표현은 '하나의 외부 스키마만 존재'이다. 하나의 데이터베이스 시스템에는 사용자나 응용 프로그램의 관점에 따라 여러 외부 스키마가 존재할 수 있으며, 하나의 외부 스키마를 여러 사용자나 응용 프로그램이 공유할 수도 있다.
② O - 개념 스키마는 데이터베이스 전체의 논리적 구조를 표현하므로 개체 간 관계와 무결성 제약 조건을 나타낸다.
③ O - 내부 스키마는 데이터가 저장 장치에 어떻게 배치되고 접근되는지와 같은 물리적 저장 형태를 기술하며 하나만 존재한다.
④ O - 외부 스키마는 개별 프로그래머나 사용자가 자신의 관점에서 필요로 하는 데이터베이스의 논리적 구조를 정의한 것이다.
⑤ O - 개념 스키마는 모든 사용자와 응용 프로그램에 필요한 데이터를 통합한 조직 전체 관점의 논리적 데이터베이스이며 하나만 존재한다.

오답 포인트

①은 '하나의 외부 스키마를 여러 사용자가 공유할 수 있다'는 사실을 '외부 스키마가 하나만 존재한다'는 뜻으로 바꾼 것이 함정이다. 공유 가능성과 전체 외부 스키마의 개수는 별개의 문제이다.

다시 나오면 이렇게 판단

관점의 주체를 먼저 확인한다. 개별 사용자·응용 프로그램의 관점이면 외부 스키마로서 여러 개가 가능하고, 조직 전체의 통합 논리 구조이면 개념 스키마로서 하나이며, 물리적 저장 구조이면 내부 스키마로서 하나이다.

암기 포인트

외부는 사용자별로 여러 개, 개념은 전체 논리 구조 하나, 내부는 물리 저장 구조 하나이다.

NoteBase