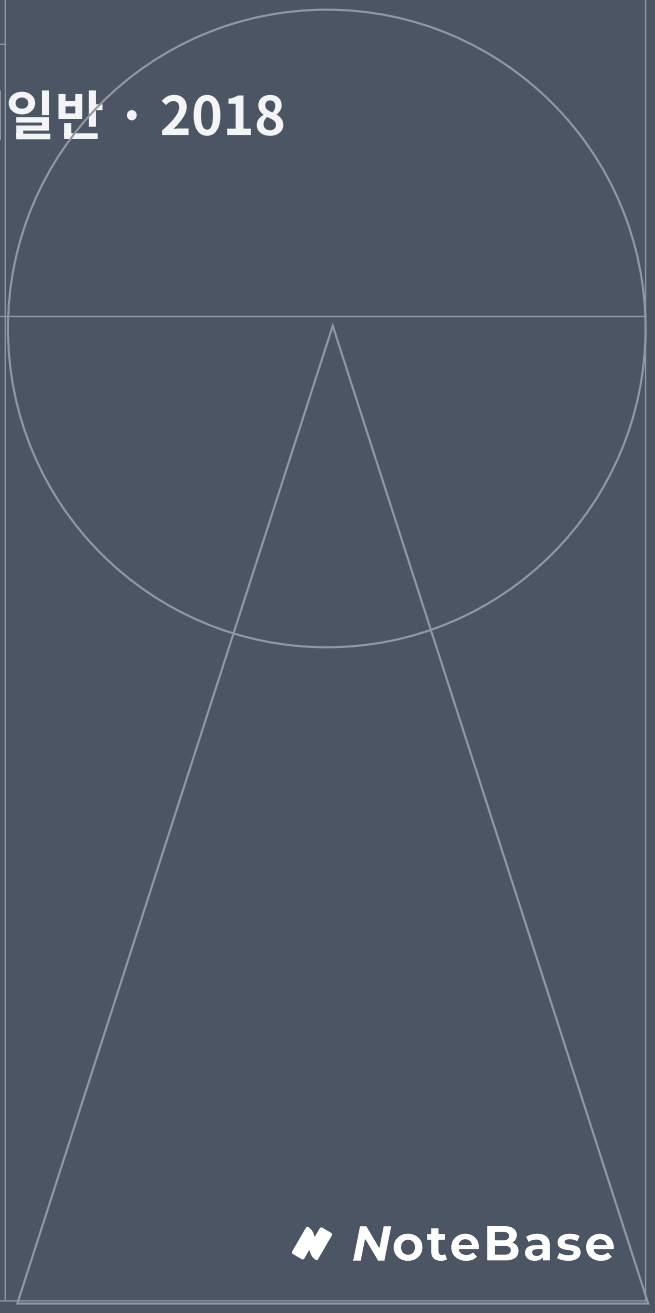
기출OX 해설집

공무원/계리직/9급 · 컴퓨터일반 · 2018



NoteBase

기출OX 해설집

공무원/계리직/9급 · 컴퓨터일반 · 2018

공무원/계리직/9급 · 컴퓨터일반 · 2018

문 1. 다음에서 설명하는 입·출력 장치로 옳은 것은?

> ○ 중앙처리장치로부터 입·출력을 지시받은 후에는 자신의 명령어를 실행시켜 입·출력을 수행하는 독립된 프로세서이다.
> ○ 하나의 명령어에 의해 여러 개의 블록을 입·출력할 수 있다.

① 버스(Bus)
② 채널(Channel)
③ 스풀링(Spooling)
④ DMA(Direct Memory Access)

정답 2번

출제 포인트

CPU의 입·출력 부담을 덜기 위해 독립적으로 입·출력 명령을 수행하는 채널의 특징을 다른 입·출력 관련 개념과 구별하는 문제이다.

핵심 개념

채널(Channel)은 CPU로부터 입·출력 작업을 지시받은 뒤 자체 명령어를 실행하여 입·출력을 처리하는 독립적인 입·출력 전용 프로세서이다. 하나의 입·출력 명령으로 여러 데이터 블록을 처리할 수 있어 CPU의 개입과 부담을 줄인다.
버스는 장치 사이에서 데이터·주소·제어 신호가 이동하는 공통 전송 경로이고, 스풀링은 보조기억장치를 버퍼처럼 사용하여 저속 입·출력 작업을 대기시키는 방식이다. DMA는 CPU를 거치지 않고 주기억장치와 입·출력 장치 사이에서 데이터 블록을 직접 전송하는 방식이다.

NoteBase

선지별 해설

① X — 버스는 CPU, 기억장치, 입·출력 장치 사이에서 데이터·주소·제어 신호를 전달하는 공통 통로이다. 자체 명령어를 실행하는 독립 프로세서라는 표현이 틀렸으며, 올바르게는 제시문의 장치를 채널이라고 해야 한다.
② O — 채널은 CPU의 지시를 받은 뒤 자체 채널 명령어를 실행하여 입·출력을 독립적으로 처리하며, 하나의 명령으로 여러 데이터 블록을 입·출력할 수 있다. 제시문의 두 특징에 모두 맞는다.
③ X — 스풀링은 디스크 같은 보조기억장치에 입·출력할 작업을 임시 저장해 두고 순서대로 처리하는 방식이다. 독립된 입·출력 프로세서라는 표현이 틀렸으며, 올바르게는 제시문의 장치를 채널이라고 해야 한다.
④ X — DMA는 입·출력 장치와 주기억장치 사이의 데이터 블록을 CPU의 지속적인 개입 없이 직접 전송하는 방식이다. 자체 명령어로 여러 입·출력 작업을 통제하는 독립 프로세서라는 표현이 틀렸으며, 올바르게는 제시문의 장치를 채널이라고 해야 한다.

오답 포인트

④ DMA도 CPU의 부담을 줄이므로 채널과 혼동하기 쉽다. 그러나 DMA의 핵심은 '주기억장치와 입·출력 장치 사이의 직접 데이터 전송'이고, 제시문처럼 '자체 명령어 실행'과 '한 명령으로 여러 블록 처리'를 강조하면 채널이다.

다시 나오면 이렇게 판단

'독립된 입·출력 프로세서', '자체 명령어 실행', '한 명령으로 여러 블록 처리'가 나오면 채널로 판단한다. '신호가 오가는 통로'는 버스, '디스크에 작업을 대기'는 스풀링, '주기억장치와 입·출력 장치 간 직접 전송'은 DMA로 구별한다.

암기 포인트

채널은 명령을 실행하고, DMA는 데이터를 직접 전송하며, 버스는 신호가 지나가고, 스풀링은 작업을 대기시킨다.

NoteBase

문 2. 고객계좌 테이블에서 잔고가 100,000원에서 3,000,000원 사이인 고객들의 등급을 '우대고객'으로 변경하고자 <보기>와 같은 SQL문을 작성하였다. ㉠과 ㉡의 내용으로 옳은 것은?

<보기>

```
UPDATE 고객계좌
( ㉠ ) 등급 = '우대고객'
WHERE 잔고 ( ㉡ ) 100000 AND 3000000
```

	㉠	㉡
①	SET	IN
②	SET	BETWEEN
③	VALUES	IN
④	VALUES	BETWEEN

정답 2번

출제 포인트

이 문제는 UPDATE 문에서 변경할 열을 지정하는 SET과 연속된 범위를 조건으로 지정하는 BETWEEN ... AND ...의 쓰임을 묻는다.

핵심 개념

UPDATE 문은 기존 행의 값을 수정하는 명령이며, 기본 형식은 UPDATE 테이블명 SET 열명 = 값 WHERE 조건이다.
BETWEEN 하한값 AND 상한값은 하한값 이상이면서 상한값 이하인 연속 범위를 뜻한다. 따라서 보기의 SQL은 UPDATE 고객계좌 SET 등급 = '우대고객' WHERE 잔고 BETWEEN 100000 AND 3000000으로 완성된다.
IN은 열 값이 괄호 안에 나열된 값 중 하나와 같은지를 검사하므로 연속 범위를 나타내지 않는다.

NoteBase

선지별 해설

① X — ㄱ의 SET은 맞지만 ㄴ의 IN이 틀리다. IN은 IN (100000, 3000000)처럼 값 목록과 함께 사용하며, 이렇게 쓰면 두 끝값만 검사할 뿐 그 사이의 잔고 전체를 선택하지 못한다. 범위를 지정하려면 IN을 BETWEEN으로 고쳐야 한다.
② O — UPDATE 문에서 변경할 열과 값을 지정할 때 SET을 사용하고, 100,000원 이상 3,000,000원 이하의 범위에는 BETWEEN 100000 AND 3000000을 사용한다.
③ X — ㄱ의 VALUES와 ㄴ의 IN이 모두 틀리다. VALUES는 INSERT 문에서 삽입할 값을 지정할 때 주로 사용하며 UPDATE 문에서는 SET으로 고쳐야 한다. 또한 연속 범위 조건은 IN이 아니라 BETWEEN으로 고쳐야 한다.
④ X — ㄴ의 BETWEEN은 맞지만 ㄱ의 VALUES가 틀리다. UPDATE 문에서 수정할 열과 값을 지정하는 표현은 VALUES가 아니라 SET이다. 따라서 SET과 BETWEEN의 조합으로 고쳐야 한다.

오답 포인트

IN과 BETWEEN이 모두 여러 값을 조건으로 다룬다는 점이 함정이다. IN은 나열된 개별 값의 일치 여부를 검사하고, BETWEEN은 두 경계값을 포함한 연속 범위를 검사한다. 또한 VALUES는 값 입력과 관련된 단어이지만 UPDATE 문에서는 변경값 앞에 SET을 사용한다.

다시 나오면 이렇게 판단

기존 데이터 수정이면 먼저 UPDATE ... SET 구조를 확인한다. 조건이 'A에서 B 사이'이면 BETWEEN A AND B를 선택하고, 'A, B, C 중 하나'이면 IN (A, B, C)를 선택한다.

암기 포인트

수정은 UPDATE-SET, 연속 범위는 BETWEEN-AND, 값 목록은 IN이다.

NoteBase

문 3. 네트워크 장치에 대한 설명으로 옳지 않은 것은?

① 허브(Hub)는 여러 대의 단말 장치가 하나의 근거리 통신망(LAN)에 접속할 수 있도록 지원하는 중계 장치이다.

② 리피터(Repeater)는 물리 계층(Physical Layer)에서 동작하며 전송 신호를 재생·중계해 주는 증폭 장치이다.

③ 브리지(Bridge)는 데이터 링크 계층(Data Link Layer)에서 동작하며 같은 MAC 프로토콜(Protocol)을 사용하는 근거리 통신망 사이를 연결하는 통신 장치이다.

④ 게이트웨이(Gateway)는 네트워크 계층(Network Layer)에서 동작하며 동일 전송 프로토콜을 사용하는 분리된 2개 이상의 네트워크를 연결해주는 통신 장치이다.

정답 4번

출제 포인트

허브, 리피터, 브리지, 게이트웨이의 기능과 동작 계층을 구별하는 문제이다. 특히 게이트웨이와 라우터의 역할을 혼동하지 않아야 한다.

핵심 개념

허브: 여러 단말을 하나의 LAN에 연결하는 물리 계층 장치이며, 들어온 신호를 다른 포트로 전달한다.
리피터: 물리 계층에서 약해지거나 왜곡된 신호를 재생·증폭하여 전송 거리를 늘리는 장치이다.
브리지: 데이터 링크 계층에서 MAC 주소를 기준으로 프레임을 전달하며, 같은 종류의 LAN 세그먼트를 연결하는 장치이다.
게이트웨이: 서로 다른 통신 프로토콜이나 구조를 사용하는 네트워크 사이에서 프로토콜을 변환하여 통신하게 하는 장치이다. 네트워크 계층에서 같은 프로토콜의 네트워크를 연결하는 장치는 라우터이다.

선지별 해설

① O — 허브는 여러 단말이 하나의 LAN에 접속하도록 연결하는 다중 포트 중계 장치이므로 옳다.
② O — 리피터는 물리 계층에서 전송 신호를 재생·증폭하여 중계하는 장치이므로 옳다.
③ O — 브리지는 데이터 링크 계층에서 동작하며, 같은 MAC 프로토콜을 사용하는 LAN 사이에서 MAC 주소를 바탕으로 프레임을 전달하므로 옳다.
④ X — 틀린 표현은 '네트워크 계층에서 동작'과 '동일 전송 프로토콜을 사용하는 네트워크를 연결'이다. 이는 라우터에 해당하는 설명이다. 올바르게는 게이트웨이는 서로 다른 프로토콜이나 구조를 사용하는 네트워크 사이에서 프로토콜을 변환하여 연결하는 장치이다.

NoteBase

오답 포인트

④는 '네트워크를 연결한다'는 공통점 때문에 게이트웨이와 라우터를 바꾸어 놓은 선지이다. 네트워크 계층에서 동일 계열 프로토콜의 네트워크를 연결하는 것은 라우터이고, 서로 다른 프로토콜을 변환하며 연결하는 것은 게이트웨이이다.

다시 나오면 이렇게 판단

장치마다 '동작 계층－처리 대상－연결 방식'을 묶어 판단한다. 리피터·허브는 물리 계층의 신호, 브리지는 데이터 링크 계층의 MAC 주소와 프레임, 라우터는 네트워크 계층의 IP 패킷, 게이트웨이는 서로 다른 프로토콜 사이의 변환이 핵심이다.

암기 포인트

신호는 리피터·허브, MAC은 브리지, IP 경로는 라우터, 서로 다른 프로토콜의 변환은 게이트웨이이다.

NoteBase

문 4. ㉠에 들어갈 용어로 옳은 것은?

> (㉠) (은)는 유사한 문제를 해결하기 위해 설계들을 분류하고 각 문제 유형별로 가장 적합한 설계를 일반화하여 체계적으로 정리해 놓은 것으로 소프트웨어 개발에서 효율성과 재사용성을 높일 수 있다.

① 디자인 패턴
② 요구사항 정의서
③ 소프트웨어 개발 생명주기
④ 소프트웨어 프로세스 모델

정답 1번

출제 포인트

유사한 문제를 해결한 설계를 유형별로 일반화하여 재사용하는 소프트웨어 설계 개념이 무엇인지 묻는 문제이다.

핵심 개념

디자인 패턴은 소프트웨어 설계에서 반복적으로 나타나는 문제에 적용할 수 있도록 검증된 설계 해법의 구조를 일반화한 것이다. 완성된 코드를 그대로 복사하는 것이 아니라 문제 상황, 구성 요소와 그 관계, 해결 방식의 틀을 재사용한다.
요구사항 정의서는 시스템이 무엇을 해야 하는지를 정리한 문서이고, 소프트웨어 개발 생명주기는 개발의 전체 단계를 나타내며, 소프트웨어 프로세스 모델은 그 개발 절차를 구조화한 모형이다.

선지별 해설

① O — 디자인 패턴은 반복되는 설계 문제와 그에 적합한 해결 구조를 유형별로 일반화해 정리한 것이므로, 제시문의 효율성과 재사용성을 높이는 개념에 해당한다.
② X — 틀린 표현은 '요구사항 정의서'이다. 요구사항 정의서는 소프트웨어가 제공해야 할 기능과 성능·제약 조건 등을 명세한 문서이다. 제시문의 설명은 올바르게는 '디자인 패턴'이다.
③ X — 틀린 표현은 '소프트웨어 개발 생명주기'이다. 소프트웨어 개발 생명주기는 계획, 요구사항 분석, 설계, 구현, 시험, 운영·유지보수 등 개발 전 과정을 단계로 나타낸 개념이다. 제시문의 설명은 올바르게는 '디자인 패턴'이다.
④ X — 틀린 표현은 '소프트웨어 프로세스 모델'이다. 소프트웨어 프로세스 모델은 개발 활동과 진행 순서를 체계화한 모형으로, 폭포수 모델 등이 이에 해당한다. 제시문의 설명은 올바르게는 '디자인 패턴'이다.

NoteBase

오답 포인트

제시문에 '문제 유형', '체계적으로 정리', '소프트웨어 개발'이 함께 나와 개발 단계나 프로세스 모델로 오인하기 쉽다. 그러나 정답을 가르는 표현은 '유사한 문제', '가장 적합한 설계', '일반화', '재사용성'이며, 이는 반복되는 설계 해법의 틀인 디자인 패턴을 가리킨다.

다시 나오면 이렇게 판단

반복되는 설계 문제의 해결책을 일반화하여 재사용한다는 설명이면 디자인 패턴으로 판단한다. 무엇을 만들지 명세하면 요구사항 정의서, 개발의 전체 단계를 말하면 개발 생명주기, 개발 활동의 진행 방식을 모형화하면 프로세스 모델이다.

암기 포인트

반복되는 설계 문제의 재사용 가능한 해법 틀은 디자인 패턴이다.

NoteBase

문 5. 결합도(Coupling)는 모듈 간의 상호 의존 정도 또는 모듈 간의 연관 관계를 의미한다. 아래에 나타낸 결합도를 약한 정도에서 강한 정도 순으로 올바르게 나열한 것은?

ㄱ. 내용 결합도(Content Coupling)
ㄴ. 제어 결합도(Control Coupling)
ㄷ. 자료 결합도(Data Coupling)
ㄹ. 공통 결합도(Common Coupling)

① ㄷ – ㄴ – ㄹ – ㄱ
② ㄷ – ㄹ – ㄱ – ㄴ
③ ㄹ – ㄴ – ㄷ – ㄱ
④ ㄹ – ㄷ – ㄱ – ㄴ

정답 1번

출제 포인트

모듈 사이의 의존성을 나타내는 결합도의 유형을 약한 것부터 강한 것까지 배열할 수 있는지를 묻는 문제이다.

핵심 개념

자료 결합도: 모듈 사이에 필요한 자료만 매개변수로 전달하는 결합으로, 제시된 유형 중 가장 약하다.
제어 결합도: 한 모듈이 다른 모듈의 처리 흐름을 결정하는 제어 정보나 플래그를 전달하는 결합이다.
공통 결합도: 여러 모듈이 전역 변수와 같은 공통 자료 영역을 함께 사용하는 결합이다.
내용 결합도: 한 모듈이 다른 모듈의 내부 자료나 코드를 직접 참조하거나 변경하는 결합으로, 제시된 유형 중 가장 강하다.
따라서 약한 순서에서 강한 순서는 자료 결합도(ㄷ) → 제어 결합도(ㄴ) → 공통 결합도(ㄹ) → 내용 결합도(ㄱ)이다.

NoteBase

선지별 해설

① O — ㄷ(자료) → ㄴ(제어) → ㄹ(공통) → ㄱ(내용)은 결합도가 약한 것에서 강한 것으로 커지는 올바른 순서이다.
② X — ㄷ → ㄹ → ㄱ → ㄴ은 제어 결합도(ㄴ)를 내용 결합도(ㄱ)보다 강하게 둔 부분이 틀리다. 올바르게는 ㄷ → ㄴ → ㄹ → ㄱ이다.
③ X — ㄹ → ㄴ → ㄷ → ㄱ은 공통 결합도(ㄹ)를 자료 결합도(ㄷ)보다 약하게 둔 부분이 틀리다. 올바르게는 ㄷ → ㄴ → ㄹ → ㄱ이다.
④ X — ㄹ → ㄷ → ㄱ → ㄴ은 공통 결합도(ㄹ)를 자료 결합도(ㄷ)보다 약하게 두고, 내용 결합도(ㄱ)를 제어 결합도(ㄴ)보다 약하게 둔 부분이 틀리다. 올바르게는 ㄷ → ㄴ → ㄹ → ㄱ이다.

오답 포인트

결합도의 종류를 아는 것과 배열 방향을 확인하는 것이 모두 필요하다. 이 문제는 강한 순서가 아니라 약한 순서에서 강한 순서를 요구하므로, 내부 의존이 가장 적은 자료 결합도에서 시작하여 다른 모듈의 내부를 직접 건드리는 내용 결합도로 끝나야 한다.

다시 나오면 이렇게 판단

모듈이 단순한 자료만 전달하면 결합도가 약하고, 상대 모듈의 제어 흐름에 관여하거나 공통 영역을 공유하며, 나아가 상대 모듈의 내부 내용에 직접 접근할수록 결합도가 강하다고 판단한다.

암기 포인트

약한 순서부터 '자료 → 제어 → 공통 → 내용'이다.

NoteBase

문 6. 컴퓨터 알고리즘에 대한 설명으로 옳지 않은 것을 <보기>에서 모두 고른 것은?

<보기>

ㄱ. 힙 정렬(Heap Sort) 알고리즘의 시간 복잡도는 $O(n^2)$이다.
ㄴ. 0/1 배낭(0/1 Knapsack) 문제에 대하여 다항시간(Polynomial time) 내에 해결 가능한 알고리즘이 개발되었다.
ㄷ. 모든 NP(Non-deterministic Polynomial time) 문제는 컴퓨터를 이용하여 다항시간에 해결할 수 없다.

① ㄱ
② ㄱ, ㄴ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

정답 4번

출제 포인트

힙 정렬의 시간 복잡도, 0/1 배낭 문제의 계산 복잡도, P와 NP의 관계를 구분하여 옳지 않은 설명을 모두 찾는 문제이다.

핵심 개념

힙 정렬: 힙 자료구조를 이용하는 비교 정렬로, 최악의 경우에도 시간 복잡도가 $O(n \log n)$이다.
0/1 배낭 문제: 각 물건을 넣거나 넣지 않는 조건에서 가치를 최대화하는 문제이다. 일반적인 정확 해법은 다항시간 알고리즘이 알려져 있지 않으며, 용량을 W라 할 때 동적 계획법의 $O(nW)$는 입력값의 크기에 대한 의사 다항시간이다.
NP: 주어진 해답을 다항시간에 검증할 수 있는 결정 문제들의 집합이다. P는 NP에 포함되므로 NP 문제 중에는 다항시간에 풀 수 있는 문제도 있다.

NoteBase

선지별 해설

① X
ㄱ만 옳지 않은 것이 아니다. ㄱ의 $O(n^2)$는 틀리고 올바르게는 힙 정렬의 시간 복잡도가 $O(n \log n)$이며, ㄴ과 ㄷ도 틀리므로 옳지 않은 것은 ㄱ·ㄴ·ㄷ 모두이다.

② X
ㄱ과 ㄴ만 옳지 않은 것이 아니다. ㄴ의 '다항시간 내에 해결 가능한 알고리즘이 개발되었다'는 틀리고, 일반적인 0/1 배낭 문제의 정확한 다항시간 알고리즘은 알려져 있지 않다. ㄷ도 틀리므로 올바른 선택지 구성은 ㄱ·ㄴ·ㄷ이다.

③ X
ㄴ과 ㄷ만 옳지 않은 것이 아니다. ㄷ의 '모든 NP 문제는 다항시간에 해결할 수 없다'는 틀리고, P에 속하는 문제는 NP에도 속하면서 다항시간에 해결할 수 있다. ㄱ도 틀리므로 올바른 선택지 구성은 ㄱ·ㄴ·ㄷ이다.

④ O
ㄱ은 힙 정렬의 복잡도를 잘못 제시했고, ㄴ은 0/1 배낭 문제에 일반적인 정확 다항시간 해법이 있는 것처럼 서술했으며, ㄷ은 P에 속하는 NP 문제까지 다항시간에 풀 수 없다고 단정했다. 따라서 ㄱ·ㄴ·ㄷ이 모두 옳지 않다.

오답 포인트

ㄴ에서는 $O(nW)$ 동적 계획법을 다항시간 알고리즘으로 오해하기 쉽다. W의 값 자체에 비례하는 시간은 입력 길이에 대한 다항시간이 아니라 의사 다항시간이다. ㄷ에서는 'NP'를 '다항시간에 풀 수 없는 문제'로 정의하는 오류가 함정이다.

다시 나오면 이렇게 판단

정렬 문제는 대표 시간 복잡도를 확인하고, 최적화 문제는 의사 다항시간과 진정한 다항시간을 구분한다. NP 관련 선지는 '검증이 다항시간'이라는 정의와 $P \subseteq NP$의 관계를 기준으로 판단한다.

암기 포인트

힙 정렬은 $O(n \log n)$, 0/1 배낭의 $O(nW)$는 의사 다항시간, P는 NP의 부분집합이다.

NoteBase

문 7. JAVA 프로그램의 실행 결과로 옳은 것은?

```
class Test {
    public static void main(String[] args) {
        int a = 101;
        System.out.println((a>>2) << 3);
    }
}
```

① 0 ② 200 ③ 404 ④ 600

정답 2번

출제 포인트

Java의 비트 시프트 연산자 >>와 <<의 의미를 알고, 괄호에 따라 연산한 결과를 구하는 문제이다.

조건 정리

주어진 값은 int형 양의 정수 a = 101이다.
출력할 식은 (a >> 2) << 3이므로, a를 오른쪽으로 2비트 이동한 뒤 그 결과를 왼쪽으로 3비트 이동한다.
양의 정수를 오른쪽으로 n비트 이동하면 몫 기준으로 2^n으로 나눈 것과 같고, 왼쪽으로 n비트 이동하면 오버플로가 없는 범위에서 2^n을 곱한 것과 같다.

사용할 공식

오른쪽 시프트: x >> n = x/2^n
왼쪽 시프트: x << n = x × 2^n
이 문제의 a는 양수이고 계산 결과가 int 범위를 넘지 않으므로 위 관계를 그대로 적용한다.

계산 과정

1. 101 >> 2 = 101/2^2 = 25.25 = 25이다.
2. 25 << 3 = 25 × 2^3 = 25 × 8 = 200이다.
3. 따라서 System.out.println이 출력하는 값은 200이다.

NoteBase

선지 판단

① X — 0은 오른쪽 시프트에서 모든 유효 비트가 사라진다고 잘못 판단한 값이다. 101을 2비트 오른쪽으로 이동한 값은 25이고 최종 출력값은 200이므로, 0은 200으로 고쳐야 한다.
② O — (101 >> 2)의 값 25를 다시 3비트 왼쪽으로 이동하면 25 × 8 = 200이 되므로 옳다.
③ X — 404는 원래 값 101을 2비트 왼쪽으로 이동하여 101×2^2로 잘못 계산한 값이다. 실제 식은 오른쪽으로 2비트 이동한 뒤 왼쪽으로 3비트 이동하므로, 404는 200으로 고쳐야 한다.
④ X — 600이 되려면 왼쪽 시프트 직전 값이 75여야 하지만, 실제 중간값은 25이다. 따라서 600은 200으로 고쳐야 한다.

오답 포인트

괄호 안의 a >> 2를 생략하거나, 시프트 수 2와 3을 단순히 상쇄하는 것이 함정이다. 이 식에서는 오른쪽 시프트 과정에서 나머지가 버려지므로 각 연산을 순서대로 계산해야 한다.

다시 나오면 이렇게 판단

시프트 연산식은 괄호와 연산 순서를 먼저 확인하고, 양수에 대해 >> n은 2^n으로 나눈 몫, << n은 2^n을 곱한 값으로 바꾸어 단계별로 계산한다.

암기 포인트

>> n은 오른쪽으로 n비트 이동하여 2^n으로 나눈 몫을 만들고, << n은 왼쪽으로 n비트 이동하여 2^n을 곱한 값을 만든다.

NoteBase

문 8. 암호 방식에 대한 설명으로 옳은 것을 <보기>에서 모두 고른 것은?

<보기>

ㄱ. 대칭키 암호 방식(Symmetric Key Cryptosystem)은 암호화 키와 복호화 키가 동일하다.
ㄴ. 공개키 암호 방식(Public Key Cryptosystem)은 사용자 수가 증가하면 관리해야 할 키의 수가 증가하여 키 변화의 빈도가 높다.
ㄷ. 대칭키 암호 방식은 공개키 암호 방식에 비하여 암호화 속도가 빠르다.
ㄹ. 공개키 암호 방식은 송신자와 발신자가 서로 같은 키를 사용하여 통신을 수행한다.

① ㄱ, ㄴ ② ㄱ, ㄷ ③ ㄴ, ㄷ ④ ㄴ, ㄹ

정답 2번

출제 포인트

대칭키 암호 방식과 공개키 암호 방식의 키 사용 구조, 키 관리 특성, 암호화 속도를 구분하는 문제이다.

핵심 개념

대칭키 암호 방식: 암호화와 복호화에 동일한 비밀키를 사용하는 방식이다. 처리 속도가 빠르지만, 통신 상대별로 비밀키를 안전하게 공유하고 관리해야 한다.
공개키 암호 방식: 서로 연관된 공개키와 개인키의 쌍을 사용하는 방식이다. 일반적으로 수신자의 공개키로 암호화하고 수신자의 개인키로 복호화하며, 대칭키 방식보다 처리 속도는 느리지만 사용자 수가 늘어날 때 키 배포와 관리에 유리하다.
보기 판단: ㄱ과 ㄷ은 옳고, ㄴ과 ㄹ은 틀리다.

NoteBase

선지별 해설

① X — ㄱ은 옳지만 ㄴ은 틀리므로 'ㄱ, ㄴ'은 옳은 보기의 조합이 아니다. ㄴ의 '관리해야 할 키의 수가 증가하여 키 변화의 빈도가 높다'는 설명은 공개키 방식의 장점과 반대이다. 올바른 조합은 'ㄱ, ㄷ'이다.
② O — ㄱ은 대칭키 방식이 암호화와 복호화에 같은 키를 사용한다는 설명이고, ㄷ은 대칭키 방식이 공개키 방식보다 일반적으로 암호화 속도가 빠르다는 설명이므로 모두 옳다.
③ X — ㄷ은 옳지만 ㄴ은 틀리므로 'ㄴ, ㄷ'은 옳은 보기의 조합이 아니다. 공개키 방식은 각 사용자가 공개키와 개인키의 쌍을 가지므로, 사용자 수가 늘어날 때 통신 상대마다 별도의 비밀키를 두는 대칭키 방식보다 키 관리에 유리하다. 올바른 조합은 'ㄱ, ㄷ'이다.
④ X — ㄴ과 ㄹ은 모두 틀리므로 'ㄴ, ㄹ'은 옳은 보기의 조합이 아니다. ㄹ의 '서로 같은 키'가 틀린 표현이며, 올바르게는 공개키 암호 방식에서 공개키와 개인키라는 서로 다른 키를 사용한다. 올바른 조합은 'ㄱ, ㄷ'이다.

오답 포인트

ㄴ은 공개키 방식에서도 사용자마다 키 쌍이 필요하다는 사실을 '키 관리가 더 불리하다'는 뜻으로 잘못 연결한 함정이다. ㄹ은 대칭키 방식의 '같은 키 사용' 특성을 공개키 방식에 뒤바꿔 놓았다.

다시 나오면 이렇게 판단

먼저 암호화 키와 복호화 키가 같은지 다른지를 확인한다. 같은 키이면 대칭키 방식, 공개키와 개인키처럼 다른 키의 쌍을 쓰면 공개키 방식이다. 그다음 속도는 대칭키 방식이 빠르고, 다수 사용자 환경의 키 배포·관리는 공개키 방식이 유리하다고 판단한다.

암기 포인트

대칭키는 같은 키·빠른 속도, 공개키는 서로 다른 키의 쌍·유리한 키 관리이다.

NoteBase

문 9. 학생 테이블에 튜플들이 아래와 같이 저장되어 있을 때, <NULL, '김영희', '서울'> 튜플을 삽입하고자 한다. 해당 연산에 대한 [결과]와 [원인]으로 옳은 것은?(단, 학생 테이블의 기본 키는 학번이다.)

학번	이름	주소
1	김철희	경기
2	이철수	천안
3	박민수	제주

	[결과]		[원인]
①	삽입 가능	–	무결성 제약조건 만족
②	삽입 불가	–	관계 무결성 위반
③	삽입 불가	–	개체 무결성 위반
④	삽입 불가	–	참조 무결성 위반

정답 3번

출제 포인트

이 문제는 기본 키가 설정된 테이블에 새 튜플을 삽입할 때 개체 무결성 제약조건을 적용할 수 있는지를 묻는 문제이다.

사례 정리

핵심 사실 1: 학생 테이블의 기본 키는 학번이다.
핵심 사실 2: 삽입하려는 튜플은 <NULL, '김영희', '서울'>이다.
판단에 중요한 조건: 기본 키를 구성하는 속성에는 NULL을 넣을 수 없다.

핵심 개념

개체 무결성은 각 튜플을 유일하게 식별하는 기본 키의 값이 NULL이 될 수 없도록 하는 제약조건이다.
참조 무결성은 외래 키 값이 참조 대상 테이블의 기본 키 값과 일치하거나 NULL이어야 한다는 제약조건이다.

적용 기준

새 튜플의 학번은 NULL이고 학번은 기본 키이므로 개체 무결성을 위반한다. 따라서 해당 튜플은 삽입할 수 없다. 이름이나 주소 값은 이 결론을 바꾸지 않는다.

NoteBase

선지별 해설

① X — 기본 키인 학번에 NULL이 들어가므로 무결성 제약조건을 만족하지 않는다. 틀린 표현 '삽입 가능 - 무결성 제약조건 만족'은 '삽입 불가 - 개체 무결성 위반'으로 고쳐야 한다.
② X — 삽입이 불가능하다는 결과는 맞지만, 위반한 제약조건은 관계 무결성이 아니라 개체 무결성이다. 틀린 표현 '관계 무결성 위반'은 '개체 무결성 위반'으로 고쳐야 한다.
③ O — 기본 키인 학번이 NULL이므로 개체 무결성을 위반하며, 해당 튜플은 삽입할 수 없다.
④ X — 문제의 튜플에는 다른 테이블을 참조하는 외래 키가 제시되지 않았으므로 참조 무결성 위반으로 판단할 수 없다. 틀린 표현 '참조 무결성 위반'은 '개체 무결성 위반'으로 고쳐야 한다.

오답 포인트

②와 ④도 '삽입 불가'라고 되어 있어 결과만 보면 그럴듯하지만, 원인까지 일치해야 한다. 이 문제의 직접적인 단서는 '기본 키는 학번'이라는 조건과 삽입 학번이 NULL이라는 점이다.

다시 나오면 이렇게 판단

먼저 문제가 된 속성이 기본 키인지 외래 키인지 확인한다. 기본 키가 NULL이면 개체 무결성 위반이고, 외래 키가 존재하지 않는 참조 대상 값을 가리키면 참조 무결성 위반이다.

암기 포인트

기본 키의 NULL은 개체 무결성 위반이다.

NoteBase

문 10. 10진수 -2.75를 아래와 같이 IEEE 754 표준에 따른 32비트 단정도 부동소수점(Single Precision Floating Point) 표현 방식에 따라 2진수로 표기했을 때 옳은 것은?

부호	지수부	가수부

(부호: 1비트, 지수부: 8비트, 가수부: 23비트)

① 1000 0000 0000 0000 0000 0000 0000 1011

② 1000 0000 1011 0000 0000 0000 0000 0000

③ 1010 0000 0110 0000 0000 0000 0000 0000

④ 1100 0000 0011 0000 0000 0000 0000 0000

정답 4번

출제 포인트

이 문제는 음의 10진수를 2진수로 바꾸고 정규화한 뒤, IEEE 754 단정도 형식의 부호부·지수부·가수부를 구성하는 문제이다.

조건 정리

주어진 값: -2.75
구해야 할 값: IEEE 754 단정도 32비트 표현
비트 구성: 부호부 1비트, 지수부 8비트, 가수부 23비트
주의할 조건: 지수부에는 실제 지수에 바이어스 127을 더하고, 가수부에는 정규화된 수의 맨 앞 1을 제외한 소수 부분만 기록한다.

사용할 공식

정규화된 IEEE 754 단정도 수는 다음과 같이 해석한다.
$(-1)^s \times (1.f)_2 \times 2^{E-127}$
여기서 s는 부호 비트, E는 지수부를 부호 없는 2진수로 읽은 값, f는 가수부이다. -2.75는 0이 아닌 정규화 가능한 수이므로 이 형식을 사용한다.

NoteBase

계산 과정

1. $2.75_{10} = 10.11_2$이므로 $-2.75_{10} = -10.11_2$이다.
2. 정규화하면 $-10.11_2 = -1.011_2 \times 2^1$이다.
3. 음수이므로 부호부는 1이다.
4. 실제 지수는 1이므로 바이어스를 더한 지수는 $1 + 127 = 128 = 10000000_2$이다.
5. 정규화된 1.011_2에서 맨 앞의 1은 저장하지 않으므로 가수부는 01100000000000000000000이다.
6. 세 부분을 이어 붙이면 11000000001100000000000000000000, 즉 1100 0000 0011 0000 0000 0000 0000 0000이다.

선지 판단

① X — 1 | 00000000 | 00000000000000000001011로 나뉘어 지수부와 가수부가 모두 계산 결과와 다르다. 틀린 표현은 지수부 00000000과 끝부분의 가수 1011이며, 올바르게는 1 | 10000000 | 01100000000000000000000이다.
② X — 가수부 011000…은 맞지만 지수부가 00000001이다. 실제 지수 1에는 바이어스 127을 더해야 하므로 올바른 지수부는 10000000이고, 전체 표현은 1 | 10000000 | 01100000000000000000000이다.
③ X — 지수부가 01000000이고 가수부가 110000…이어서 정규화 결과와 일치하지 않는다. 올바르게는 실제 지수 1을 바이어스 처리한 지수부 10000000과, 선행 1을 뺀 가수부 011000…을 써야 한다.
④ O — 1 | 10000000 | 01100000000000000000000으로 나뉘며, 부호·바이어스 지수·가수가 모두 -2.75의 계산 결과와 일치한다.

오답 포인트

핵심 함정은 실제 지수 1을 그대로 저장하거나, 정규화된 수의 선행 1까지 가수부에 적는 것이다. ②는 가수부 모양 때문에 정답처럼 보이지만 지수부에 바이어스 127을 반영하지 않아 틀린다.

다시 나오면 이렇게 판단

10진수를 2진수로 변환한 뒤 $1.x \times 2^e$ 꼴로 정규화하고, 부호 비트 결정 → $e + 127$을 8비트로 변환 → 선행 1을 제외한 가수 23비트 작성 순서로 판단한다. 마지막에는 1비트+8비트+23비트가 총 32비트인지 확인한다.

암기 포인트

단정도는 부호 1비트·지수 8비트·가수 23비트이며, 지수 바이어스는 127이고 정규화 수의 선행 1은 저장하지 않는다.

NoteBase

문 11. ㉠에 들어갈 용어로 옳은 것은?

> 주기억장치의 물리적 크기의 한계를 해결하기 위한 기법으로 주기억장치의 크기에 상관없이 프로그램이 메모리의 주소를 논리적인 관점에서 참조할 수 있도록 하는 것을 (㉠)라고 한다.

① 레지스터(Register)
② 정적 메모리(Static Memory)
③ 가상 메모리(Virtual Memory)
④ 플래시 메모리(Flash Memory)

정답 3번

출제 포인트

물리적 주기억장치의 용량 한계를 보완하고 프로그램에 논리적 주소 공간을 제공하는 가상 메모리의 정의를 묻는 문제이다.

핵심 개념

가상 메모리는 보조기억장치의 일부를 주기억장치처럼 활용하여, 프로그램이 실제 주기억장치의 물리적 크기와 독립된 더 큰 논리 주소 공간을 사용할 수 있게 하는 기법이다.
레지스터는 CPU 내부의 소량·고속 기억장치이고, 정적 메모리는 전원이 공급되는 동안 저장 상태를 유지하는 반도체 메모리이며, 플래시 메모리는 전원이 꺼져도 데이터가 보존되는 비휘발성 반도체 메모리이다.

선지별 해설

① X — 레지스터는 CPU 내부에서 연산에 필요한 데이터나 주소를 일시적으로 저장하는 소량·고속 기억장치이다. 물리적 주기억장치의 용량 한계를 해결하는 기법이 아니므로, '레지스터'가 아니라 '가상 메모리'로 고쳐야 한다.
② X — 정적 메모리(SRAM)는 전원이 공급되는 동안 데이터를 유지하며 주로 캐시 메모리에 사용되는 반도체 메모리이다. 논리 주소 공간을 물리 메모리와 분리하는 기법이 아니므로, '정적 메모리'가 아니라 '가상 메모리'로 고쳐야 한다.
③ O — 가상 메모리는 보조기억장치를 이용하여 프로그램이 물리적 주기억장치의 크기와 무관한 논리 주소 공간을 참조할 수 있게 하므로 제시문의 설명과 일치한다.
④ X — 플래시 메모리는 전기적으로 데이터를 기록·삭제하며 전원이 꺼져도 내용을 유지하는 비휘발성 기억장치이다. 주기억장치의 물리적 용량 한계를 해결하는 주소 공간 관리 기법이 아니므로, '플래시 메모리'가 아니라 '가상 메모리'로 고쳐야 한다.

NoteBase

오답 포인트

제시문에 '메모리'가 반복되어 기억장치의 종류를 고르기 쉽지만, 결정적 표현은 '물리적 크기의 한계 해결'과 '논리적인 관점에서 주소 참조'이다. 이는 특정 저장장치의 명칭이 아니라 가상 메모리라는 메모리 관리 기법을 뜻한다.

다시 나오면 이렇게 판단

물리 메모리보다 큰 주소 공간, 논리 주소와 물리 주소의 분리, 보조기억장치의 활용이 제시되면 가상 메모리로 판단한다. CPU 내부의 초고속 임시 저장은 레지스터, 캐시에 주로 쓰이는 휘발성 반도체 메모리는 정적 메모리, 비휘발성 저장은 플래시 메모리로 구분한다.

암기 포인트

가상 메모리는 '작은 물리 메모리로 큰 논리 주소 공간을 제공하는 기법'이다.

NoteBase

문 12. C 프로그램의 실행 결과로 옳은 것은?

```
#include<stdio.h>
int main( )
{
    int i, sum=0;
    for(i=1; i<=10; i+=2) {
        if(i%2 && i%3) continue;
        sum += i;
    }
    printf("%d\n", sum);
    return 0;
}
```

① 6 ② 12 ③ 25 ④ 55

정답 2번

출제 포인트

이 문제는 for문의 반복값과 continue문의 실행 조건을 차례로 추적하여 최종 합계를 구하는 문제이다.

조건 정리

반복값: $i = 1, 3, 5, 7, 9$
조건식: i%2와 i%3이 모두 0이 아닐 때 continue가 실행된다.
주의할 조건: C에서 0은 거짓, 0이 아닌 값은 참이며, continue가 실행되면 그 반복의 sum += i는 건너뛴다.

사용할 원리

논리곱 &&는 두 피연산자가 모두 참일 때 참이다. 이 반복문에서 i는 항상 홀수이므로 i%2는 항상 참이다. 따라서 i%3도 참인 1, 5, 7에서는 continue가 실행되고, 3의 배수인 3과 9에서는 조건이 거짓이 되어 sum에 더해진다.

계산 과정

1. $i = 1$: i%2와 i%3이 모두 참이므로 continue가 실행된다.
2. $i = 3$: i%3이 0이므로 조건이 거짓이며 sum은 3이 된다.
3. $i = 5, 7$: 조건이 참이므로 각각 continue가 실행된다.
4. $i = 9$: i%3이 0이므로 sum에 9를 더한다.
5. 최종값은 $sum = 3 + 9 = 12$이고 printf는 12를 출력한다.

NoteBase

선지 판단

① X — 실행 결과를 6으로 보는 것은 반복 추적 결과와 맞지 않는다. 합산되는 값은 3과 9이므로 올바르게는 12이다.
② O — continue가 실행되지 않는 3과 9만 더하면 12이므로 실행 결과와 일치한다.
③ X — 25는 1+3+5+7+9로, continue를 무시하고 모든 반복값을 더한 결과이다. 올바르게는 continue로 1, 5, 7을 제외하여 12이다.
④ X — 55는 1부터 10까지를 모두 더한 값으로, i+=2와 continue를 모두 반영하지 않은 결과이다. 올바르게는 반복값 1, 3, 5, 7, 9 중 3과 9만 더한 12이다.

오답 포인트

핵심 함정은 i%2 && i%3을 '2와 3으로 모두 나누어떨어질 때'로 반대로 해석하는 것이다. 나머지가 0이면 거짓이므로, 이 조건은 2로도 3으로도 나누어떨어지지 않을 때 참이 되어 continue를 실행한다. 또한 continue 뒤의 sum += i는 실행되지 않는다.

다시 나오면 이렇게 판단

반복문의 i값을 먼저 나열하고, 각 값에서 조건식의 참·거짓, continue 실행 여부, 누적값을 차례로 추적한다. 나머지 연산을 논리식에 직접 쓰면 0은 거짓이고 0이 아닌 나머지는 참으로 판단한다.

암기 포인트

continue는 현재 반복의 남은 문장을 건너뛴다. C의 조건식에서는 0은 거짓, 0이 아니면 참이다.

NoteBase

문 13. 다음에서 설명하는 소프트웨어 개발 방법론으로 옳은 것은?

> 프로세스와 도구 중심이 아닌 개발 과정의 소통을 중요하게 생각하는 소프트웨어 개발 방법론으로 반복적인 개발을 통한 잦은 출시를 목표로 한다.

① 애자일 개발 방법론
② 구조적 개발 방법론
③ 객체지향 개발 방법론
④ 컴포넌트 기반 개발 방법론

정답 1번

출제 포인트

소프트웨어 개발 방법론을 구분하는 핵심 특징을 묻는 문제이다. 제시문의 '개발 과정의 소통', '반복적인 개발', '작은 출시'가 애자일 개발 방법론의 단서이다.

핵심 개념

애자일 개발 방법론은 고객 및 개발 구성원과의 소통을 중시하고, 짧은 주기로 개발과 피드백을 반복하여 동작하는 소프트웨어를 점진적으로 출시하는 방법론이다.
구조적 개발 방법론은 기능과 처리 절차를 중심으로 시스템을 단계적으로 분석·설계하는 방법론이다.
객체지향 개발 방법론은 데이터와 기능을 객체로 묶고 캡슐화·상속·다형성 등을 활용하는 방법론이다.
컴포넌트 기반 개발 방법론은 독립적으로 사용할 수 있는 소프트웨어 컴포넌트를 조립·재사용하여 시스템을 개발하는 방법론이다.

NoteBase

선지별 해설

① O — 애자일 개발 방법론은 프로세스와 도구 자체보다 사람 사이의 소통과 협력을 중시하며, 반복적·점진적 개발을 통해 작은 단위의 결과물을 자주 출시한다. 제시문의 설명과 일치한다.
② X — 틀린 부분은 제시문의 특징을 '구조적 개발 방법론'으로 본 것이다. 올바르게는 기능과 처리 절차를 중심으로 단계적인 분석과 설계를 수행하는 방법론이 구조적 개발 방법론이다. 제시문은 애자일 개발 방법론에 해당한다.
③ X — 틀린 부분은 제시문의 특징을 '객체지향 개발 방법론'으로 본 것이다. 올바르게는 객체를 중심으로 시스템을 구성하고 캡슐화·상속·다형성을 활용하는 방법론이 객체지향 개발 방법론이다. 제시문은 애자일 개발 방법론에 해당한다.
④ X — 틀린 부분은 제시문의 특징을 '컴포넌트 기반 개발 방법론'으로 본 것이다. 올바르게는 독립적인 소프트웨어 컴포넌트를 조립하고 재사용하여 개발하는 방법론이 컴포넌트 기반 개발 방법론이다. 제시문은 애자일 개발 방법론에 해당한다.

오답 포인트

'반복적 개발'만 보고 객체지향의 객체 반복 사용이나 컴포넌트의 재사용과 혼동하기 쉽다. 그러나 제시문의 결정적 표현은 소통을 중시하고 작은 결과물을 반복적으로 출시한다는 점이며, 이는 애자일의 특징이다.

다시 나오면 이렇게 판단

소통·협력, 변화에 대한 신속한 대응, 짧은 개발 주기, 반복적·점진적 개발, 작은 단위의 빈번한 출시가 함께 제시되면 애자일 개발 방법론으로 판단한다. 객체·클래스가 중심이면 객체지향, 재사용 가능한 부품의 조립이 중심이면 컴포넌트 기반, 기능과 처리 절차의 단계적 분석이 중심이면 구조적 개발 방법론으로 구분한다.

암기 포인트

애자일은 소통하며 짧게 반복하고, 작은 결과물을 자주 출시하는 방법론이다.

NoteBase

문 14. 불 대수(Boolean Algebra)에 대한 최소화로 옳지 않은 것은?

① $A(A+B) = A$

② $A+\overline{A}B = A+B$

③ $A(\overline{A}+B) = AB$

④ $AB+A\overline{B}+\overline{A}B = A$

정답 4번

출제 포인트

불 대수의 분배법칙, 보수법칙, 항등법칙, 흡수법칙을 이용해 각 식이 올바르게 최소화되었는지 판단하는 문제이다.

핵심 개념

불 대수에서는 $A + A = A$, $A\overline{A} = 0$, $A + \overline{A} = 1$이 성립한다. 흡수법칙 $A + AB = A$와 변형된 흡수법칙 $A + \overline{A}B = A + B$를 함께 사용하면 식을 간단히 줄일 수 있다.

선지별 해설

① O — $A(A + B) = AA + AB = A + AB = A$이므로 흡수법칙에 따라 옳다.

② O — $A + \overline{A}B = (A + \overline{A})(A + B) = 1 \cdot (A + B) = A + B$이므로 옳다.

③ O — $A(\overline{A} + B) = A\overline{A} + AB = 0 + AB = AB$이므로 옳다.

④ X — 틀린 부분은 좌변을 A로 최소화한 것이다. 실제로는 $AB + A\overline{B} + \overline{A}B = A(B + \overline{B}) + \overline{A}B = A + \overline{A}B = A + B$이다. 따라서 올바른 식은 $AB + A\overline{B} + \overline{A}B = A + B$이다.

오답 포인트

④에서 앞의 두 항 $AB + A\overline{B}$만 묶으면 A가 되지만, 남은 항 $\overline{A}B$까지 더한 뒤에는 변형된 흡수법칙에 따라 $A + B$가 된다. 남은 항을 빠뜨리고 곧바로 A라고 판단하기 쉽다.

다시 나오면 이렇게 판단

곱의 합 형태는 공통 인수를 먼저 묶고, 보수쌍을 1 또는 0으로 바꾼 다음, 남은 모든 항에 흡수법칙을 적용해 최종식을 확인한다.

암기 포인트

$A + AB = A$, $A + \overline{A}B = A + B$이다. 보수가 붙은 흡수식은 결과에 B가 남는다.

NoteBase

문 15. 배열(Array)과 연결리스트(Linked List)에 대한 설명으로 옳지 않은 것은?

① 연결리스트는 배열에 비하여 희소행렬을 표현하는데 비효율적이다.

② 연결리스트에 비하여 배열은 원소를 임의의 위치에 삽입하는 비용이 크다.

③ 연결리스트에 비하여 배열은 임의의 위치에 있는 원소를 접근할 때 효율적이다.

④ n개의 원소를 관리할 때, 연결리스트가 n 크기의 배열보다 메모리 사용량이 더 크다.

정답 1번

출제 포인트

배열과 연결리스트를 원소 접근, 삽입, 메모리 사용, 희소행렬 표현의 관점에서 비교하는 문제이다.

핵심 개념

배열: 원소를 연속된 메모리 공간에 저장하는 자료구조이다. 인덱스로 임의 위치의 원소에 빠르게 접근할 수 있지만, 중간 삽입 시 뒤 원소들을 이동해야 한다.
연결리스트: 각 노드가 데이터와 다음 노드를 가리키는 연결 정보를 함께 저장하는 자료구조이다. 위치를 이미 알고 있다면 연결 정보만 바꾸어 삽입할 수 있지만, 임의 위치의 원소에 접근하려면 앞에서부터 따라가야 하고 연결 정보만큼 추가 메모리가 필요하다.
희소행렬: 대부분의 원소가 0인 행렬이다. 0이 아닌 원소와 그 위치만 연결리스트로 저장하면 모든 원소를 배열에 저장하는 것보다 공간을 절약할 수 있다.

NoteBase

선지별 해설

① X — 틀린 표현은 '연결리스트는 배열에 비하여 희소행렬을 표현하는 데 비효율적이다'이다. 희소행렬은 0이 아닌 원소와 위치 정보만 연결리스트에 저장할 수 있으므로, 모든 위치를 저장하는 일반 배열보다 메모리를 효율적으로 사용할 수 있다. 올바르게는 '연결리스트는 배열에 비하여 희소행렬을 표현하는 데 효율적이다'이다.
② O — 배열의 임의 위치에 원소를 삽입하면 그 뒤의 원소들을 이동해야 하므로, 연결 위치를 알고 있을 때 연결 정보만 변경하는 연결리스트보다 삽입 비용이 크다.
③ O — 배열은 인덱스를 이용해 임의 위치의 원소에 직접 접근할 수 있어 접근 시간이 $O(1)$이다. 연결리스트는 첫 노드부터 순서대로 따라가야 하므로 임의 위치 접근이 $O(n)$이고, 배열이 더 효율적이다.
④ O — 연결리스트의 각 노드는 데이터뿐 아니라 다음 노드를 가리키는 포인터 등의 연결 정보도 저장한다. 따라서 같은 n개의 원소를 관리할 때 원소만 연속 저장하는 n 크기의 배열보다 일반적으로 메모리 사용량이 더 크다.

오답 포인트

①은 연결리스트가 포인터 때문에 추가 메모리를 쓴다는 일반적 단점을 희소행렬에도 그대로 적용하도록 유도한다. 그러나 희소행렬에서는 다수의 0을 저장하지 않는 이점이 연결 정보의 추가 비용보다 중요하므로 '비효율적'이 아니라 '효율적'이다. 또한 ②는 삽입할 위치를 찾는 탐색 비용이 아니라, 위치가 정해진 뒤의 삽입 연산 비용을 비교한 표현으로 판단한다.

다시 나오면 이렇게 판단

임의 접근은 인덱스를 쓰는 배열이 유리하고, 중간 삽입·삭제는 연결을 바꾸는 연결리스트가 유리하다고 구분한다. 메모리는 일반적으로 배열이 유리하지만, 희소행렬처럼 저장하지 않아도 되는 값이 매우 많으면 필요한 원소만 저장하는 연결리스트가 유리할 수 있다.

암기 포인트

배열은 접근, 연결리스트는 삽입·삭제, 희소행렬은 필요한 값만 연결해 저장한다.

NoteBase

문 16. 프로세스 P1, P2, P3, P4를 선입선출(First In First Out) 방식으로 스케줄링을 수행할 경우 평균응답시간으로 옳은 것은? (단, 응답시간은 프로세스 도착시간부터 처리가 종료될 때까지의 시간을 말한다)

프로세스	도착시간	처리시간
P1	0	2
P2	2	2
P3	3	3
P4	4	9

① 3 ② 4 ③ 5 ④ 6

정답 3번

출제 포인트

선입선출(FIFO) 방식에서 도착 순서대로 실행 구간을 정한 뒤, 각 프로세스의 응답시간과 그 평균을 계산하는 문제이다.

조건 정리

도착시간과 처리시간은 각각 P1=(0, 2), P2=(2, 2), P3=(3, 3), P4=(4, 9)이다.
FIFO에서는 먼저 도착한 프로세스부터 처리하며, 응답시간은 문제의 정의에 따라 도착 시점부터 처리 종료 시점까지의 시간이다.

사용할 공식

프로세스 i의 도착시간을 A_i, 종료시간을 C_i, 응답시간을 R_i라고 하면 $R_i = C_i - A_i$이다. 평균응답시간은 다음과 같다.

$$\bar{R} = \frac{\sum_{i=1}^{n} R_i}{n}$$

FIFO는 도착 순서대로 비선점 실행되므로 먼저 각 프로세스의 종료시간을 구해야 한다.

계산 과정

1. P1은 0부터 2까지 실행되므로 응답시간은 $2 - 0 = 2$이다.
2. P2는 2부터 4까지 실행되므로 응답시간은 $4 - 2 = 2$이다.
3. P3은 P2가 끝난 뒤인 4부터 7까지 실행되므로 응답시간은 $7 - 3 = 4$이다.
4. P4는 P3이 끝난 뒤인 7부터 16까지 실행되므로 응답시간은 $16 - 4 = 12$이다.
5. 평균응답시간은 $\frac{2+2+4+12}{4} = 5$이다.

NoteBase

선지 판단

① X — 3은 계산 결과와 다르다. 올바르게는 네 응답시간 2, 2, 4, 12의 평균인 5이다.
② X — 4는 P3 한 개의 응답시간일 뿐 전체 평균이 아니다. 올바르게는 네 프로세스의 응답시간을 모두 합해 4로 나눈 5이다.
③ O — 평균응답시간이 5이므로 옳다.
④ X — 6은 계산 결과와 다르다. 올바르게는 P4가 P3 종료 후 실행된다는 점까지 반영하여 평균을 5로 계산한다.

오답 포인트

응답시간을 처리시간 또는 대기시간과 혼동하면 틀리기 쉽다. 이 문항의 응답시간은 도착부터 종료까지이므로 P4의 응답시간에는 3만큼의 대기시간과 9만큼의 처리시간이 모두 포함되어 12가 된다.

다시 나오면 이렇게 판단

FIFO 문항에서는 도착 순서대로 실행 구간과 종료시간을 먼저 정하고, 각 프로세스마다 종료시간에서 도착시간을 뺀 뒤 그 결과의 평균을 구한다.

암기 포인트

이 문항에서 응답시간은 종료시간-도착시간이며, FIFO는 먼저 온 프로세스부터 끝까지 처리한다.

NoteBase

문 17. TCP/IP 프로토콜에 대한 설명으로 옳은 것은?

① TCP는 비연결형 프로토콜 방식을 사용한다.

② TCP는 네트워크 계층(Network Layer)에 속한다.

③ IP는 잘못 전송된 패킷에 대하여 재전송을 요청하는 기능을 제공한다.

④ IP는 각 패킷의 주소 부분을 처리하여 패킷이 목적지에 도달할 수 있도록 한다.

정답 4번

출제 포인트

TCP와 IP가 담당하는 계층과 기능을 구분하는 문제이다.

핵심 개념

TCP: 전송 계층에서 동작하는 연결형 프로토콜로, 신뢰성 있는 데이터 전송을 위해 오류 확인과 재전송 등을 담당한다.
IP: 네트워크 계층에서 동작하는 비연결형 프로토콜로, 패킷의 IP 주소를 바탕으로 목적지까지 전달되도록 한다.
핵심 구분: TCP는 신뢰성 있는 전송, IP는 주소 지정과 패킷 전달을 담당한다.

선지별 해설

① X — 틀린 표현은 'TCP는 비연결형 프로토콜 방식을 사용한다'이다. TCP는 통신 전에 연결을 설정하는 연결형 프로토콜이다.
② X — 틀린 표현은 'TCP는 네트워크 계층에 속한다'이다. TCP는 전송 계층에 속하고, IP가 네트워크 계층에 속한다.
③ X — 틀린 표현은 'IP는 잘못 전송된 패킷에 대하여 재전송을 요청한다'이다. IP 자체는 재전송을 보장하지 않으며, 신뢰성 제어와 재전송은 TCP가 담당한다.
④ O — IP는 패킷의 출발지·목적지 주소 정보를 처리하고 경로를 따라 전달되도록 하므로 옳다.

오답 포인트

TCP와 IP가 함께 사용된다는 이유로 두 프로토콜의 계층과 역할을 뒤섞기 쉽다. 특히 ③은 재전송 기능을 IP의 기능처럼 제시하지만, 재전송을 통한 신뢰성 확보는 TCP의 역할이다.

다시 나오면 이렇게 판단

연결 설정·신뢰성·재전송이 나오면 TCP, 주소 지정·네트워크 계층·패킷 전달이 나오면 IP로 판단한다.

NoteBase

암기 포인트

TCP는 연결과 신뢰성, IP는 주소와 전달이다.

NoteBase

문 18. 다음에서 설명하는 용어로 가장 옳은 것은?

> 프랭크 로젠블라트(Frank Rosenblatt)가 고안한 것으로 인공신경망 및 딥러닝의 기반이 되는 알고리즘이다.

① 빠른 정렬(Quick Sort)
② 맵리듀스(MapReduce)
③ 퍼셉트론(Perceptron)
④ 디지털 포렌식(Digital Forensics)

정답 3번

출제 포인트

프랭크 로젠블라트가 고안했으며 인공신경망과 딥러닝의 기초가 된 알고리즘의 명칭을 묻는 문제이다.

핵심 개념

퍼셉트론은 여러 입력에 가중치를 적용한 값을 바탕으로 결과를 판별하는 초기 인공신경망 모델이다. 프랭크 로젠블라트가 고안했으며, 신경망 학습의 기초가 되었다.
빠른 정렬은 피벗을 기준으로 데이터를 나누어 정렬하는 알고리즘이고, 맵리듀스는 대규모 데이터를 분산 처리하는 방식이며, 디지털 포렌식은 디지털 증거를 수집·보존·분석하는 절차와 기술이다.

NoteBase

선지별 해설

① X
빠른 정렬은 피벗을 기준으로 원소를 분할하고 각 부분을 정렬하는 정렬 알고리즘이므로 제시문의 설명과 다르다. 틀린 표현은 '빠른 정렬'이며, 올바르게는 '퍼셉트론'이다.

② X
맵리듀스는 데이터를 Map 단계에서 나누어 처리하고 Reduce 단계에서 결과를 결합하는 대규모 분산 처리 방식이므로 로젠블라트가 고안한 신경망 모델이 아니다. 틀린 표현은 '맵리듀스'이며, 올바르게는 '퍼셉트론'이다.

③ O
퍼셉트론은 프랭크 로젠블라트가 고안한 초기 인공신경망 모델로, 제시문의 설명과 일치한다.

④ X
디지털 포렌식은 디지털 기기와 저장매체에서 증거를 수집·보존·분석하는 분야이므로 인공신경망의 기반 알고리즘이 아니다. 틀린 표현은 '디지털 포렌식'이며, 올바르게는 '퍼셉트론'이다.

오답 포인트

선지들이 모두 컴퓨터 분야의 용어이지만 쓰임이 서로 다르다. 제시문의 '프랭크 로젠블라트'와 '인공신경망'이라는 두 단서를 놓치면 정렬·분산 처리·디지털 증거 분석 용어와 혼동하기 쉽다.

다시 나오면 이렇게 판단

인물과 핵심 분야를 함께 대응한다. '로젠블라트＋인공신경망'이면 퍼셉트론, '피벗＋분할 정렬'이면 빠른 정렬, 'Map과 Reduce＋분산 처리'이면 맵리듀스, '디지털 증거＋수집·분석'이면 디지털 포렌식으로 판단한다.

암기 포인트

로젠블라트－퍼셉트론－인공신경망의 기초로 연결한다.

NoteBase

문 19. 관계형 데이터베이스의 뷰(View)에 대한 장점으로 옳지 않은 것은?

① 뷰는 데이터의 논리적 독립성을 일정 부분 제공할 수 있다.
② 뷰를 통해 데이터의 접근을 제어함으로써 보안을 제공할 수 있다.
③ 뷰에 대한 연산의 제약이 없어서 효율적인 응용프로그램의 개발이 가능하다.
④ 뷰는 여러 사용자의 상이한 응용이나 요구를 지원할 수 있어서 데이터 관리를 단순하게 한다.

정답 3번

출제 포인트

관계형 데이터베이스의 뷰가 제공하는 장점과 뷰 연산의 제약을 구별하는 문제이다.

핵심 개념

뷰(View)는 하나 이상의 기본 테이블에서 필요한 데이터만 골라 만든 가상 테이블이다. 사용자마다 필요한 형태로 데이터를 보여 주므로 논리적 독립성, 접근 제어, 사용 편의성을 제공하지만, 기본 테이블의 구조와 뷰의 정의 방식에 따라 삽입·삭제·갱신 같은 연산이 제한될 수 있다.

선지별 해설

① O — 사용자는 기본 테이블의 물리적 구조나 전체 구성을 직접 다루지 않고 뷰를 통해 필요한 논리적 구조를 사용할 수 있으므로 논리적 데이터 독립성을 일정 부분 제공한다.

② O — 사용자별로 허용된 행이나 열만 포함한 뷰를 제공하면 원본 데이터에 대한 접근 범위를 제한할 수 있으므로 보안에 도움이 된다.

③ X — 틀린 표현은 '뷰에 대한 연산의 제약이 없어서'이다. 실제로는 조인, 집계, 그룹화 등을 포함한 뷰처럼 정의 방식에 따라 삽입·삭제·갱신 연산이 제한될 수 있으므로 모든 연산에 제약이 없다고 할 수 없다.

④ O — 같은 기본 데이터로도 사용자나 업무별 요구에 맞는 여러 뷰를 만들 수 있어 데이터 접근과 관리를 단순하게 할 수 있다.

NoteBase

오답 포인트

③은 뷰가 응용 프로그램 개발과 사용 편의성에 도움을 준다는 사실에 '연산의 제약이 없다'는 잘못된 이유를 붙였다. 뷰의 편의성과 갱신 가능성은 별개의 성질이다.

다시 나오면 이렇게 판단

뷰의 장점으로는 필요한 데이터만 제공하는 보안성, 사용자별 표현, 논리적 독립성과 편의성을 확인한다. 반면 '모든 연산이 가능하다', '갱신 제약이 없다'와 같은 절대적 표현은 뷰의 갱신 제한에 어긋나는지 판단한다.

암기 포인트

뷰는 보안·독립성·편의성에는 유리하지만, 갱신 연산은 제한될 수 있다.

NoteBase

문 20. 다음에서 설명하는 알고리즘 설계 기법으로 가장 알맞은 것은?

> 해결하고자 하는 문제의 최적해(Optimal Solution)가 부분 문제들의 최적해들로 구성되어 있을 경우, 이를 이용하여 문제의 최적해를 구하는 기법이다.

① 동적 계획법(Dynamic Programming)
② 탐욕적 알고리즘(Greedy Algorithm)
③ 재귀 프로그래밍(Recursive Programming)
④ 근사 알고리즘(Approximation Algorithm)

정답 1번

출제 포인트

전체 문제의 최적해가 부분 문제의 최적해들로 구성되는 성질을 이용하는 알고리즘 설계 기법을 구별하는 문제이다.

핵심 개념

동적 계획법은 큰 문제를 겹치는 부분 문제로 나누고, 각 부분 문제의 해를 저장한 뒤 재사용하여 전체 문제의 최적해를 구하는 기법이다.
결정적 키워드는 '전체 최적해가 부분 문제의 최적해로 구성된다'는 최적 부분 구조와 '부분 문제의 해를 재사용한다'는 점이다.

선지별 해설

① O — 동적 계획법은 부분 문제의 최적해를 구해 저장하고 이를 결합·재사용하여 전체 문제의 최적해를 구하므로 제시문의 설명과 일치한다.
② X — 틀린 표현은 '탐욕적 알고리즘'이다. 탐욕적 알고리즘은 각 단계에서 당장 가장 좋아 보이는 선택을 하는 기법이며, 제시문처럼 부분 문제의 최적해를 저장하고 이용하는 기법은 동적 계획법이다.
③ X — 틀린 표현은 '재귀 프로그래밍'이다. 재귀는 함수가 자기 자신을 호출하여 문제를 푸는 구현 방식일 뿐, 부분 문제의 최적해를 저장·재사용하여 전체 최적해를 구한다는 설계 기법을 뜻하지 않는다. 올바르게는 동적 계획법이다.
④ X — 틀린 표현은 '근사 알고리즘'이다. 근사 알고리즘은 정확한 최적해를 효율적으로 구하기 어려울 때 최적해에 가까운 해를 구하는 기법이다. 제시문처럼 부분 문제의 최적해를 이용해 정확한 최적해를 구하는 기법은 동적 계획법이다.

NoteBase

오답 포인트

②의 탐욕적 알고리즘도 '최적'이라는 말과 함께 설명되는 경우가 있어 혼동하기 쉽다. 그러나 탐욕적 알고리즘은 매 단계의 국소적 최선 선택이 핵심이고, 동적 계획법은 부분 문제의 해를 저장하고 결합하는 것이 핵심이다.

다시 나오면 이렇게 판단

'부분 문제의 최적해를 저장·재사용하여 전체 최적해를 구성한다'가 보이면 동적 계획법으로 판단한다. '매 순간 가장 좋은 선택'은 탐욕적 알고리즘, '최적해에 가까운 해'는 근사 알고리즘으로 구별한다.

암기 포인트

부분 최적해를 저장해 전체 최적해를 만든다 = 동적 계획법이다.

NoteBase