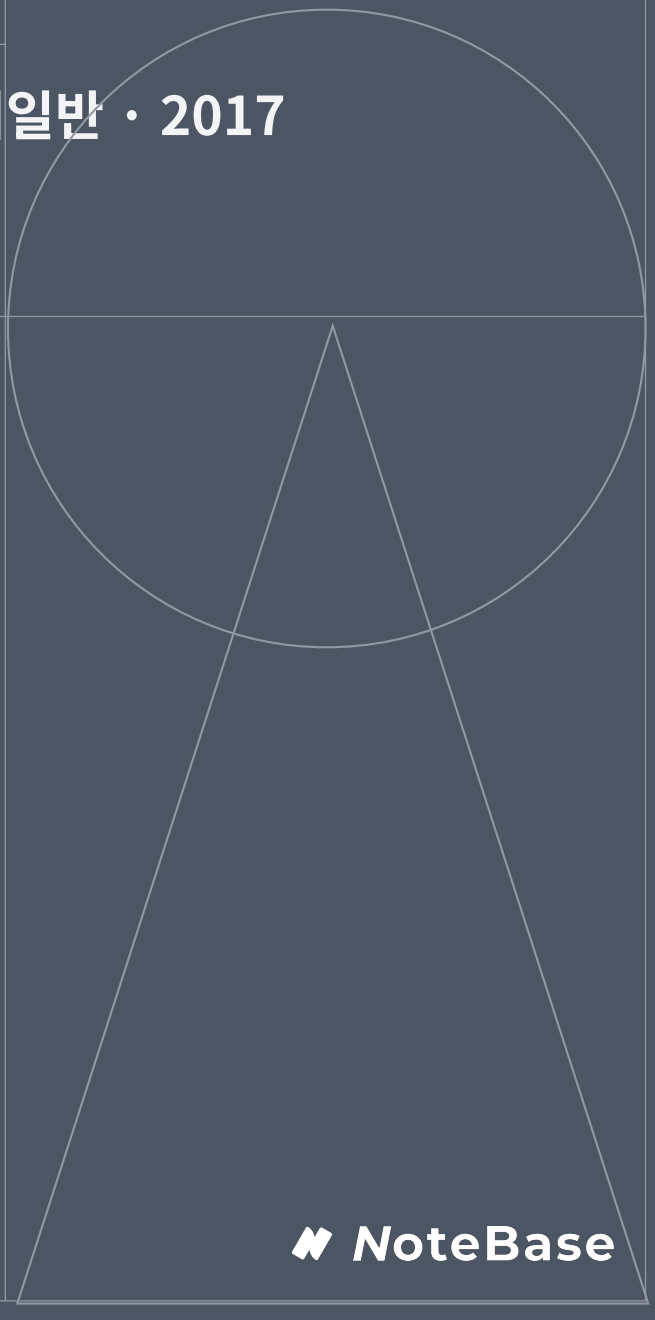
# 기출OX 해설집

공무원/국회직/9급 · 컴퓨터일반 · 2017



NoteBase

# 기출OX 해설집

## 공무원/국회직/9급 · 컴퓨터일반 · 2017

### 공무원/국회직/9급 · 컴퓨터일반 · 2017

1. CPU의 주소지정 방식 중 간접 주소지정 방식(indirect addressing mode)을 이용하는 이유로 옳은 것은?

① 메모리에 직접 접근하지 못하게 하여 보안을 강화한다.
② 메모리 접근 속도를 빠르게 한다.
③ 제작 비용을 절감한다.
④ CPU의 실행 속도를 빠르게 한다.
⑤ 지정 가능한 주소 범위를 확대한다.

**정답 5번**

**출제 포인트**

간접 주소지정 방식의 동작 원리와 이를 사용하는 핵심 이유인 주소 범위 확장을 묻는 문제이다.

**핵심 개념**

간접 주소지정 방식은 명령어의 주소 필드가 피연산자의 실제 주소를 곧바로 나타내지 않고, 실제 유효주소가 저장된 메모리 위치를 가리키는 방식이다. 메모리에 저장된 주소를 한 번 더 따라가므로 명령어의 제한된 주소 필드를 보완하여 더 넓은 주소 범위를 지정할 수 있다. 다만 유효주소를 얻기 위한 메모리 접근이 추가되므로 속도 향상을 위한 방식은 아니다.

NoteBase

**선지별 해설**

① X — 틀린 표현은 '메모리에 직접 접근하지 못하게 하여 보안을 강화한다'이다. 간접 주소지정도 유효주소를 얻고 피연산자를 읽기 위해 메모리에 접근하며, 보안 강화가 목적이 아니다. 올바르게는 '메모리에 저장된 주소를 거쳐 피연산자의 유효주소를 얻는다'이다.
② X — 틀린 표현은 '메모리 접근 속도를 빠르게 한다'이다. 유효주소를 읽기 위한 메모리 접근이 추가되므로 일반적으로 직접 주소지정보다 느리다. 올바르게는 '추가 메모리 접근으로 접근 시간이 늘어날 수 있다'이다.
③ X — 틀린 표현은 '제작 비용을 절감한다'이다. 간접 주소지정은 주소 해석 기능과 추가 메모리 접근이 필요한 방식이므로 제작 비용 절감이 핵심 이유가 아니다. 올바르게는 '제한된 주소 필드를 보완하여 지정 가능한 주소 범위를 넓힌다'이다.
④ X — 틀린 표현은 'CPU의 실행 속도를 빠르게 한다'이다. 간접 주소를 해석하려면 유효주소를 얻는 단계가 추가되므로 실행 속도 향상과는 반대이다. 올바르게는 '주소 해석 단계가 늘어 실행 시간이 증가할 수 있다'이다.
⑤ O — 메모리에 저장된 주소를 다시 참조하면 명령어의 주소 필드만으로 직접 표현하기 어려운 주소까지 지정할 수 있으므로 지정 가능한 주소 범위가 확대된다.

**오답 포인트**

'간접'이라는 말 때문에 ①의 보안 기능으로 오해하거나, 주소를 한 번 더 거치므로 ②·④처럼 속도가 빨라진다고 착각하기 쉽다. 실제로 한 단계가 추가되면 속도는 느려질 수 있고, 얻는 이점은 주소 범위의 확대이다.

**다시 나오면 이렇게 판단**

주소지정 방식의 장점을 물으면 먼저 주소를 몇 단계로 얻는지 본다. 메모리 속 주소를 다시 따라가는 간접 주소지정은 '추가 접근으로 속도는 불리하지만 주소 표현 범위는 유리하다'고 판단한다.

**암기 포인트**

간접 주소지정은 주소를 한 번 더 따라가므로 속도는 느려질 수 있고, 주소 범위는 넓어진다.

NoteBase

2. 자신이 수행한 행동(action)에 따른 보상(reward)을 통해 스스로 문제 해결방법을 찾아내도록 하는 기계학습(machine learning) 기법으로 옳은 것은?

① 베이시안 네트워크(bayesian network)
② 지도학습(supervised learning)
③ 군집화(clustering)
④ 경사 하강법(gradient descent method)
⑤ 강화학습(reinforcement learning)

**정답 5번**

**출제 포인트**

이 문제는 행동의 결과로 받은 보상을 이용해 더 나은 행동 전략을 학습하는 기계학습 방법을 구별하는 문제이다.

**핵심 개념**

강화학습은 에이전트가 환경에서 행동하고 보상을 받는 과정을 반복하면서 누적 보상을 크게 만드는 행동 전략을 학습하는 방법이다.
지도학습은 정답이 표시된 학습 데이터로 입력과 정답의 관계를 학습하고, 군집화는 정답 표시 없이 비슷한 데이터끼리 묶는다. 베이지안 네트워크는 변수 사이의 확률적 의존 관계를 그래프로 표현하며, 경사 하강법은 손실 함수를 줄이기 위한 최적화 방법이다.

**선지별 해설**

① X — 베이지안 네트워크는 확률변수와 변수 사이의 조건부 의존 관계를 방향성 그래프로 표현하는 확률 모형이다. '행동에 따른 보상으로 학습하는 기법'이라는 연결이 틀렸으며, 올바른 개념은 강화학습이다.
② X — 지도학습은 정답이 붙은 입력·출력 데이터로 예측 규칙을 학습한다. '행동에 따른 보상으로 스스로 해결방법을 찾는다'는 설명이 틀렸으며, 올바르게는 강화학습에 해당한다.
③ X — 군집화는 정답표 없이 데이터의 유사성을 기준으로 여러 집단으로 묶는 비지도학습 기법이다. '보상을 기준으로 행동을 개선한다'는 설명이 틀렸으며, 올바른 개념은 강화학습이다.
④ X — 경사 하강법은 손실 함수의 값을 줄이는 방향으로 모델의 매개변수를 반복 갱신하는 최적화 방법이다. '행동과 보상의 상호작용으로 문제 해결 전략을 찾는다'는 설명이 틀렸으며, 올바르게는 강화학습이다.
⑤ O — 강화학습은 자신이 선택한 행동에 대해 환경이 제공하는 보상을 이용하여 어떤 행동을 선택할지에 관한 전략을 학습하므로 문제의 설명과 일치한다.

**오답 포인트**

②의 지도학습도 학습 결과를 개선한다는 점 때문에 헷갈릴 수 있지만, 지도학습의 핵심 신호는 정답이고 강화학습의 핵심 신호는 행동 뒤에 받는 보상이다. 또한 ④ 경사 하강법은 모델을 학습시키는 최적화 수단이지, 행동·보상 구조 자체를 뜻하지 않는다.

NoteBase

**다시 나오면 이렇게 판단**

문장에서 ‘행동’, ‘환경’, ‘보상’이 함께 제시되면 강화학습으로 판단한다. 정답 데이터가 제시되면 지도학습, 유사한 데이터끼리 묶으면 군집화, 손실을 줄이는 매개변수 갱신이면 경사 하강법으로 구별한다.

**암기 포인트**

지도학습은 정답, 군집화는 유사성, 경사 하강법은 손실 감소, 강화학습은 행동과 보상이다.

NoteBase

3. 가상메모리(virtual memory) 관리 기법에 대한 설명으로 옳지 않은 것은?

① 프로그램의 크기가 실제 메모리의 크기보다 커도 실행이 가능하다.
② 페이지 방식은 페이지 사상 작업으로 인하여 비용이 증가하고 수행 속도가 감소한다.
③ 페이지 방식은 페이지를 작게 설계하여 내부 단편화를 줄일 수 있다.
④ 세그먼트 방식은 동적 메모리 할당 방법으로 외부 단편화가 발생하지 않는다.
⑤ 세그먼트 방식은 세그먼트 번호와 변위(offset)로 구성하는 가상주소를 만든다.

**정답 4번**

**출제 포인트**

가상 메모리의 페이징과 세그먼테이션이 메모리를 나누는 방식, 주소 구성, 단편화에 미치는 영향을 구별하는 문제이다.

**핵심 개념**

가상 메모리는 프로그램이 실제 주기억장치보다 큰 연속 주소 공간을 사용하는 것처럼 보이게 하고, 필요한 부분만 주기억장치에 적재하는 관리 기법이다.
페이징은 가상 주소 공간과 물리 메모리를 같은 크기의 고정 단위인 페이지와 프레임으로 나눈다. 외부 단편화는 없지만 마지막 페이지 등에 내부 단편화가 생길 수 있다.
세그먼테이션은 코드·데이터·스택처럼 논리적 의미에 따라 서로 다른 크기의 세그먼트로 나누는 가변 크기 할당 방식이다. 주소는 세그먼트 번호와 변위로 구성되며 외부 단편화가 생길 수 있다.

NoteBase

### 선지별 해설

① O — 가상 메모리는 필요한 부분만 실제 메모리에 올리고 나머지는 보조기억장치에 둘 수 있으므로, 실제 메모리보다 큰 프로그램도 실행할 수 있다.
② O — 페이징은 페이지 테이블을 이용한 주소 변환과 페이지 교체 등의 관리 작업이 필요하므로 시간·공간 오버헤드가 발생하고 수행 속도가 저하될 수 있다.
③ O — 페이지 크기가 작아지면 마지막 페이지에서 사용되지 않고 남는 공간이 줄어 내부 단편화를 줄일 수 있다. 다만 페이지 수와 페이지 테이블 크기는 증가할 수 있다.
④ X — 틀린 표현은 '외부 단편화가 발생하지 않는다'이다. 세그먼테이션은 서로 다른 크기의 세그먼트를 동적으로 배치하므로 빈 공간이 흩어지는 외부 단편화가 발생할 수 있다. 올바르게는 '세그먼트 방식은 동적 메모리 할당 방법으로 외부 단편화가 발생할 수 있다'이다.
⑤ O — 세그먼테이션의 가상 주소는 어느 세그먼트인지를 나타내는 세그먼트 번호와 그 세그먼트 내부 위치를 나타내는 변위로 구성된다.

### 오답 포인트

④의 '동적 메모리 할당'과 '외부 단편화가 발생하지 않는다'를 함께 옳은 특징으로 착각하기 쉽다. 가변 크기로 할당하는 세그먼테이션은 오히려 할당과 해제가 반복될 때 빈 공간이 여러 곳에 흩어져 외부 단편화가 발생한다.

### 다시 나오면 이렇게 판단

고정 크기인 페이징은 외부 단편화가 없고 내부 단편화가 생길 수 있으며, 가변 크기인 세그먼테이션은 외부 단편화가 생길 수 있다고 구별한다. 주소 구성은 페이징이면 페이지 번호와 변위, 세그먼테이션이면 세그먼트 번호와 변위로 판단한다.

### 암기 포인트

페이징은 고정 크기·내부 단편화, 세그먼테이션은 가변 크기·외부 단편화이다.

NoteBase

4. UNIX 파일 시스템에서 i-node에 존재하는 정보로 옳지 않은 것은?

① 파일 이름
② 파일 저장 위치
③ 파일 생성 일시
④ 파일 소유자
⑤ 파일 접근권한 비트

**정답 1번**

**출제 포인트**

UNIX 파일 시스템에서 i-node에 저장되는 파일의 메타데이터와 디렉터리 엔트리에 저장되는 파일 이름을 구분하는 문제이다.

**핵심 개념**

i-node는 파일의 소유자, 접근 권한, 시간 정보, 크기, 데이터 블록 위치 등 파일을 관리하는 정보를 저장하는 구조이다. 파일 이름은 i-node 내부가 아니라 디렉터리 엔트리에 i-node 번호와 함께 저장된다.

**선지별 해설**

① X
파일 이름은 i-node에 저장되지 않는다. 틀린 표현은 '파일 이름'이며, 올바르게는 디렉터리 엔트리가 파일 이름과 i-node 번호의 대응 관계를 저장한다.

② O
i-node에는 파일 데이터가 저장된 디스크 블록을 가리키는 주소 또는 포인터가 있어 파일의 저장 위치를 찾을 수 있다.

③ O
전통적인 UNIX의 ctime은 파일 생성 시각이 아니라 i-node 상태 변경 시각이며, 생성 시각의 별도 저장 여부는 파일 시스템에 따라 다르다. 다만 이 문항에서는 '파일 생성 일시'를 i-node의 시간 메타데이터로 분류하는 출제 관행에 따라 ③을 옳은 것으로 처리한다.

④ O
i-node에는 파일 소유자를 식별하는 사용자 ID가 저장된다.

⑤ O
i-node에는 소유자·그룹·기타 사용자에게 허용된 읽기, 쓰기, 실행 권한을 나타내는 접근권한 비트가 저장된다.

NoteBase

**오답 포인트**

파일을 열 때 파일 이름으로 접근하므로 파일 이름도 i-node에 들어 있다고 생각하기 쉽다. 그러나 디렉터리는 '파일 이름 → i-node 번호'를 연결하고, 실제 파일 관리 정보는 해당 i-node가 보관한다.

**다시 나오면 이렇게 판단**

이름과 i-node 번호의 연결은 디렉터리 엔트리, 소유자·권한·시간·데이터 블록 위치 같은 파일 관리 정보는 i-node로 구분한다.

**암기 포인트**

파일 이름은 디렉터리, 파일의 메타데이터와 블록 위치는 i-node이다.

NoteBase

5. 스키마(schema)에 대한 설명으로 옳지 않은 것은?

① 내부 스키마는 범 기관적 입장에서 데이터베이스를 정의한 것이다.
② 개념 스키마는 모든 데이터 개체, 관계, 제약 조건, 접근권한, 무결성 규칙 등을 명세한다.
③ 개념 스키마는 일반적으로 스키마를 의미한다.
④ 외부 스키마는 사용자나 응용프로그래머가 접근할 수 있는 데이터베이스를 정의한다.
⑤ 스키마는 데이터베이스의 논리적 정의, 데이터 구조와 제약 조건에 관한 명세를 기술한 것이다.

**정답 1번**

**출제 포인트**

데이터베이스의 3단계 스키마인 외부 스키마, 개념 스키마, 내부 스키마의 관점과 역할을 구별하는 문제이다.

**핵심 개념**

외부 스키마: 개별 사용자나 응용 프로그램이 보는 데이터베이스의 논리적 관점이다. 키워드는 사용자별 뷰이다.
개념 스키마: 조직 전체의 관점에서 데이터베이스의 전체 논리 구조와 제약 조건을 정의한 통합 스키마이다.
키워드는 범기관적·전체적 관점이다.
내부 스키마: 데이터가 저장 장치에 실제로 어떻게 저장되는지를 정의한 물리적 저장 관점의 스키마이다.
키워드는 저장 구조와 접근 경로이다.
핵심 대비는 외부는 사용자별 관점, 개념은 조직 전체의 논리적 관점, 내부는 물리적 저장 관점이라는 점이다.

**선지별 해설**

① X - 틀린 표현은 '내부 스키마는 범기관적 입장에서 데이터베이스를 정의한다'이다. 범기관적 입장에서 데이터베이스 전체의 논리 구조를 정의하는 것은 개념 스키마이며, 내부 스키마는 물리적 저장 구조와 접근 경로를 정의한다.
② O - 개념 스키마는 조직 전체의 데이터 개체, 관계, 제약 조건, 접근 권한, 무결성 규칙 등 데이터베이스의 전체 논리 구조를 명세한다.
③ O - 개념 스키마는 데이터베이스 전체의 통합된 논리 구조를 나타내므로 일반적으로 스키마라고 할 때 가리키는 스키마이다.
④ O - 외부 스키마는 사용자나 응용 프로그램이 필요한 데이터에 접근하도록 구성한 사용자별 논리적 뷰를 정의한다.
⑤ O - 스키마는 데이터베이스의 논리적 정의로서 데이터 구조와 그 구조에 적용되는 제약 조건에 관한 명세를 기술한다.

NoteBase

### 오답 포인트

①의 '범기관적'이라는 말 때문에 내부 스키마를 데이터베이스 내부 전체의 관점으로 오해하기 쉽다. 그러나 '내부'는 조직 내부가 아니라 저장 장치의 물리적 내부 표현을 뜻하며, 범기관적·전체적 관점은 개념 스키마에 해당한다.

### 다시 나오면 이렇게 판단

사용자별 화면이나 뷰이면 외부 스키마, 조직 전체의 통합된 논리 구조이면 개념 스키마, 파일 조직·저장 구조·접근 경로와 같은 물리적 표현이면 내부 스키마로 판단한다.

### 암기 포인트

외부는 사용자별 뷰, 개념은 조직 전체의 논리 구조, 내부는 물리적 저장 구조이다.

NoteBase

6. 다음 SQL 문장 중 구문이 옳은 것은?

① DELETE FROM STUDENT, ENROL
WHERE SNO = 100;

② SELECT COUNT(DISTINCT CNO)
FROM ENROL
WHERE SNO = 100;

③ SELECT SNO, SNAME
FROM STUDENT
WHERE DEPT = NULL;

④ INSERT STUDENT INTO VALUES(100, '홍길동', 2, '전산과')

⑤ SELECT DNO, SUM(SNAME)
FROM STUDENT
GROUP BY DNO
WHERE SUM(SNAME)<100;

**정답 2번**

**출제 포인트**

SQL에서 DELETE·SELECT·INSERT의 기본 구문, NULL 비교, 집계 함수와 GROUP BY·HAVING의 사용법을 구별하는 문제이다.

**핵심 개념**

DELETE는 DELETE FROM 테이블명 WHERE 조건의 순서로 작성한다. NULL 여부는 =가 아니라 IS NULL 또는 IS NOT NULL로 판정한다. INSERT는 INSERT INTO 테이블명 VALUES(값 목록)의 순서로 작성한다. 집계 전 행을 거르는 조건은 WHERE, 그룹화한 뒤 집계 결과를 거르는 조건은 HAVING에 둔다. SUM은 합계를 계산할 수 있는 수치형 값에 사용한다.

NoteBase

**선지별 해설**

① X
DELETE FROM 뒤에 STUDENT, ENROL처럼 두 테이블을 나열한 부분이 틀렸다. 일반적인 DELETE 문은 한 번에 하나의 대상 테이블을 지정하므로, 올바르게는 DELETE FROM STUDENT WHERE SNO = 100;과 DELETE FROM ENROL WHERE SNO = 100;처럼 각 테이블에 대해 별도의 문장을 작성한다.

② O
SELECT COUNT(DISTINCT CNO) FROM ENROL WHERE SNO = 100;은 ENROL에서 학번 SNO가 100인 행만 고른 뒤 중복되지 않는 과목번호 CNO의 개수를 계산하는 올바른 구문이다.

③ X
DEPT = NULL이 틀렸다. NULL은 일반 비교 연산자 =로 비교하지 않으므로, 올바르게는 WHERE DEPT IS NULL로 작성한다.

④ X
INSERT STUDENT INTO의 키워드 순서가 틀렸다. 올바르게는 INSERT INTO STUDENT VALUES(100, '홍길동', 2, '전산과');로 작성한다.

⑤ X
문자형 이름 SNAME에 SUM을 적용한 점, GROUP BY 뒤에 WHERE를 둔 점, 집계 결과를 WHERE로 제한한 점이 모두 틀렸다. 부서별 학생 수가 100명 미만인 그룹을 구하려는 경우에는 SELECT DNO, COUNT(*) FROM STUDENT GROUP BY DNO HAVING COUNT(*) < 100;으로 작성한다.

**오답 포인트**

③에서는 NULL을 보통 값처럼 =로 비교하기 쉽고, ⑤에서는 WHERE와 HAVING을 혼동하기 쉽다. 특히 집계 함수가 포함되었다고 해서 어떤 열에나 SUM을 쓸 수 있는 것은 아니며, 집계 결과 조건은 GROUP BY 뒤의 HAVING에 둔다.

**다시 나오면 이렇게 판단**

각 문장을 명령어별 기본 뼈대와 대조한다. NULL은 IS NULL인지, INSERT INTO의 순서가 맞는지, 집계 함수의 대상 자료형이 적절한지, SQL 절의 순서가 WHERE → GROUP BY → HAVING인지 차례로 확인한다.

**암기 포인트**

NULL은 IS NULL, 삽입은 INSERT INTO, 그룹 조건은 HAVING이다.

NoteBase

7. 다음 정수 리스트를 퀵 정렬 알고리즘으로 오름차순 정렬할 때, 리스트를 처음 분할한 직후의 분할된 두 리스트의 상태로 옳은 것은? (단, 제어키는 5로 한다.)

| (5 | 2 | 6 | 4 | 7 | 3 | 8 | 1) |
|---|---|---|---|---|---|---|---|

① (5 2 6 4), (7 3 8 1)
② (2 4 3 1), (6 7 8)
③ (3 1 2 4), (7 6 8)
④ (3 2 1 4), (7 8 6)
⑤ (1 2 3 4), (6 7 8)

**정답 4번**

**출제 포인트**

퀵 정렬에서 제어기 5를 기준으로 양쪽을 탐색하며 잘못 놓인 원소를 교환하고, 탐색 위치가 엇갈리면 제어기를 경계 위치로 옮기는 첫 분할 과정을 묻는 문제이다.

**올바른 순서**

1. 제어기 5의 왼쪽에서는 5보다 큰 첫 원소 6을 찾고, 오른쪽에서는 5보다 작은 첫 원소 1을 찾아 서로 교환한다. 그 결과는 (5 2 1 4 7 3 8 6)이다.
2. 다시 탐색하여 왼쪽의 7과 오른쪽의 3을 교환한다. 그 결과는 (5 2 1 4 3 7 8 6)이다.
3. 양쪽 탐색 위치가 엇갈리면 제어기 5와 오른쪽 탐색이 멈춘 3을 교환한다. 그 결과는 (3 2 1 4 5 7 8 6)이다.
4. 제어기 5를 제외하면 분할된 두 리스트는 (3 2 1 4), (7 8 6)이다.

왜 이 순서: 제어기보다 큰 원소와 작은 원소를 먼저 맞바꾼 뒤 제어기를 경계에 놓아야 왼쪽은 모두 5보다 작고 오른쪽은 모두 5보다 큰 상태가 되기 때문이다.

NoteBase

### 선지별 해설

① X — (5 2 6 4), (7 3 8 1)은 원래 리스트를 위치상 둘로만 나눈 상태이다. 왼쪽에 6, 오른쪽에 3과 1이 남아 있어 제어기 5를 기준으로 분할되지 않았다. 틀린 표현 (5 2 6 4), (7 3 8 1) → 올바르게는 (3 2 1 4), (7 8 6)이다.
② X — 각 리스트에 들어간 값의 대소 관계는 맞지만, 실제 교환 순서를 반영한 직후 상태가 아니다. 첫 분할 직후 왼쪽의 배열 순서는 3, 2, 1, 4이고 오른쪽의 배열 순서는 7, 8, 6이다. 틀린 표현 (2 4 3 1), (6 7 8) → 올바르게는 (3 2 1 4), (7 8 6)이다.
③ X — 제어기 5를 기준으로 한 원소의 소속은 맞지만, 실제 분할 과정에서 5와 마지막으로 교환되는 값은 3이므로 왼쪽 첫 원소는 3이 되고 2와 1의 상대 위치는 바뀌지 않는다. 틀린 표현 (3 1 2 4), (7 6 8) → 올바르게는 (3 2 1 4), (7 8 6)이다.
④ O — 6과 1, 7과 3을 차례로 교환한 뒤 5와 3을 교환하면 전체 상태는 (3 2 1 4 5 7 8 6)이다. 따라서 제어기 5를 제외한 두 리스트는 (3 2 1 4), (7 8 6)이다.
⑤ X — (1 2 3 4), (6 7 8)은 두 부분 리스트까지 오름차순으로 정렬한 결과이며, 문제는 최초 분할 직후를 묻는다. 틀린 표현 (1 2 3 4), (6 7 8) → 올바르게는 (3 2 1 4), (7 8 6)이다.

### 오답 포인트

첫 분할은 전체 정렬이나 부분 리스트의 정렬이 아니다. ②와 ③처럼 제어기보다 작은 값과 큰 값을 올바른 쪽에 모으기만 한 배열도 실제 교환 순서와 다르면 첫 분할 직후의 상태가 아니다. ⑤는 분할 결과를 각 부분 리스트의 정렬 완료 상태로 착각하게 하는 선지이다.

### 다시 나오면 이렇게 판단

첫 원소가 제어기인 분할에서는 왼쪽에서 제어기보다 큰 값을, 오른쪽에서 제어기보다 작은 값을 찾아 교환한다. 탐색 위치가 엇갈리면 제어기와 오른쪽 탐색이 멈춘 값을 교환한 뒤 제어기를 제외한 양쪽 리스트를 읽는다.

### 암기 포인트

퀵 정렬의 첫 분할은 완전 정렬이 아니라 '제어기보다 작은 값 | 제어기 | 제어기보다 큰 값'의 경계를 만드는 과정이다.

NoteBase

8. 다음 C 프로그램에서 main() 함수를 실행할 때 fib() 함수가 호출되는 횟수로 옳은 것은?

```
#include <stdio.h>
int fib(int n)
{
    if(n == 0) return 0;
    if(n == 1) return 1;
    return(fib(n-1) + fib(n-2));
}
int main()
{
    fib(5);
}
```

① 3번
② 5번
③ 10번
④ 12번
⑤ 15번

**정답 5번**

**출제 포인트**

재귀 함수에서 최초 호출과 기저 조건에 도달한 호출까지 포함한 전체 함수 호출 횟수를 구하는 문제이다.

**조건 정리**

주어진 값: main()에서 fib(5)를 한 번 호출한다.
구해야 할 값: fib(5)를 수행하는 동안 발생하는 fib()의 전체 호출 횟수이다.
주의할 조건: fib(0)과 fib(1)도 각각 함수가 한 번 호출된 뒤 반환되므로 호출 횟수에 포함하며, 최초의 fib(5) 호출도 포함한다.

**사용할 공식**

$C(n)$을 fib(n)을 한 번 실행할 때 발생하는 전체 fib() 호출 횟수라고 하면, 기저 조건은 $C(0) = C(1) = 1$이다. $n \geq 2$에서는 현재 호출 한 번과 두 재귀 호출을 합하므로 $C(n) = 1 + C(n-1) + C(n-2)$가 된다. 이 점화식은 fib(n)이 fib(n-1)과 fib(n-2)를 각각 호출하는 코드 구조를 그대로 센 것이다.

NoteBase

### 계산 과정

$C(0) = 1,\ C(1) = 1$
$C(2) = 1 + C(1) + C(0) = 3$
$C(3) = 1 + C(2) + C(1) = 5$
$C(4) = 1 + C(3) + C(2) = 9$
$C(5) = 1 + C(4) + C(3) = 15$
따라서 fib()는 모두 15번 호출된다.

### 선지 판단

① X — 3번은 $C(2)$의 호출 횟수이므로 fib(5)의 결과가 아니다. 올바르게는 15번이다.
② X — 5번은 $C(3)$의 호출 횟수이므로 fib(5)의 결과가 아니다. 올바르게는 15번이다.
③ X — 10번은 재귀 호출 트리의 모든 노드를 센 값이 아니다. 최초 호출과 기저 조건 호출을 모두 포함하면 올바르게는 15번이다.
④ X — 12번은 $C(n) = 1 + C(n-1) + C(n-2)$에 따라 계산한 값이 아니다. 올바르게는 15번이다.
⑤ O — $C(5) = 1 + C(4) + C(3) = 1 + 9 + 5 = 15$이므로 15번이 맞다.

### 오답 포인트

피보나치 수의 값 fib(5)=5와 함수 호출 횟수를 혼동하기 쉽다. 또한 현재 실행 중인 함수 자체의 한 번을 빼거나, fib(0)과 fib(1)이 즉시 반환된다는 이유로 이 호출들을 제외하면 오답이 된다.

### 다시 나오면 이렇게 판단

재귀 함수 호출 횟수는 호출 트리의 모든 노드 수로 판단한다. 기저 조건도 호출 한 번으로 세고, 비기저 호출에서는 '현재 호출 1번 + 각 재귀 호출의 전체 횟수'를 더한다.

### 암기 포인트

재귀 호출 횟수는 반환값이 아니라 호출 트리의 전체 노드 수이며, 기저 호출과 최초 호출도 각각 1번이다.

NoteBase

**9. 객체지향 언어에 대한 설명으로 옳지 않은 것은?**

① 객체는 재사용이 가능하나 클래스는 불가능하다.
② 객체는 하나의 클래스 타입으로 선언된다.
③ 오버라이딩(overriding), 오버로딩(overloading) 모두 다형성을 지원한다.
④ 추상메소드는 선언만 하고 그 내용은 기술하지 않은 메소드이다.
⑤ Smalltalk, Simula, Java 언어는 객체지향 개념을 잘 표현하고 있다.

**정답 1번**

### 출제 포인트

객체지향의 핵심 요소인 클래스와 객체의 관계, 다형성, 추상 메소드, 객체지향 언어의 사례를 구별하는 문제이다.

### 핵심 개념

클래스는 객체의 속성과 동작을 정의한 설계도이며, 객체는 그 클래스를 바탕으로 생성된 인스턴스이다. 객체지향에서는 클래스도 상속·포함 등의 방식으로 재사용할 수 있다. 오버라이딩은 상속받은 메소드를 재정의하는 것이고, 오버로딩은 같은 이름의 메소드를 매개변수 구성에 따라 여러 개 정의하는 것이다. 추상 메소드는 선언만 있고 구현 내용은 없는 메소드이다.

### 선지별 해설

① X — 틀린 표현은 '클래스는 불가능하다'이다. 객체지향에서는 클래스도 상속이나 포함 등을 통해 재사용할 수 있으므로, 올바르게는 '클래스도 재사용이 가능하다'이다.
② O — 클래스는 객체의 타입과 구조를 정의하고, 객체는 해당 클래스를 바탕으로 생성되는 인스턴스이므로 하나의 클래스 타입으로 선언된다.
③ O — 오버라이딩은 상속 관계에서 같은 메소드를 다르게 동작하게 하고, 오버로딩은 같은 메소드 이름으로 여러 형태의 호출을 허용하므로 모두 다형성을 지원한다.
④ O — 추상 메소드는 메소드의 이름·매개변수·반환형 등 선언부만 두고 구체적인 실행 내용은 하위 클래스에서 구현하도록 한 메소드이다.
⑤ O — Smalltalk, Simula, Java는 클래스와 객체, 상속, 메시지 또는 메소드 호출 등의 객체지향 개념을 표현하는 대표적인 객체지향 언어이다.

### 오답 포인트

①은 객체의 재사용 가능성을 강조하면서 클래스의 재사용 가능성을 부정해 혼동을 유도한다. 클래스는 객체 생성에 반복해서 사용될 뿐 아니라 상속과 포함을 통해서도 재사용할 수 있다.

NoteBase

### 다시 나오면 이렇게 판단

객체지향 선지는 클래스와 객체의 관계, 상속·재사용, 다형성, 추상화의 정의를 기준으로 판단한다. ‘클래스는 재사용할 수 없다’처럼 핵심 특성을 전면 부정하는 표현이 있으면 그 부분을 우선 확인한다.

### 암기 포인트

클래스는 객체의 설계도이자 재사용 단위이고, 오버라이딩·오버로딩은 다형성을 지원하며, 추상 메소드는 선언만 있고 구현은 없다.

NoteBase

10. 인터넷에서 사용되는 TCP(Transmission Control Protocol)에 대한 설명으로 옳지 않은 것은?

① 데이터를 송수신 하기 위해서는 송신측과 수신측이 서로 미리 연결을 맺어야 한다.
② 수신측의 버퍼가 오버플로우되지 않도록 데이터 흐름 제어(flow control)를 수행한다.
③ 네트워크 내 패킷 수가 과도하게 증가하는 현상을 방지하기 위하여 혼잡 제어(congestion control)를 수행한다.
④ 서버와 클라이언트 간 파일을 전송하기 위한 FTP(File Transfer Protocol)에서 이용한다.
⑤ 송신측에서는 수신측이 데이터를 받았는지 확인할 수 없다.

**정답 5번**

**출제 포인트**

TCP의 연결 지향성, 흐름 제어, 혼잡 제어, 신뢰성 보장과 TCP를 사용하는 응용 계층 프로토콜을 구별하는 문제이다.

**핵심 개념**

TCP는 송신 측과 수신 측이 연결을 설정한 뒤 데이터를 신뢰성 있게 전송하는 연결 지향형 전송 계층 프로토콜이다. 수신 측의 처리 능력에 맞추는 흐름 제어, 네트워크의 혼잡 상태에 맞추는 혼잡 제어, 확인 응답과 재전송을 이용한 오류 복구를 제공한다.

**선지별 해설**

① O — TCP는 연결 지향형 프로토콜이므로 데이터를 송수신하기 전에 송신 측과 수신 측이 논리적 연결을 설정한다.
② O — TCP의 흐름 제어는 수신 측이 처리할 수 있는 범위에 맞추어 송신량을 조절하므로 수신 버퍼의 오버플로를 방지한다.
③ O — TCP는 네트워크에 패킷이 지나치게 몰리는 혼잡을 줄이기 위해 혼잡 제어를 수행한다.
④ O — FTP는 서버와 클라이언트 사이에서 파일을 전송하는 응용 계층 프로토콜이며, 신뢰성 있는 전송을 위해 TCP를 사용한다.
⑤ X — 틀린 부분은 '송신측에서는 수신측이 데이터를 받았는지 확인할 수 없다'이다. 실제로 TCP 송신 측은 수신 측의 확인 응답을 통해 데이터 수신 여부를 확인하며, 필요한 경우 데이터를 재전송한다.

**오답 포인트**

⑤는 IP나 UDP처럼 전달 확인이 보장되지 않는 특성을 TCP에 잘못 적용한 선지이다. TCP의 대표 특징은 확인 응답과 재전송을 통한 신뢰성 있는 전송이다.

NoteBase

### 다시 나오면 이렇게 판단

TCP 선지는 연결 설정, 확인 응답, 재전송, 흐름 제어, 혼잡 제어라는 핵심 기능에 해당하는지 확인한다. 수신 여부를 확인할 수 없다고 하면 TCP의 신뢰성 보장과 모순되므로 틀린 설명이다.

### 암기 포인트

TCP는 연결하고, 확인하고, 필요하면 다시 보내며, 흐름과 혼잡을 제어한다.

NoteBase

**11. 직접 메모리 접근(Direct Memory Access, DMA)에 대한 설명으로 옳지 않은 것은?**

① 입출력 장치와 메인메모리 사이에 데이터를 전송하는 방식이다.
② 사이클 스틸링(cycle stealing)과 같은 의미이다.
③ 각 입출력 장치마다 DMA 제어기(DMA controller)가 한 개씩 사용된다.
④ CPU가 메인메모리에 접근하지 않는 동안에 데이터를 전송한다.
⑤ 데이터의 전송이 완료되면 DMA 제어기가 CPU로 인터럽트 신호를 전송한다.

**정답 3번**

### 출제 포인트

DMA에서 CPU, DMA 제어기, 입출력 장치와 메인메모리가 각각 어떤 역할을 하는지 묻는 문제이다.

### 핵심 개념

DMA는 CPU가 데이터 전송을 일일이 처리하지 않고 DMA 제어기가 입출력 장치와 메인메모리 사이의 블록 데이터를 직접 전송하는 방식이다. DMA 제어기는 전송 중 시스템 버스와 메모리 사이클을 사용하고, 전송이 끝나면 인터럽트로 CPU에 완료 사실을 알린다. 하나의 DMA 제어기는 여러 채널을 두어 여러 입출력 장치의 전송을 관리할 수 있다.

### 선지별 해설

① O — DMA는 입출력 장치와 메인메모리 사이에서 데이터를 직접 전송하여 CPU의 전송 부담을 줄이는 방식이다.
② O — 이 문항은 DMA 제어기가 CPU의 메모리 사이클을 잠시 넘겨받아 전송하는 사이클 스틸링 방식을 DMA의 특징과 같은 취지로 표현한 것이다. 엄밀히 말하면 사이클 스틸링은 DMA 자체의 동의어가 아니라 DMA 제어기가 시스템 버스를 획득하는 방식 중 하나이다.
③ X — 틀린 표현은 '각 입출력 장치마다 DMA 제어기가 한 개씩 사용된다'이다. 올바르게는 하나의 DMA 제어기가 여러 DMA 채널을 통해 여러 입출력 장치의 데이터 전송을 관리할 수 있다.
④ O — DMA 제어기가 메모리 사이클을 사용하는 동안에는 CPU가 메인메모리에 접근하지 못하며, 그 사이 DMA 제어기가 데이터를 전송한다.
⑤ O — DMA 전송이 완료되면 DMA 제어기는 CPU에 인터럽트 신호를 보내 전송 완료를 알린다.

NoteBase

### 오답 포인트

DMA가 입출력 장치의 전송을 담당한다는 사실을 '장치마다 별도의 DMA 제어기가 반드시 필요하다'고 확대 해석하기 쉽다. 핵심은 장치별 전용 제어기가 아니라, 여러 채널을 가진 DMA 제어기가 복수 장치의 전송을 관리할 수 있다는 점이다.

### 다시 나오면 이렇게 판단

DMA 문제에서는 전송 경로가 '입출력 장치↔메인메모리', 전송 담당이 'DMA 제어기', CPU의 역할이 '초기 설정과 완료 인터럽트 처리'인지 확인한다. '입출력 장치마다 DMA 제어기 한 개'처럼 일대일 구성을 단정하면 틀린 설명으로 판단한다.

### 암기 포인트

DMA는 CPU 대신 메모리로 직접 전송하고, 완료 후 CPU에 인터럽트로 알린다. DMA 제어기 하나가 여러 채널을 관리할 수 있다.

NoteBase

12. 4개의 512 × 4 비트 RAM을 직렬로 연결하여 구성한 메모리의 마지막 주소를 16진수로 표시한 것으로 옳은 것은?

① 1FF
② 200
③ 7FF
④ 800
⑤ FFF

**정답 3번**

**출제 포인트**

이 문제는 메모리 칩을 직렬로 연결하여 주소 공간을 확장했을 때 전체 기억장소 수와 마지막 주소를 구하는 문제이다.

**조건 정리**

주어진 값: 칩 1개의 구성은 512 × 4비트이고, 칩은 4개이다.
구해야 할 값: 전체 메모리의 마지막 주소이다.
주의할 조건: 직렬 연결은 워드 수, 즉 주소 수를 늘리며 각 주소에 저장되는 데이터 폭 4비트는 그대로이다.
주소는 0부터 시작하므로 마지막 주소는 전체 주소 수보다 1 작다.

**사용할 공식**

전체 주소 수=칩 1개의 주소 수×칩 수
마지막 주소=전체 주소 수-1
각 칩의 512개 주소 공간을 순서대로 이어 붙이는 구성이므로 전체 주소 수를 곱셈으로 구한다.

**계산 과정**

1. 전체 주소 수는 $512 \times 4 = 2048 = 2^{11}$개이다.
2. 주소 범위는 0부터 $2048 - 1 = 2047$까지이다.
3. 2047을 16진수로 바꾸면 $2047 = 7FF_{16}$이다. 따라서 마지막 주소는 $7FF$이다.

NoteBase

### 선지 판단

① X — 1*FF*는 16진수로 511이므로 칩 1개의 주소 범위인 0~1*FF*의 마지막 주소이다. 틀린 표현 "전체 메모리의 마지막 주소가 1*FF*" → 올바르게는 "칩 1개의 마지막 주소가 1*FF*"이다.
② X — 200은 16진수로 512이므로 두 번째 칩에 해당하는 주소 구간의 시작 주소이지 전체 메모리의 마지막 주소가 아니다. 틀린 표현 "마지막 주소가 200" → 올바르게는 "두 번째 512개 주소 구간의 시작 주소가 200"이다.
③ O — 전체 주소 수가 2048개이므로 주소 범위는 0~*7FF*이고, 마지막 주소는 *7FF*이다.
④ X — 800은 16진수로 2048이며 전체 주소 수를 나타내는 값이지만, 0부터 세면 마지막 주소 다음의 범위 밖 주소이다. 틀린 표현 "마지막 주소가 800" → 올바르게는 "전체 주소 수는 $800_{16}$개이고 마지막 주소는 *7FF*"이다.
⑤ X — *FFF*는 0부터 시작하는 4096개 주소 공간의 마지막 주소이므로 이 구성보다 주소 공간이 두 배 크다. 틀린 표현 "마지막 주소가 *FFF*" → 올바르게는 "2048개 주소 공간의 마지막 주소가 *7FF*"이다.
계산 결과 마지막 주소가 *7FF*이므로 정답은 3번이다.

### 오답 포인트

칩의 데이터 폭 4비트를 주소 수에 다시 곱하거나, 전체 주소 수 $800_{16}$을 마지막 주소로 착각하기 쉽다. 이 문제에서 직렬 연결은 주소 수만 512 × 4로 늘리며, 주소가 0부터 시작하므로 마지막 주소는 반드시 1을 뺀 *7FF*이다.

### 다시 나오면 이렇게 판단

메모리 주소 문제는 먼저 전체 워드 수를 구하고, 주소가 0부터 시작한다는 점을 반영하여 1을 뺀 뒤, 그 값을 요구된 진법으로 변환하는 순서로 판단한다.

### 암기 포인트

주소가 0부터 시작하면 *N*개 기억장소의 마지막 주소는 $N - 1$이다.

NoteBase

13. 인터넷에서 인접한 노드(node) 간의 프레임을 전송하며 목적지의 주소 지정, 흐름제어 및 오류 제어 기능을 제공하는 계층으로 옳은 것은?

① 물리 계층(physical layer)
② 데이터 링크 계층(data link layer)
③ 네트워크 계층(network layer)
④ 트랜스포트 계층(transport layer)
⑤ 응용 계층(application layer)

**정답 2번**

**출제 포인트**

인접한 노드 사이에서 프레임을 전송하고 물리 주소 지정, 흐름 제어, 오류 제어를 수행하는 계층을 구별하는 문제이다.

**핵심 개념**

데이터 링크 계층은 하나의 링크로 직접 연결된 인접 노드 사이에서 프레임 단위 전송을 담당하는 계층이다. MAC 주소와 같은 물리 주소를 사용하며, 프레임의 흐름 제어와 오류 검출·제어 기능을 제공한다.
물리 계층은 비트와 전기·기계적 신호, 네트워크 계층은 패킷과 논리 주소·경로 선택, 트랜스포트 계층은 종단 간 전송, 응용 계층은 사용자 응용 서비스를 담당한다.

NoteBase

### 선지별 해설

① X
틀린 표현은 '물리 계층'이다. 물리 계층은 전송 매체를 통해 비트 신호를 전달하며, 프레임 단위의 물리 주소 지정·흐름 제어·오류 제어를 담당하는 계층은 데이터 링크 계층이다.

② O
데이터 링크 계층은 인접 노드 사이에서 프레임을 전송하고 물리 주소 지정, 흐름 제어 및 오류 제어 기능을 제공한다.

③ X
틀린 표현은 '네트워크 계층'이다. 네트워크 계층은 패킷의 논리 주소 지정과 목적지까지의 경로 선택을 담당하며, 인접 노드 간 프레임 전송을 담당하는 계층은 데이터 링크 계층이다.

④ X
틀린 표현은 '트랜스포트 계층'이다. 트랜스포트 계층은 송신 호스트와 수신 호스트 사이의 종단 간 전송과 종단 간 흐름·오류 제어를 담당하며, 인접 노드 간 프레임 제어는 데이터 링크 계층의 기능이다.

⑤ X
틀린 표현은 '응용 계층'이다. 응용 계층은 사용자나 응용 프로그램에 네트워크 서비스를 제공하며, 인접 노드 간 프레임 전송을 담당하는 계층은 데이터 링크 계층이다.

### 오답 포인트

'흐름 제어와 오류 제어'라는 표현만 보고 트랜스포트 계층을 고르기 쉽다. 이 문항의 결정적 단서는 '인접한 노드 간'과 '프레임'이다. 데이터 링크 계층은 링크 단위로, 트랜스포트 계층은 종단 간으로 제어한다.

### 다시 나오면 이렇게 판단

전송 단위와 통신 범위를 함께 확인한다. 비트·신호는 물리 계층, 프레임·인접 노드는 데이터 링크 계층, 패킷·경로 선택은 네트워크 계층, 종단 간 전송은 트랜스포트 계층으로 판단한다.

### 암기 포인트

인접 노드 + 프레임 + MAC 주소 = 데이터 링크 계층이다.

NoteBase

14. 리눅스와 같은 운영체제에서 사용되는 프로세스(process)와 스레드(thread)에 대한 설명으로 옳지 않은 것은?

① 프로세스는 서로 다른 수의 스레드를 가질 수 있다.
② 다른 프로세스에 속한 스레드 간의 교착상태는 발생하지 않는다.
③ 한 프로세스에 속한 스레드들은 메모리를 공유한다.
④ 스레드 단위로 CPU 스케줄링이 일어난다.
⑤ 같은 프로세스에 속한 스레드로 문맥교환(context switching)하는 것이 다른 프로세스에 속한 스레드로 문맥교환하는 것보다 빠르다.

**정답 2번**

**출제 포인트**

이 문제는 프로세스와 스레드의 메모리 공유 범위, CPU 스케줄링 단위, 문맥교환 비용 및 교착상태의 발생 조건을 구분하는 문제이다.

**핵심 개념**

프로세스는 실행 중인 프로그램의 자원 소유 단위로서 독립된 주소 공간과 자원을 가진다. 스레드는 프로세스 내부의 실행 흐름으로서 같은 프로세스의 코드·데이터·힙 등의 주소 공간과 자원을 공유하지만, 레지스터와 스택 등 실행 상태는 각자 가진다.
교착상태는 여러 실행 주체가 서로 필요한 자원을 기다리며 진행하지 못하는 상태이다. 같은 프로세스의 스레드 사이뿐 아니라 서로 다른 프로세스나 그 소속 스레드가 공유 자원과 동기화 수단을 두고 경쟁할 때도 발생할 수 있다.

**선지별 해설**

① O — 하나의 프로세스는 한 개 이상의 스레드를 가질 수 있으므로 프로세스마다 스레드 수가 다를 수 있다.
② X — 틀린 표현은 '다른 프로세스에 속한 스레드 간의 교착상태는 발생하지 않는다'이다. 서로 다른 프로세스의 스레드도 파일, 입출력 장치, 공유 메모리의 잠금 등 공동 자원을 서로 점유하고 기다리면 교착상태에 빠질 수 있다. 올바르게는 '서로 다른 프로세스에 속한 스레드 간에도 공유 자원을 둘러싼 교착상태가 발생할 수 있다'이다.
③ O — 같은 프로세스의 스레드들은 그 프로세스의 주소 공간에 있는 코드·데이터·힙 등을 공유한다. 다만 각 스레드의 스택과 레지스터 같은 실행 상태는 독립적이다.
④ O — 스레드는 CPU가 실행할 작업 흐름이므로 스레드 단위로 CPU 스케줄링이 이루어진다.
⑤ O — 같은 프로세스의 스레드끼리 문맥교환할 때는 주소 공간을 그대로 공유하므로, 주소 공간과 메모리 관리 정보까지 바꾸어야 하는 서로 다른 프로세스 소속 스레드 간 문맥교환보다 일반적으로 빠르다.

NoteBase

### 오답 포인트

서로 다른 프로세스는 주소 공간이 분리된다는 사실을 교착상태가 불가능하다는 뜻으로 오해하기 쉽다. 주소 공간이 분리되어도 운영체제가 관리하는 파일·장치나 프로세스 간 공유 메모리 같은 자원을 함께 사용할 수 있으므로 교착상태가 발생할 수 있다.

### 다시 나오면 이렇게 판단

메모리 공유 여부와 교착상태 가능 여부를 별개로 판단한다. 같은 프로세스의 스레드는 주소 공간을 공유하고, 서로 다른 프로세스의 스레드라도 공동 자원을 서로 기다리는 구조라면 교착상태가 가능하다. 문맥교환 비용은 주소 공간까지 교체하는지를 기준으로 비교한다.

### 암기 포인트

같은 프로세스의 스레드는 메모리를 공유하고 문맥교환이 가볍다. 프로세스가 달라도 공유 자원을 기다리면 교착상태는 발생할 수 있다.

NoteBase

15. 현재 모바일 기기에 탑재되는 임베디드 데이터베이스 엔진의 대표적인 예로서, 서버가 따로 필요하지 않아 널리 활용되는 것으로 옳은 것은?

① SQLite
② MySQL
③ mSQL
④ Oracle
⑤ SQL Server

**정답 1번**

**출제 포인트**

모바일 기기에 탑재할 수 있고 별도의 데이터베이스 서버 프로세스 없이 동작하는 대표적인 임베디드 데이터베이스 엔진을 구별하는 문제이다.

**핵심 개념**

SQLite는 응용 프로그램에 라이브러리 형태로 포함되어 동작하는 경량 관계형 데이터베이스 엔진이다. 별도의 서버를 설치하거나 운영하지 않고 하나의 데이터베이스 파일을 직접 읽고 쓸 수 있어 모바일·임베디드 환경에서 널리 사용된다.
SQLite는 서버리스·임베디드·파일 기반이 핵심이고, MySQL·mSQL·Oracle·SQL Server는 일반적인 제품 구조에서 별도의 데이터베이스 서버에 접속하여 사용하는 서버형 DBMS이다.

**선지별 해설**

① O — SQLite는 응용 프로그램 내부에 포함되어 별도의 서버 프로세스 없이 데이터베이스 파일에 직접 접근하는 대표적인 임베디드 데이터베이스 엔진이다.
② X — MySQL은 일반적으로 MySQL 서버에 클라이언트가 접속하는 서버형 DBMS이다. 틀린 표현은 'MySQL'이며, 이 문항의 조건에 맞게 고치면 'SQLite'이다.
③ X — mSQL은 클라이언트가 데이터베이스 서버에 접속하는 서버형 SQL DBMS이므로 대표적인 모바일 임베디드 서버리스 엔진이 아니다. 틀린 표현은 'mSQL'이며, 올바르게는 'SQLite'이다.
④ X — Oracle은 일반적으로 별도의 데이터베이스 서버를 구축해 사용하는 서버형 DBMS이다. 틀린 표현은 'Oracle'이며, 별도 서버가 필요 없는 대표적 임베디드 엔진으로 고치면 'SQLite'이다.
⑤ X — SQL Server는 일반적으로 별도의 데이터베이스 서버 인스턴스를 운영하는 서버형 DBMS이다. 틀린 표현은 'SQL Server'이며, 이 문항의 설명에 맞는 표현은 'SQLite'이다.

NoteBase

### 오답 포인트

선지에 제시된 제품이 모두 관계형 데이터베이스 계열이라는 점만 보고 판단하면 혼동하기 쉽다. 핵심은 SQL 지원 여부가 아니라 '응용 프로그램에 내장되는가'와 '별도의 서버 프로세스가 필요한가'이다.

### 다시 나오면 이렇게 판단

문제에서 모바일·임베디드, 경량, 별도 서버 불필요, 단일 파일 같은 표현이 함께 나오면 SQLite를 우선 판단한다. 반대로 클라이언트가 별도의 데이터베이스 서버에 접속하는 구조이면 서버형 DBMS로 구분한다.

### 암기 포인트

SQLite는 서버 없이 앱 안에서 파일로 동작하는 대표적인 임베디드 DB이다.

NoteBase

**16. 뷰(view)에 대한 설명으로 옳지 않은 것은?**

① 뷰는 독자적인 인덱스를 가질 수 없다.
② 뷰의 정의를 변경하기 위해 ALTER 문을 사용할 수 있다.
③ 뷰의 정의만 시스템 카탈로그에 저장하였다가 필요시 실행시간에 테이블을 구축한다.
④ 데이터에 대한 보안을 제공한다.
⑤ 뷰는 또 다른 뷰의 정의에 사용될 수 있다.

**정답 2번**

**출제 포인트**

뷰의 저장 방식, 정의 변경 방법, 인덱스·보안·중첩 뷰의 특징을 구분하는 문제이다.

**핵심 개념**

뷰는 하나 이상의 기본 테이블이나 다른 뷰에 대한 질의로 정의되는 가상 테이블이다. 일반적인 뷰는 실제 데이터가 아니라 정의만 시스템 카탈로그에 저장되며, 참조할 때 질의를 실행하여 결과를 만든다. 뷰의 질의 정의를 바꾸려면 기존 뷰를 삭제한 뒤 다시 생성하는 것이 원칙이며, DBMS가 지원하면 CREATE OR REPLACE VIEW를 사용할 수 있다.

**선지별 해설**

① O — 일반적인 뷰는 데이터를 독립적으로 저장하는 실체 테이블이 아니므로 뷰 자체에 독자적인 인덱스를 가질 수 없다.

② X — 틀린 표현은 '뷰의 정의를 변경하기 위해 ALTER 문을 사용할 수 있다'이다. ALTER VIEW는 일반적으로 뷰를 이루는 SELECT 질의 자체를 변경하는 명령이 아니다. 올바르게는 기존 뷰를 삭제한 뒤 CREATE VIEW로 다시 정의하며, DBMS가 지원하면 CREATE OR REPLACE VIEW를 사용한다.

③ O — 일반적인 뷰에는 질의 정의만 시스템 카탈로그에 저장되고, 뷰가 참조될 때 그 정의를 실행하여 가상 테이블의 결과를 구성한다.

④ O — 사용자에게 필요한 행이나 열만 보이게 하고 기본 테이블에 대한 직접 접근을 제한할 수 있으므로 데이터 보안에 활용된다.

⑤ O — 뷰를 정의하는 질의의 FROM 절에서 다른 뷰를 참조할 수 있으므로 뷰를 바탕으로 또 다른 뷰를 정의할 수 있다.

NoteBase

**오답 포인트**

②의 ALTER라는 단어가 함정이다. 테이블 구조 변경에 ALTER TABLE을 쓰는 것과 달리, 뷰를 구성하는 SELECT 정의의 변경은 기존 뷰의 삭제 후 재생성 또는 DBMS가 지원하는 CREATE OR REPLACE VIEW로 처리한다.

**다시 나오면 이렇게 판단**

뷰 관련 선지는 '실제 데이터가 저장되는가', '정의만 저장되는가', '기본 테이블 접근을 제한하는가', '다른 뷰를 참조할 수 있는가'를 차례로 확인한다. 정의 변경을 ALTER로 단정하면 틀린 선지로 판단한다.

**암기 포인트**

뷰는 정의만 저장하는 가상 테이블이며, 정의 변경은 삭제 후 재생성이 원칙이다.

NoteBase

17. 중위 표기법(infix notation)으로 된 다음 식의 후위 표기법(postfix notation)으로 옳은 것은?

| (8 - 2 * 3) / 2 + 6 * 3 / 2 |
|---|

① 8 2 3 * - 2 / 6 3 * 2 / +
② 2 3 * 8 - 2 / 6 3 * 2 / +
③ 8 2 - 3 * 2 / 6 + 3 * 2 /
④ 8 2 3 * - 2 / 6 3 * 2 + /
⑤ + / - 8 * 2 3 2 / * 6 3 2

**정답 1번**

**출제 포인트**

이 문제는 연산자의 우선순위와 결합 순서를 보존하면서 중위 표기식을 후위 표기식으로 변환하는 원리를 묻는다.

**조건 정리**

주어진 식: (8 − 2 × 3)/2 + 6 × 3/2
구해야 할 것: 같은 연산 순서를 나타내는 후위 표기식
주의할 조건: 괄호 안을 먼저 처리하고, 곱셈·나눗셈을 덧셈·뺄셈보다 먼저 처리하며, 같은 우선순위의 곱셈·나눗셈은 왼쪽부터 결합한다.

**사용할 공식**

후위 표기식에서는 피연산자를 먼저 쓰고 그 뒤에 연산자를 쓴다. 즉, AopB는 *A B* op로 바뀐다. 이 원리를 쓰면 괄호나 우선순위 기호 없이도 원래 식의 계산 순서를 나타낼 수 있다.

**계산 과정**

1. 괄호 안의 8 − 2 × 3은 곱셈을 먼저 반영하여 8 2 3 × −가 된다.
2. 이를 2로 나누는 첫째 항은 8 2 3 × − 2 /가 된다.
3. 둘째 항 6 × 3/2는 왼쪽부터 계산하므로 6 3 × 2 /가 된다.
4. 두 항을 더하는 연산자 +를 맨 뒤에 붙이면 전체 식은 8 2 3 × − 2 / 6 3 × 2 / +가 된다. 실제 값도 첫째 항이 1, 둘째 항이 9이므로 전체는 10이다.

NoteBase

**선지 판단**

① O — 8 2 3 × − 2 /와 6 3 × 2 /를 차례로 적고 마지막에 +를 두었으므로 원래 식의 연산 순서와 일치한다.
② X — 앞부분의 2 3 × 8 −는 2 × 3 − 8을 뜻하여 뺄셈의 피연산자 순서가 뒤바뀌었다. 올바르게는 8 2 3 × −이다.
③ X — 앞부분의 8 2 − 3 ×는 (8 − 2) × 3을 뜻하므로 원래 식의 8 − (2 × 3)과 다르다. 올바르게는 8 2 3 × −로 시작하고, 두 항을 각각 완성한 뒤 마지막에 +를 둔다.
④ X — 끝부분의 6 3 × 2 + /는 2를 둘째 항의 나눗셈 피연산자로 처리하지 않고 앞 계산과 더한 뒤 나누게 한다. 올바른 끝부분은 6 3 × 2 / +이다.
⑤ X — 연산자 +, /, −가 필요한 피연산자보다 먼저 나와 후위 표기식의 스택 계산 규칙을 만족하지 못한다. 올바른 후위 표기식은 피연산자들을 먼저 두고 각 연산자를 뒤에 배치한 8 2 3 × − 2 / 6 3 × 2 / +이다.

**오답 포인트**

②와 ③은 뺄셈에서 피연산자의 순서와 곱셈 우선순위를 바꾸는 함정이다. ④는 둘째 항의 나눗셈을 끝내기 전에 덧셈을 배치한 함정이다. 후위 표기식에서는 교환법칙이 성립하지 않는 뺄셈과 나눗셈의 피연산자 순서를 특히 보존해야 한다.

**다시 나오면 이렇게 판단**

중위 표기식을 괄호와 우선순위에 따라 작은 부분식으로 나누고, 각 부분식을 '왼쪽 피연산자 → 오른쪽 피연산자 → 연산자' 순서로 바꾼 뒤 마지막 연산자를 맨 뒤에 붙인다. 완성된 후위 표기식을 왼쪽부터 스택으로 계산하여 원래 식의 값과 같은지도 검산한다.

**암기 포인트**

후위 표기법은 '피연산자 먼저, 연산자는 뒤'이며, 뺄셈과 나눗셈은 좌우 순서를 그대로 보존한다.

NoteBase

18. 다음 C 프로그램의 출력 결과로 옳은 것은?

```
#include <stdio.h>
int main()
{
    int i = 3;
    int j = 4;
    if((++i > j--) && (i++ < --j)) i = i-- + ++j;
    else j = i-- - --j;
    printf("%d\n", i);
}
```

① 1
② 2
③ 3
④ 4
⑤ 5

**정답 3번**

**출제 포인트**

이 문제는 C 언어의 전위·후위 증감 연산자와 논리 AND 연산자 &&의 단락 평가를 적용하여 변수 i의 최종값을 추적하는 문제이다.

**올바른 순서**

1. 처음에는 i=3, j=4이다.
2. ++i로 i가 먼저 4가 되고, j--는 비교에 기존 값 4를 사용한 뒤 j를 3으로 만든다. 따라서 ++i > j--는 4 > 4가 되어 거짓이다.
3. &&의 왼쪽 조건이 거짓이므로 오른쪽 조건 i++ < --j는 단락 평가되어 실행되지 않는다. 이 단계에서 i=4, j=3이다.
4. if 조건이 거짓이므로 else 문을 실행한다. i--는 계산에 4를 사용한 뒤 i를 3으로 만들고, --j는 j를 2로 만든 뒤 계산에 2를 사용한다. 따라서 j에는 4-2인 2가 저장된다.
5. printf는 j가 아니라 i를 출력하므로 최종 출력값은 3이다.

왜 이 순서인가: &&의 왼쪽 조건 결과가 오른쪽 조건의 실행 여부를 결정하며, 후위 감소는 현재 값을 사용한 뒤 변수값을 감소시키기 때문이다.

NoteBase

### 선지별 해설

① X — 출력값은 1이 아니라 3이다. else 문에서 i--가 계산에 4를 사용한 뒤 i를 3으로 만들며, printf는 이 i를 출력한다.
② X — 출력값은 2가 아니라 3이다. 2는 else 문 실행 후 j에 저장되는 값이고, 출력 대상은 i이다.
③ O — &&의 단락 평가로 두 번째 조건은 실행되지 않고, else 문의 i-- 처리 후 i가 3이 되므로 3이 출력된다.
④ X — 출력값은 4가 아니라 3이다. 첫 번째 조건 평가 직후 i는 4이지만, else 문의 i-- 때문에 출력 전에 3으로 감소한다.
⑤ X — 출력값은 5가 아니라 3이다. 두 번째 조건의 i++는 단락 평가로 실행되지 않으므로 i가 5가 되지 않으며, 최종 i는 3이다.

### 오답 포인트

첫째, 4 > 4는 거짓이므로 &&의 오른쪽 조건을 계산하지 않는다. 둘째, j--와 i--는 감소 전 값을 식에 사용하고 나서 변수값을 감소시킨다. 셋째, else 문에서 계산된 j=2를 출력값으로 착각하기 쉽지만 printf의 인수는 i이다.

### 다시 나오면 이렇게 판단

증감 연산자가 있는 조건식은 각 연산 직후의 변수값을 기록하고, &&에서는 왼쪽이 거짓이면 오른쪽을 실행하지 않는다는 단락 평가를 먼저 적용한다. 그다음 실제로 선택된 if 또는 else 문만 추적하고, 마지막으로 printf의 출력 변수를 확인한다.

### 암기 포인트

전위형은 먼저 증감한 값을 사용하고, 후위형은 현재 값을 사용한 뒤 증감한다. &&는 왼쪽이 거짓이면 오른쪽을 실행하지 않는다.

NoteBase

**19. 네트워크 장비에 대한 설명으로 옳지 않은 것은?**

① 허브(hub)는 여러 곳으로부터 들어온 데이터를 다른 여러 곳으로 보내는 역할을 하는 장비로, 더미 허브(dummy hub)와 스위칭 허브(switching hub)가 있다.

② 리피터(repeater)는 네트워크의 전송 거리를 연장하기 위하여 사용하는 장비로, 장거리 전송으로 인해 약해진 신호를 재생하여 전송해 준다.

③ 브리지(bridge)는 동일한 기관의 두 개 이상의 LAN의 분할된 세그먼트를 서로 연결하여 하나의 네트워크로 만드는 장비로, 네트워크에 흐르는 프레임의 물리주소를 필터링한다.

④ 게이트웨이(gateway)는 다른 네트워크로 들어가는 입구 역할을 하거나 나가는 출구 역할을 하는 장비로, 모뎀(modem)이 있다.

⑤ 라우터(router)는 LAN, MAN, WAN과 같은 네트워크를 서로 연결해 주는 장비로, 네트워크에 흐르는 패킷의 논리 주소(IP 주소)에 따라 패킷을 라우팅해 준다.

**정답 4번**

### 출제 포인트

허브, 리피터, 브리지, 게이트웨이, 라우터가 처리하는 정보와 수행하는 기능을 구별하는 문제이다.

### 핵심 개념

허브는 여러 장치를 한곳에 연결해 수신한 신호를 다른 포트로 전달하는 장비이다. 리피터는 약해진 신호를 재생해 전송 거리를 늘린다. 브리지는 MAC 주소를 기준으로 LAN 세그먼트 사이의 프레임을 걸러 전달한다. 게이트웨이는 서로 다른 네트워크나 프로토콜 체계 사이를 연결하고 변환한다. 라우터는 IP 주소를 기준으로 패킷의 전달 경로를 정한다. 모뎀은 디지털 신호와 아날로그 신호를 상호 변환하는 변복조 장치이다.

NoteBase

**선지별 해설**

① O
허브는 여러 포트를 연결하고 한 포트로 들어온 데이터를 다른 포트로 전달한다. 더미 허브는 수신 데이터를 여러 포트로 그대로 내보내고, 스위칭 허브는 MAC 주소를 이용해 필요한 포트로 선택적으로 전달한다.

② O
리피터는 장거리 전송 과정에서 감쇠되거나 왜곡된 신호를 재생하여 전송 가능 거리를 연장한다.

③ O
브리지는 같은 종류의 LAN을 여러 세그먼트로 나누거나 그 세그먼트들을 연결하며, 프레임의 물리 주소인 MAC 주소를 확인해 필요한 프레임만 통과시킨다.

④ X
틀린 표현은 게이트웨이의 예로 모뎀을 든 부분이다. 게이트웨이는 서로 다른 네트워크 또는 프로토콜 체계를 연결하고 필요한 변환을 수행하는 장비이고, 모뎀은 디지털 신호와 아날로그 신호를 상호 변환하는 변복조 장치이다. 올바르게는 모뎀을 게이트웨이의 한 종류로 볼 수 없다고 고쳐야 한다.

⑤ O
라우터는 LAN, MAN, WAN 등 서로 다른 네트워크를 연결하며, 패킷의 논리 주소인 IP 주소와 라우팅 정보를 바탕으로 전달 경로를 선택한다.

**오답 포인트**

④의 '다른 네트워크로 들어가는 입구나 나가는 출구'라는 일상적 표현만 보면 모뎀도 게이트웨이처럼 느껴질 수 있다. 그러나 장비는 위치가 아니라 핵심 기능으로 구별해야 한다. 게이트웨이의 핵심은 네트워크·프로토콜 간 연결과 변환이고, 모뎀의 핵심은 신호의 변조와 복조이다.

**다시 나오면 이렇게 판단**

장비가 무엇을 기준으로 처리하는지 확인한다. 신호 재생은 리피터, MAC 주소 기반 프레임 필터링은 브리지, IP 주소 기반 경로 선택은 라우터, 프로토콜 변환은 게이트웨이, 디지털·아날로그 신호 변환은 모뎀으로 판단한다.

**암기 포인트**

리피터는 신호, 브리지는 MAC, 라우터는 IP, 게이트웨이는 프로토콜, 모뎀은 변복조이다.

NoteBase

20. 다음 <보기>의 CPU 스케줄링 기법 중 기아(starvation) 현상이 발생할 수 있는 스케줄링 기법으로 옳은 것을 모두 고르면?

<보 기>

ㄱ. FIFO(First In First Out) 스케줄링
ㄴ. RR(Round Robin) 스케줄링
ㄷ. Priority 스케줄링
ㄹ. HRN(Highest Response ratio Next) 스케줄링
ㅁ. SJF(Shortest Job First) 스케줄링

① ㅁ
② ㄱ, ㄴ
③ ㄷ, ㅁ
④ ㄴ, ㄷ, ㄹ
⑤ ㄷ, ㄹ, ㅁ

**정답 3번**

**출제 포인트**

CPU 스케줄링에서 실행 기회를 계속 얻지 못하는 기아 현상이 발생할 수 있는 기법을 구별하는 문제이다.

**핵심 개념**

기아: 준비 큐에 있는 프로세스가 다른 프로세스에 계속 밀려 대기 시간이 무한히 길어질 수 있는 현상이다.
Priority 스케줄링: 우선순위가 높은 프로세스를 먼저 선택하므로 낮은 우선순위 프로세스에 기아가 생길 수 있다.
SJF 스케줄링: 실행 시간이 가장 짧은 작업을 먼저 선택하므로 짧은 작업이 계속 도착하면 긴 작업에 기아가 생길 수 있다.
FIFO는 도착 순서대로 처리하고, RR은 각 프로세스에 시간 할당량을 순환 배정하므로 통상적인 전제에서는 기아가 발생하지 않는다. HRN은 대기 시간이 길어질수록 응답비가 높아지므로 기아를 완화한다.

NoteBase

**선지별 해설**

① X — ㅁ(SJF)만 고른 점이 틀렸다. SJF뿐 아니라 ㄷ(Priority)에서도 낮은 우선순위 프로세스가 계속 밀려 기아가 발생할 수 있으므로 올바른 조합은 ㄷ, ㅁ이다.
② X — ㄱ(FIFO)과 ㄴ(RR)을 기아 발생 기법으로 본 점이 틀렸다. FIFO는 도착 순서를 보장하고 RR은 모든 준비 프로세스에 차례로 CPU 시간을 주므로 올바른 조합은 ㄷ, ㅁ이다.
③ O — ㄷ(Priority)은 낮은 우선순위 프로세스가, ㅁ(SJF)은 긴 실행 시간을 가진 프로세스가 계속 선택되지 못할 수 있으므로 두 기법 모두 기아가 발생할 수 있다.
④ X — ㄴ(RR)과 ㄹ(HRN)을 포함한 점이 틀렸다. RR은 순환 방식으로 실행 기회를 주며, HRN은 대기 시간을 응답비에 반영해 오래 기다린 작업의 선택 가능성을 높인다. 올바른 조합은 ㄷ, ㅁ이다.
⑤ X — ㄹ(HRN)을 포함한 점이 틀렸다. HRN은 대기 시간이 길수록 우선도가 높아져 기아를 완화하므로 올바른 조합은 ㄷ, ㅁ이다.

**오답 포인트**

SJF가 평균 대기 시간을 줄이는 효율적인 기법이라는 점과 기아 가능성을 혼동하기 쉽다. 또한 HRN의 이름에 우선 선택의 의미가 있어 Priority와 비슷해 보이지만, HRN은 대기 시간을 반영하여 SJF의 기아 문제를 보완한다.

**다시 나오면 이렇게 판단**

특정 프로세스가 새로 도착하는 유리한 작업에 계속 밀릴 수 있는지를 확인한다. 고정된 낮은 우선순위가 밀리면 Priority, 긴 작업이 짧은 작업에 밀리면 SJF이므로 두 기법을 기아 가능 기법으로 판단한다.

**암기 포인트**

기아 가능은 Priority와 SJF, 기아 보완은 aging과 HRN이다.

NoteBase