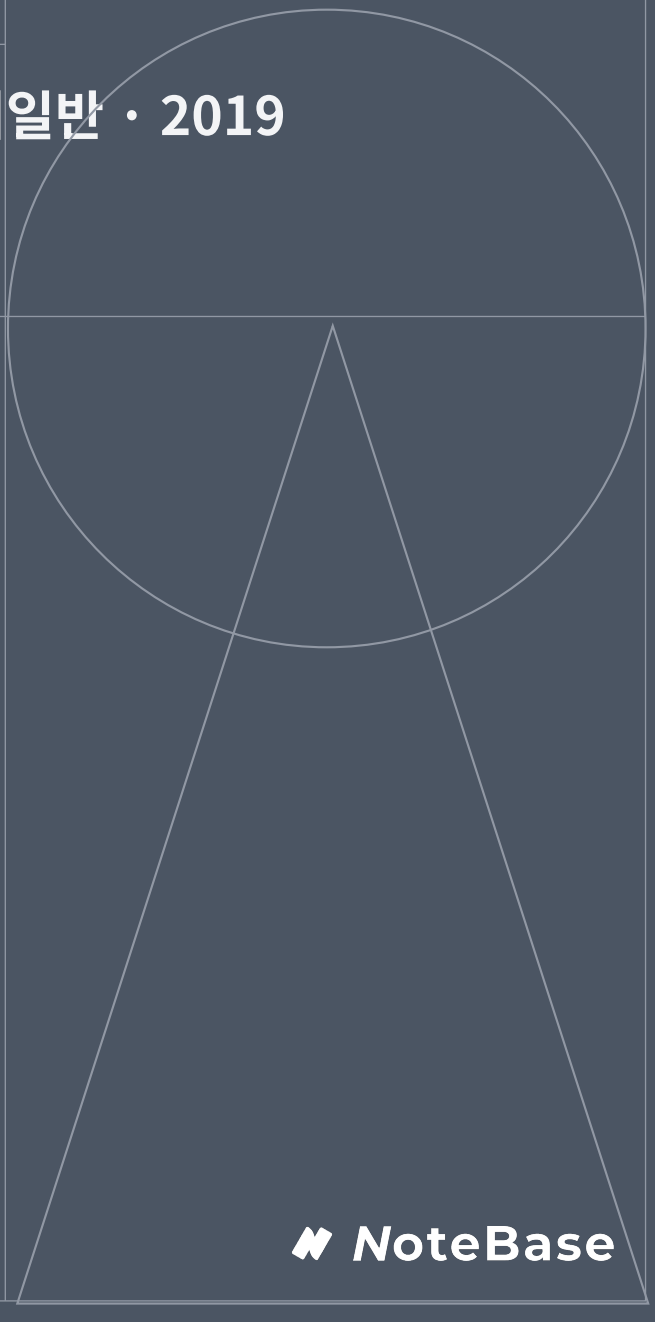
기출OX 해설집

공무원/계리직/9급 · 컴퓨터일반 · 2019



NoteBase

기출OX 해설집

공무원/계리직/9급 · 컴퓨터일반 · 2019

공무원/계리직/9급 · 컴퓨터일반 · 2019

문 1. 동기식 전송(Synchronous Transmission)에 대한 설명으로 옳지 않은 것은?

① 정해진 숫자만큼의 문자열을 묶어 일시에 전송한다.

② 작은 비트블록 앞뒤에 Start Bit와 Stop Bit를 삽입하여 비트블록을 동기화한다.

③ 2,400bps 이상 속도의 전송과 원거리 전송에 이용된다.

④ 블록과 블록 사이에 유휴시간(Idle Time)이 없어 전송효율이 높다.

정답 2번

출제 포인트

이 문제는 동기식 전송과 비동기식 전송을 구분하고, 동기식 전송의 특성이 아닌 것을 찾는 문제이다.

핵심 개념

동기식 전송은 송신 측과 수신 측이 동기를 맞춘 뒤 여러 문자를 블록이나 프레임 단위로 묶어 연속 전송하는 방식이다. 문자마다 Start Bit와 Stop Bit를 붙이지 않으므로 전송 효율이 높고 고속·원거리 전송에 적합하다. 비동기식 전송은 각 문자 앞뒤에 Start Bit와 Stop Bit를 붙여 문자 단위로 동기를 맞추는 방식이다. 문자 사이에 휴지 시간이 생길 수 있고 부가 비트로 인한 오버헤드가 크다.

선지별 해설

① O — 동기식 전송은 정해진 수의 문자를 블록이나 프레임으로 묶어 연속적으로 전송한다.

② X — 틀린 표현은 '작은 비트블록 앞뒤에 Start Bit와 Stop Bit를 삽입'한다는 부분이다. 이는 비동기식 전송의 특징이다. 올바르게는 동기식 전송은 문자마다 Start Bit와 Stop Bit를 붙이지 않고, 송수신 측의 동기를 맞춘 뒤 블록이나 프레임 단위로 연속 전송한다.

③ O — 동기식 전송은 문자마다 시작·정지 비트를 덧붙이는 부담이 작아 일반적으로 2,400*bps* 이상의 고속 전송과 원거리 전송에 이용된다.

④ O — 동기식 전송은 블록을 연속적으로 보내므로 블록 사이의 휴지 시간이 없고, 문자마다 시작·정지 비트를 붙이지 않아 전송 효율이 높다.

NoteBase

오답 포인트

②의 Start Bit와 Stop Bit라는 용어 때문에 '동기화'라는 말만 보고 동기식 전송으로 착각하기 쉽다. Start Bit와 Stop Bit를 각 문자 앞뒤에 붙이는 방식은 이름과 달리 비동기식 전송이다.

다시 나오면 이렇게 판단

문자마다 Start Bit와 Stop Bit를 붙이면 비동기식, 여러 문자를 블록이나 프레임으로 묶어 연속 전송하면 동기식으로 판단한다.

암기 포인트

동기식은 블록·연속·고효율, 비동기식은 문자별 Start/Stop이다.

NoteBase

문 2. 어떤 프로젝트를 완성하기 위해 작업 분할(Work Breakdown)을 통해 파악된, 다음 소작업(activity) 목록을 AOE(Activity On Edge) 네트워크로 표현하였을 때, 이 프로젝트가 끝날 수 있는 가장 빠른 소요시간은?

소작업 이름	소요시간	선행 소작업
a	5	없음
b	5	없음
c	8	a, b
d	2	c
e	3	b, c
f	4	d
g	5	e, f

① 13 ② 21 ③ 24 ④ 32

정답 3번

출제 포인트

이 문제는 각 소작업의 선행관계를 반영하여 AOE 네트워크의 최조 시작시간과 최조 완료시간을 구하는 문제이다.

조건 정리

주어진 값: a=5, b=5, c=8, d=2, e=3, f=4, g=5이다.
구해야 할 값: 모든 소작업을 마치고 프로젝트가 끝날 수 있는 가장 빠른 시간이다.
주의할 조건: 선행 소작업이 여러 개인 작업은 그 선행 작업이 모두 끝난 뒤 시작하므로, 선행 작업들의 완료시간 중 최댓값을 사용한다.

사용할 공식

작업의 최조 완료시간은 다음과 같다.
$EF = ES + t$
선행 작업이 없는 경우 $ES = 0$이고, 선행 작업이 있는 경우에는 다음을 적용한다.
ES=(각 선행 작업의 EF)
AOE 네트워크에서는 모든 선행 작업이 완료되어야 후속 작업을 시작할 수 있으므로 이 공식을 사용한다.

NoteBase

계산 과정

1. a와 b는 선행 작업이 없으므로 각각 0에 시작하여 5에 완료된다.
2. c는 a와 b가 모두 끝난 뒤 시작하므로 $ES_c = \max(5, 5) = 5$, $EF_c = 5 + 8 = 13$이다.
3. d는 c가 끝난 뒤 시작하므로 $EF_d = 13 + 2 = 15$이다.
4. e는 b와 c가 모두 끝난 뒤 시작하므로 $ES_e = \max(5, 13) = 13$, $EF_e = 13 + 3 = 16$이다.
5. f는 d가 끝난 뒤 시작하므로 $EF_f = 15 + 4 = 19$이다.
6. g는 e와 f가 모두 끝난 뒤 시작하므로 $ES_g = \max(16, 19) = 19$, $EF_g = 19 + 5 = 24$이다. 따라서 프로젝트의 최단 소요시간은 24이다.

선지 판단

① X — 13은 c의 최조 완료시간일 뿐 프로젝트 완료시간이 아니다. 틀린 값 13은 올바르게는 프로젝트 완료시간 24로 고쳐야 한다.
② X — 21은 c-e-g 경로의 시간인 5 + 8 + 3 + 5 = 21만 더한 값으로, 더 늦게 끝나는 f를 기다려야 한다는 조건을 반영하지 못한다. 틀린 값 21은 올바르게는 24로 고쳐야 한다.
③ O — a-c-d-f-g 또는 b-c-d-f-g의 소요시간이 5 + 8 + 2 + 4 + 5 = 24이고, g는 e와 f 중 늦게 완료되는 f 이후에 시작하므로 프로젝트의 최단 소요시간은 24이다.
④ X — 32는 동시에 진행할 수 있는 작업까지 순차적으로 더한 값이어서 최조 완료시간이 아니다. 틀린 값 32는 올바르게는 24로 고쳐야 한다.

오답 포인트

②의 21은 e가 16에 끝난 뒤 곧바로 g를 시작한다고 본 값이다. 그러나 g의 선행 작업은 e와 f이므로, 둘 중 늦은 f의 완료시간 19까지 기다려야 한다. 또한 병렬 수행 가능한 작업의 시간을 모두 합산하면 ④의 32처럼 실제보다 큰 값이 나온다.

다시 나오면 이렇게 판단

선행 작업이 없는 작업부터 최조 완료시간을 순방향으로 계산한다. 선행 작업이 둘 이상이면 완료시간의 합이 아니라 최댓값을 다음 작업의 시작시간으로 삼고, 마지막 작업의 완료시간을 프로젝트 소요시간으로 판단한다.

암기 포인트

여러 선행 작업이 합류하면 모두 끝나야 하므로 합류점에서는 완료시간의 최댓값을 선택한다.

NoteBase

문 3. 다음에 제시된 입력 데이터를 엑셀 서식의 표시 형식 코드에 따라 출력한 결과로 옳은 것은?

입력 데이터 : 1234.5 표시 형식 코드 : #,##0

① 1,234 ② 1,235 ③ 1,234.5 ④ 1,234.50

정답 2번

출제 포인트

이 문제는 엑셀 사용자 지정 표시 형식에서 자릿수 표시 기호와 천 단위 구분 기호가 실제 표시값을 어떻게 바꾸는지 묻는 문제이다.

조건 정리

주어진 값: 1234.5
적용할 표시 형식: #,##0
구해야 할 값: 셀에 표시되는 결과
주의할 조건: 마지막 0은 일의 자리까지 반드시 표시하므로 소수 부분은 반올림되며, 쉼표는 천 단위 구분 기호로 표시된다.

사용할 원리

#,##0에서 #은 필요한 숫자만 표시하고, 쉼표는 천 단위를 구분하며, 0은 일의 자리 숫자를 반드시 표시한다. 소수점 아래 자리 표시 기호가 없으므로 값을 정수로 반올림해 표시한다. 이 원리를 쓰는 이유는 표시 형식이 원래 값을 바꾸는 것이 아니라 화면에 보이는 자릿수와 구분 기호를 결정하기 때문이다.

계산 과정

1. 1234.5를 소수점 아래 0자리로 표시하므로 일의 자리까지 반올림한다.
2. 반올림 결과는 1235이다.
3. 천 단위 구분 기호를 넣으면 1,235가 된다.

1234.5 → 1,235

NoteBase

선지 판단

① X — 1,234는 소수 부분을 버림한 값이므로 틀리다. 올바르게는 1234.5를 일의 자리까지 반올림한 1,235이다.
② O — 일의 자리까지 반올림한 1235에 천 단위 쉼표를 표시한 1,235와 일치한다.
③ X — 1,234.5는 원래 소수 한 자리를 그대로 표시한 값이므로 틀리다. 올바르게는 소수 자릿수를 표시하지 않고 반올림한 1,235이다.
④ X — 1,234.50은 소수 둘째 자리까지 표시한 값이므로 틀리다. 올바르게는 소수 자릿수 없이 1,235로 표시된다.

오답 포인트

①의 1,234는 표시 형식에 소수점이 없으면 소수 부분을 무조건 버린다고 착각할 때 고르는 값이다. 엑셀의 숫자 표시 형식은 지정된 소수 자릿수에 맞추어 반올림한다. ③과 ④는 입력값의 소수 자릿수가 그대로 유지되거나 0으로 채워진다고 오해한 경우이다.

다시 나오면 이렇게 판단

숫자 표시 형식 문제는 소수점 아래 자리 표시 기호의 개수를 먼저 확인하고, 그 자릿수에 맞게 반올림한 뒤, 쉼표 등 구분 기호를 적용하는 순서로 판단한다.

암기 포인트

#,##0은 '천 단위 쉼표 + 소수 0자리 반올림' 형식이다.

NoteBase

문 4. 객체지향 소프트웨어 개발 및 UML Diagram에 대한 설명이다. ㉠~㉢에 들어갈 내용을 바르게 짝지은 것은?

- (㉠)은/는 외부에서 인식할 수 있는 특성이 담긴 소프트웨어의 골격이 되는 기본 구조로, 시스템 전체에 대한 큰 밑그림이다. 소프트웨어 품질 요구 사항은 (㉠)을/를 결정하는 데 주요한 요소로 작용한다.
- (㉡)은/는 두 개 이상의 클래스에서 동일한 메시지에 대해 객체가 다르게 반응하는 것이다.
- (㉢)은/는 객체 간의 메시지 통신을 분석하기 위한 것으로 시스템의 동작을 정형화하고 객체들의 메시지 교환을 시각화한다.

	㉠	㉡	㉢
①	소프트웨어 아키텍처	다형성	시퀀스 모델
②	유스케이스	다형성	시퀀스 모델
③	클래스 다이어그램	캡슐화	상태 모델
④	디자인 패턴	캡슐화	상태 모델

정답 1번

출제 포인트

소프트웨어 아키텍처, 다형성, 시퀀스 모델의 정의를 설명과 정확히 연결할 수 있는지를 묻는 문제이다.

핵심 개념

소프트웨어 아키텍처: 외부에서 인식할 수 있는 특성을 지닌 소프트웨어의 골격이 되는 기본 구조이며, 시스템 전체의 큰 밑그림과 품질 속성을 결정하는 데 중요하다.
다형성: 서로 다른 클래스의 객체가 동일한 메시지에 각자의 방식으로 반응하는 객체지향 특성이다.
시퀀스 모델: 시간의 흐름에 따른 객체 간 메시지 교환을 나타내어 시스템의 동작을 정형화하고 시각화하는 동적 모델이다.
유스케이스는 사용자와 시스템의 상호작용 및 기능 요구를 나타내고, 클래스 다이어그램은 클래스의 속성·연산·관계 등 정적 구조를 나타낸다. 캡슐화는 데이터와 구현을 감추고 정해진 인터페이스를 통해 접근하게 하는 특성이며, 상태 모델은 한 객체의 상태와 상태 전이를 중심으로 동작을 나타낸다.

NoteBase

선지별 해설

① O — ㉠ 소프트웨어 아키텍처, ㉡ 다형성, ㉢ 시퀀스 모델이 세 설명과 모두 일치한다.
② X — ㉠의 '유스케이스'가 틀렸다. 유스케이스는 사용자 관점의 기능과 상호작용을 표현하며, 시스템의 골격과 품질 요구 사항에 큰 영향을 주는 큰 밑그림은 '소프트웨어 아키텍처'이다. 따라서 올바른 짝은 '소프트웨어 아키텍처－다형성－시퀀스 모델'이다.
③ X — ㉠의 '클래스 다이어그램', ㉡의 '캡슐화', ㉢의 '상태 모델'이 모두 틀렸다. 클래스 다이어그램은 정적 구조, 캡슐화는 데이터와 구현의 은닉, 상태 모델은 객체의 상태와 전이를 나타낸다. 올바르게는 각각 '소프트웨어 아키텍처－다형성－시퀀스 모델'이다.
④ X — ㉠의 '디자인 패턴', ㉡의 '캡슐화', ㉢의 '상태 모델'이 모두 틀렸다. 디자인 패턴은 반복되는 설계 문제에 대한 재사용 가능한 해결 방식이고, 캡슐화와 상태 모델의 의미도 제시문의 설명과 다르다. 올바르게는 각각 '소프트웨어 아키텍처－다형성－시퀀스 모델'이다.

오답 포인트

객체지향과 UML에 함께 등장하는 용어라는 이유만으로 캡슐화와 다형성, 상태 모델과 시퀀스 모델을 바꾸어 고르기 쉽다. '동일한 메시지에 서로 다르게 반응'은 다형성이고, '객체 간 메시지 교환을 시간 순서로 표현'은 시퀀스 모델이라는 문구가 결정적이다.

다시 나오면 이렇게 판단

설명의 핵심 동사를 기준으로 구분한다. 시스템의 골격과 품질 속성 결정은 소프트웨어 아키텍처, 같은 메시지에 다른 반응은 다형성, 시간 순서에 따른 객체 간 메시지 교환은 시퀀스 모델로 판단한다.

암기 포인트

골격은 아키텍처, 같은 메시지·다른 반응은 다형성, 시간순 메시지 교환은 시퀀스 모델이다.

NoteBase

문 5. 공개키 암호방식에 대한 설명으로 옳은 것은?

① 송신자는 전송메시지에 대한 MAC(Message Authentication Code)을 생성하고 수신자는 그 MAC을 점검함으로써 메시지가 전송과정에서 변조되었는지 여부를 확인한다.

② 송신자는 수신자의 개인키를 이용하여 암호화한 메시지를 송신하고 수신자는 수신한 메시지를 자신의 공개키를 이용하여 복호화한다.

③ 송수신자 규모가 동일할 경우, 공개키 암호방식이 대칭키 암호방식보다 더 많은 키들을 필요로 하기 때문에 인증기관이 키 관리를 담당한다.

④ 키 운영의 신뢰성을 공식적으로 제공하기 위하여 인증기관은 고객별로 개인키와 키 소유자 정보를 만들고 이를 해당 고객에게 인증서로 제공한다.

정답 1번

출제 포인트

공개키 암호방식에서 암호화·복호화에 쓰는 키의 관계와 MAC 및 인증기관의 역할을 구분하는 문제이다.

핵심 개념

공개키 암호방식: 공개키와 개인키를 한 쌍으로 사용하며, 기밀성을 위한 암호화에서는 수신자의 공개키로 암호화하고 수신자의 개인키로 복호화한다.
MAC: 송신자와 수신자가 공유한 비밀키와 메시지를 이용해 생성·검증하는 인증값으로, 메시지의 무결성과 송신자 인증을 확인한다.
인증기관(CA): 공개키와 그 소유자의 신원 정보를 결합해 전자서명한 인증서를 발급함으로써 공개키의 신뢰성을 보증한다. 개인키 자체를 인증서에 넣어 제공하지 않는다.

NoteBase

선지별 해설

① O — 송신자가 메시지에 대한 MAC을 생성하고 수신자가 이를 검증하면 전송 중 메시지가 변조되었는지 확인할 수 있다.
② X — 틀린 표현은 '수신자의 개인키로 암호화'하고 '자신의 공개키로 복호화'한다는 부분이다. 기밀성을 위한 공개키 암호에서는 송신자가 수신자의 공개키로 암호화하고, 수신자가 자신의 개인키로 복호화한다.
③ X — 틀린 표현은 공개키 암호방식이 대칭키 암호방식보다 더 많은 키를 필요로 한다는 부분이다. 일반적으로 다수 사용자가 서로 통신할 때 공개키 암호방식은 사용자마다 공개키·개인키 한 쌍을 사용하므로, 통신 상대마다 별도의 비밀키를 안전하게 공유해야 하는 대칭키 방식보다 키 분배와 관리 부담을 줄일 수 있다. 인증기관은 키 개수가 많아서가 아니라 공개키가 실제 소유자의 것인지 보증하기 위해 인증서를 발급한다.
④ X — 틀린 표현은 인증기관이 고객별로 '개인키'를 만들어 인증서로 제공한다는 부분이다. 인증서는 고객의 공개키와 키 소유자 정보를 결합해 인증기관이 전자서명한 것이며, 개인키는 소유자가 비밀로 보관한다.

오답 포인트

②는 공개키와 개인키의 사용 주체를 서로 뒤바꾸고, ④는 인증서에 포함되는 공개키를 개인키로 바꾸어 놓았다. ③에서는 인증기관의 존재 이유를 키의 개수 문제로 오해하기 쉽지만, 핵심은 공개키와 소유자 신원의 결합을 신뢰할 수 있게 보증하는 데 있다.

다시 나오면 이렇게 판단

기밀성 문제에서는 '수신자의 공개키로 암호화 → 수신자의 개인키로 복호화'를 먼저 확인한다. 인증서 문제에서는 '소유자 정보 + 공개키 + 인증기관의 전자서명'을 기준으로 판단하고, 개인키가 인증서에 포함되거나 공개된다는 설명은 틀린 것으로 본다.

암기 포인트

암호화는 수신자 공개키, 복호화는 수신자 개인키이며, 인증서는 공개키의 신분증이다.

NoteBase

문 6. 온라인에서 멀티미디어 콘텐츠의 불법 유통을 방지하기 위해 삽입된 워터마킹 기술의 특성으로 옳지 않은 것은?

① 부인 방지성 ② 비가시성

③ 강인성 ④ 권리정보 추출성

정답 1번

출제 포인트

디지털 워터마킹이 갖추어야 할 비가시성, 강인성, 권리정보 추출성과 다른 보안 기술의 특성인 부인 방지성을 구별하는 문제이다.

핵심 개념

디지털 워터마킹은 저작권자나 이용 정보를 멀티미디어 콘텐츠에 눈에 잘 띄지 않도록 삽입하고, 필요할 때 이를 검출하거나 추출하는 기술이다. 비가시성은 콘텐츠의 품질을 해치지 않는 성질, 강인성은 변형이나 공격 후에도 워터마크가 유지되는 성질, 권리정보 추출성은 삽입된 소유권 정보를 검출·추출할 수 있는 성질이다. 부인 방지성은 행위자가 자신의 전송이나 서명 사실을 나중에 부인하지 못하게 하는 성질로, 주로 디지털 서명과 관련된다.

선지별 해설

① X — '부인 방지성'은 전송이나 서명 행위를 나중에 부인하지 못하게 하는 디지털 서명의 대표적 성질이며, 워터마킹의 고유 특성으로 보지 않는다. 올바르게는 워터마킹의 주요 특성으로 비가시성, 강인성, 권리정보 추출성 등을 제시해야 한다.

② O — 비가시성은 워터마크를 삽입해도 사람이 콘텐츠의 품질 변화를 쉽게 알아차리지 못하게 하는 특성이다.

③ O — 강인성은 압축, 편집, 변형이나 공격이 가해져도 삽입된 워터마크가 쉽게 제거되지 않고 남는 특성이다.

④ O — 권리정보 추출성은 콘텐츠에 삽입된 저작권자·소유권 정보를 필요할 때 검출하거나 추출할 수 있는 특성이다.

오답 포인트

①의 '부인 방지성'도 정보보안에서 쓰이는 용어이므로 워터마킹의 특성처럼 보이기 쉽다. 그러나 부인 방지는 디지털 서명의 기능이고, 워터마킹은 표시가 드러나지 않는지, 변형에도 남는지, 권리정보를 찾아낼 수 있는지가 핵심이다.

다시 나오면 이렇게 판단

워터마킹의 특성은 '눈에 띄지 않게 삽입되는가', '공격이나 변형에도 남는가', '권리정보를 검출·추출할 수 있는가'로 판단한다. 전송·서명 사실의 사후 부인을 막는다는 설명은 디지털 서명에 해당한다.

NoteBase

암기 포인트

워터마킹은 비가시성·강인성·추출성, 디지털 서명은 부인 방지성이다.

NoteBase

문 7. 프로세스 관리 과정에서 발생할 수 있는 교착상태(Deadlock)를 예방하기 위한 조치로 옳은 것은?

① 상호배제(Mutual Exclusion) 조건을 제거하고자 할 경우, 프로세스 A가 점유하고 있던 자원에 대하여 프로세스 B로부터 할당 요청이 있을 때 프로세스 B에게도 해당자원을 할당하여 준다. 운영체제는 프로세스 A와 프로세스 B가 종료되는 시점에서 일관성을 점검하여 프로세스 A와 프로세스 B 중 하나를 철회시킨다.

② 점유대기(Hold and Wait) 조건을 제거하고자 할 경우, 자원을 점유한 프로세스가 다른 자원을 요청하였지만 할당받지 못하면 일단 자신이 점유한 자원을 반납한다. 이후 그 프로세스는 반납하였던 자원과 요청하였던 자원을 함께 요청한다.

③ 비선점(No Preemption) 조건을 제거하고자 할 경우, 프로세스는 시작시점에서 자신이 사용할 모든 자원들에 대하여 일괄할당을 요청한다. 일괄할당이 이루어지지 않을 경우, 일괄할당이 이루어지기까지 지연됨에 따른 성능저하가 발생할 수 있다.

④ 환형대기(Circular Wait) 조건을 제거하고자 할 경우, 자원들의 할당순서를 정한다. 자원 R_i가 자원 R_k보다 먼저 할당되는 것으로 정하였을 경우, 프로세스 A가 R_i를 할당받은 후 R_k를 요청한 상태에서 프로세스 B가 R_k를 할당받은 후 R_i를 요청하면 교착상태가 발생하므로 운영체제는 프로세스 B의 자원요청을 거부한다.

정답 4번

출제 포인트

교착상태의 네 가지 필요조건인 상호배제, 점유대기, 비선점, 환형대기 중 하나가 성립하지 않도록 만드는 예방 방법을 구별하는 문제이다.

NoteBase

핵심 개념

교착상태 예방은 네 가지 필요조건 중 적어도 하나를 구조적으로 제거하는 방법이다.
상호배제 제거: 공유 가능한 자원은 동시에 사용하게 하지만, 프린터처럼 본질적으로 공유할 수 없는 자원에는 적용하기 어렵다.
점유대기 제거: 프로세스가 필요한 자원을 처음부터 한꺼번에 요청하게 하거나, 아무 자원도 보유하지 않은 상태에서만 새 자원을 요청하게 한다.
비선점 제거: 추가 자원을 즉시 받을 수 없으면 보유 자원을 반납하게 하는 등 자원 회수가 가능하도록 한다.
환형대기 제거: 모든 자원 유형에 순서를 부여하고 프로세스가 그 순서에 따라 자원을 요청하게 한다.

선지별 해설

① X
상호배제를 없앤다는 이유로 동일 자원을 A와 B에 중복 할당한 뒤 종료 시 일관성을 검사하여 한 프로세스를 철회시키는 것은 올바른 예방 방법이 아니다. 틀린 표현은 ‘B에게도 해당 자원을 할당’하고 ‘종료 시점에 일관성을 점검’한다는 부분이다. 올바르게는 공유 가능한 자원만 동시 사용하게 하고, 공유 불가능한 자원은 한 번에 한 프로세스에만 할당해야 한다.

② X
이 선지는 다른 자원의 할당이 실패한 뒤에야 보유 자원을 반환한다고 서술한다. 그러면 추가 자원을 요청하는 순간에는 자원을 보유한 채 요청하는 형태가 된다. 틀린 표현은 ‘할당받지 못하면 일단 자신이 점유한 자원을 반환’한다는 사후 처리이다. 점유대기를 구조적으로 제거하려면 실행 전에 필요한 자원을 모두 한꺼번에 요청하거나, 추가 자원을 요청하기 전에 보유 자원을 모두 반환하여 아무 자원도 점유하지 않은 상태에서 필요한 자원을 함께 요청해야 한다.

③ X
시작 시점에 필요한 모든 자원을 일괄 요청하는 방법은 비선점 조건이 아니라 점유대기 조건을 제거하는 방법이다. 틀린 표현은 이를 ‘비선점 조건을 제거’하는 조치라고 한 부분이다. 올바르게는 비선점을 제거하려면 자원 요청이 충족되지 않을 때 보유 자원을 회수하거나 반납할 수 있게 해야 한다. 또한 일괄 할당 방식의 대표적인 단점은 자원 이용률 저하와 기아 가능성이다.

④ O
자원 유형에 전역적인 할당 순서를 두고 그 순서대로만 요청하게 하면 환형대기를 제거할 수 있다. R_i가 R_k보다 먼저인 순서에서 A가 R_i를 받은 뒤 R_k를 요청하는 것은 순서를 지키지만, B가 R_k를 받은 뒤 더 앞선 R_i를 요청하는 것은 순서를 거스른다. 따라서 운영체제가 B의 R_i 요청을 거부하는 조치는 옳다.

NoteBase

오답 포인트

②와 ③은 '자원을 한꺼번에 요청하거나 반납한다'는 표현 때문에 그럴듯하다. 그러나 ②는 추가 자원을 요청한 뒤 실패하면 반환하는 사후 처리로 서술되어 있고, ③의 일괄 요청은 비선점이 아니라 점유대기를 없애는 방법이다. ④는 자원의 대소·선후 순서를 요청 순서와 정확히 연결한 선지이다.

다시 나오면 이렇게 판단

예방 조치가 어느 조건을 끊는지 대응시킨다. 동시 공유는 상호배제, 무보유 상태에서의 일괄 요청은 점유대기, 강제 회수·반납은 비선점, 자원 번호의 단조로운 요청은 환형대기를 제거한다.

암기 포인트

공유-상호배제, 일괄 요청-점유대기, 회수-비선점, 자원 순서-환형대기이다.

NoteBase

문 8. 순차 파일과 인덱스 순차 파일에 대한 설명으로 옳은 것의 총 개수는?

ㄱ. 순차 파일에서의 데이터 레코드 증가는 적용된 순차 기준으로 마지막 위치에서 이루어진다.
ㄴ. 순차 파일에서는 접근 조건으로 제시된 순차 대상 필드 값 범위에 해당하는 대량의 데이터 레코드들을 접근할 때 효과적이다.
ㄷ. 순차 파일에서의 데이터 레코드 증가는 오버플로우 블록을 생성시키지 않는다.
ㄹ. 인덱스 순차 파일의 인덱스에는 인덱스 대상 필드 값과 그 값을 가지는 데이터 레코드를 접근할 수 있게 하는 위치 값이 기록된다.
ㅁ. 인덱스 순차 파일에서는 인덱스 갱신없이 데이터 레코드를 추가하거나 삭제하는 것이 가능하다.
ㅂ. 인덱스 순차 파일에서는 접근 조건에 해당하는 인덱스 대상 필드 값을 가지는 소량의 데이터 레코드를 순차 파일보다 효과적으로 접근할 수 있다.
ㅅ. 인덱스를 다중레벨로 구성할 경우, 최하위 레벨은 순차 파일 형식으로 구성된다.

① 2개 ② 3개 ③ 4개 ④ 5개

정답 3번

출제 포인트

순차 파일과 인덱스 순차 파일의 저장 순서, 범위 접근, 레코드 증감, 인덱스 구조의 차이를 구분하는 문제이다.

NoteBase

기출OX 해설집 · 공무원/계리직/9급 · 컴퓨터일반 · 2019

핵심 개념

순차 파일: 레코드를 정해진 순차 기준에 따라 물리적으로 배열하여 차례로 접근하는 파일이다. 같은 기준의 넓은 범위를 연속해서 처리하는 데 유리하지만, 중간 삽입 시 순서를 유지해야 하므로 빈 공간이 없으면 오버플로우 블록을 사용하고 이후 재편성할 수 있다.
인덱스 순차 파일: 레코드는 키 순서로 저장하고, 별도의 인덱스에 키 값과 레코드 위치를 두어 순차 접근과 직접 접근을 함께 지원하는 파일이다. 소량의 특정 레코드를 찾을 때 순차 파일보다 효율적이지만, 레코드가 추가·삭제되면 인덱스도 알맞게 관리해야 한다.
ㄱ X: 레코드는 무조건 마지막 위치에 추가되는 것이 아니라 적용된 순차 기준에 맞는 위치에 놓여야 한다. 빈 공간이 없으면 오버플로우 영역을 이용할 수 있다.
ㄴ O: 순차 대상 필드의 일정 범위에 해당하는 레코드가 많으면 연속적으로 읽을 수 있어 효과적이다.
ㄷ X: 순서를 유지할 위치에 빈 공간이 없으면 오버플로우 블록이 생길 수 있다. 따라서 '오버플로우 블록을 생성시키지 않는다'가 아니라 '필요하면 오버플로우 블록을 생성할 수 있다'가 맞다.
ㄹ O: 인덱스에는 인덱스 대상 필드의 키 값과 해당 레코드에 접근하기 위한 위치 값이 기록된다.
ㅁ X: 레코드의 추가·삭제는 인덱스와 실제 레코드의 대응 관계에 영향을 주므로 인덱스 관리가 필요하다. 따라서 '인덱스 갱신 없이 가능하다'가 아니라 '필요한 인덱스 갱신이나 재편성이 수반된다'가 맞다.
ㅂ O: 조건을 만족하는 레코드가 소량이면 인덱스로 위치를 좁혀 접근하므로 전체를 차례로 읽는 순차 파일보다 효율적이다.
ㅅ O: 다중 레벨 인덱스에서는 최하위 인덱스가 키 순으로 정렬된 순차 파일 형식으로 구성되고, 그 위 레벨이 아래 인덱스를 가리킨다.
따라서 옳은 것은 ㄴ, ㄹ, ㅂ, ㅅ의 4개이다.

선지별 해설

① X: '2개'가 틀렸다. 옳은 설명은 ㄴ, ㄹ, ㅂ, ㅅ의 4개이므로 '4개'가 맞다.
② X: '3개'가 틀렸다. ㄱ, ㄷ, ㅁ은 틀리고 ㄴ, ㄹ, ㅂ, ㅅ은 옳으므로 '4개'가 맞다.
③ O: 옳은 설명이 ㄴ, ㄹ, ㅂ, ㅅ으로 모두 4개이므로 맞다.
④ X: '5개'가 틀렸다. 옳은 설명은 4개이므로 '4개'가 맞다.

오답 포인트

ㄱ의 '마지막 위치'와 ㄷ의 '오버플로우 블록을 생성시키지 않는다'를 함께 옳다고 보기 쉽다. 순차 파일의 핵심은 단순한 끝 추가가 아니라 정해진 기준의 순서 유지이며, 중간 삽입 공간이 부족하면 오버플로우 블록이 생길 수 있다. 또한 ㅁ은 인덱스와 레코드 위치의 대응 관계를 관리해야 한다는 점을 놓치게 하는 표현이다.

다시 나오면 이렇게 판단

먼저 순차 파일은 '넓은 범위의 연속 처리', 인덱스 순차 파일은 '인덱스를 이용한 소량의 빠른 접근'으로 구분한다. 이어 삽입·삭제가 나오면 순서 유지, 오버플로우, 인덱스 갱신 필요 여부를 확인하고 각 문장을 O/X로 확정한 뒤 개수를 센다.

NoteBase

암기 포인트

순차 파일은 범위·연속 처리, 인덱스 순차 파일은 키와 위치 값으로 빠른 접근, 삽입 공간 부족은 오버플로우이다.

NoteBase

문 9. Java 프로그램의 실행 결과로 옳은 것은?

```
public class B extends A {
 int a = 20;
 public B() {
   System.out.print("다");
 }
 public B(int x) {
   System.out.print("라");
 }
}
```

```
public class A {
 int a = 10;
 public A() {
   System.out.print("가");
 }
 public A(int x) {
   System.out.print("나");
 }
 public static void main(String[] a){
   B b1 = new B();
   A b2 = new B(1);
  System.out.print(b1.a + b2.a);
 }
}
```

① 다라30

② 다라40

③ 가다가라30

④ 가다가라40

정답 3번

출제 포인트

이 문제는 자식 클래스의 생성자보다 부모 클래스의 생성자가 먼저 실행된다는 점과, 인스턴스 변수는 객체의 실제 클래스가 아니라 참조 변수의 선언 타입을 기준으로 접근한다는 점을 묻는다.

NoteBase

올바른 순서

1. new B()를 실행하면 B()의 첫 문장에 생략된 super()가 호출되어 A()가 먼저 '가'를 출력한다.
2. 이어서 B()의 본문이 '다'를 출력하므로 첫 번째 객체 생성까지 '가다'가 출력된다.
3. new B(1)을 실행해도 B(int)에 명시적인 super(1)이 없으므로 생략된 super()가 A()를 호출해 '가'를 출력하고, 이어 B(int)가 '라'를 출력한다. 따라서 누적 문자열은 '가다가라'이다.
4. b1은 B 타입이므로 b1.a는 20이고, b2는 A 타입이므로 b2.a는 10이다. 두 값의 합 30이 마지막에 출력된다.

왜 이 순서인가: 자식 생성자는 부모 부분의 초기화가 끝난 뒤 본문을 실행하므로 A()의 출력이 각 B 생성자의 출력보다 앞선다. 또한 a는 메서드가 아니라 숨겨진 인스턴스 변수이므로 참조 변수의 선언 타입에 따라 선택된다.

선지별 해설

① X — '다라30'은 각 B 생성자에서 먼저 호출되는 A()의 출력 '가'를 두 번 모두 누락한 결과이다. 올바른 출력은 '가다가라30'이다.

② X — '다라40'은 부모 생성자의 출력 '가'를 누락하고 b2.a를 B의 a인 20으로 잘못 계산한 결과이다. 올바르게는 b2의 선언 타입 A에 따라 b2.a가 10이므로 전체 출력은 '가다가라30'이다.

③ O — 생성자 호출로 '가다'와 '가라'가 차례로 출력되고, b1.a와 b2.a의 합이 20+10=30이므로 '가다가라30'이 출력된다.

④ X — '가다가라40'은 생성자 출력 순서는 맞지만 b2.a를 실제 객체 B의 필드 20으로 판단한 결과이다. 올바르게는 필드 접근이 참조 타입 A를 따르므로 b2.a는 10이고 전체 출력은 '가다가라30'이다.

오답 포인트

B(int)가 A(int)를 자동으로 호출한다고 생각하기 쉽지만, 인수가 있는 부모 생성자는 자동 선택되지 않는다. B(int)에 super(1)이 없으므로 컴파일러가 super()를 넣어 A()를 호출한다. 또한 필드는 오버라이딩되지 않고 숨겨지므로 A b2 = new B(1)에서 b2.a는 A의 a인 10이다.

다시 나오면 이렇게 판단

생성자마다 첫 줄의 super 호출을 먼저 확인해 출력 순서를 정한다. 그다음 메서드는 실제 객체 타입, 필드는 참조 변수의 선언 타입을 기준으로 선택하여 마지막 연산을 계산한다.

암기 포인트

생성자는 부모부터, 숨겨진 필드는 참조 타입부터 확인한다.

NoteBase

문 10. C 언어로 작성된 프로그램의 실행 결과로 옳은 것은?

```
#include <stdio.h>

double h(double *f, int d, double x){
        int i;
        double res = 0.0;
        for(i=d-1; i >= 0; i--){
                res = res * x + f[i];
        }
        return res;
}

int main() {
        double f[] = {1, 2, 3, 4};
        printf("%3.1f\n", h(f, 4, 2));
        return 0;
}
```

① 11.0 ② 26.0 ③ 49.0 ④ 112.0

정답 3번

출제 포인트

반복문에서 배열 원소를 역순으로 대입하며 갱신되는 *res*의 값을 추적하는 문제이다. 이 계산은 다항식을 효율적으로 계산하는 호너법과 같은 구조이다.

조건 정리

주어진 값: $f = \{1, 2, 3, 4\}$, $d = 4$, $x = 2$이다.
구해야 할 값: h 함수가 반환하여 printf가 소수점 첫째 자리까지 출력하는 값이다.
주의할 조건: 반복문의 인덱스 i는 $d - 1 = 3$부터 0까지 감소하므로 $f[3]$, $f[2]$, $f[1]$, $f[0]$ 순서로 사용한다.

사용할 공식

각 반복에서 다음 점화식으로 값을 갱신한다.

$res \leftarrow res \times x + f[i]$

이 공식을 쓰는 이유는 반복문 본문의 res = res * x + f[i]가 매 단계 그대로 실행되기 때문이다.

NoteBase

계산 과정

초깃값은 $res = 0$이다.
1. $i = 3$: $res = 0 \times 2 + 4 = 4$
2. $i = 2$: $res = 4 \times 2 + 3 = 11$
3. $i = 1$: $res = 11 \times 2 + 2 = 24$
4. $i = 0$: $res = 24 \times 2 + 1 = 49$

따라서 h 함수는 49를 반환하고, %.1f 형식으로 출력하면 49.0이다.

선지 판단

① X — 11.0은 $i = 2$까지 실행했을 때의 중간값일 뿐 최종 출력값이 아니다. 11.0이 아니라 끝까지 반복한 49.0이 올바른 값이다.
② X — 26.0은 배열을 $f[0]$부터 오름차순으로 계산했을 때 나오는 값이지만, 실제 반복문은 $f[3]$부터 처리한다. 26.0이 아니라 49.0이 올바른 값이다.
③ O — 네 번의 반복을 모두 실행하면 $res = 49$이고, 소수점 첫째 자리까지 출력되므로 49.0이다.
④ X — 112.0은 점화식을 실제 인덱스 순서대로 적용한 결과가 아니다. 112.0이 아니라 $0 \to 4 \to 11 \to 24 \to 49$를 거쳐 출력되는 49.0이 올바른 값이다.

오답 포인트

핵심 함정은 배열을 선언된 순서대로 읽는 것이 아니라 반복문의 인덱스가 3부터 0까지 감소한다는 점이다. 또한 11은 계산 도중의 값이므로 반복 종료 전에 멈추면 ①을 고르게 된다.

다시 나오면 이렇게 판단

초깃값을 먼저 적고, 반복문의 시작 인덱스·종료 조건·증감 방향을 확인한 뒤 각 회차의 res를 차례로 기록한다. 마지막에는 printf의 출력 형식까지 반영한다.

암기 포인트

역순 호너 계산은 '현재 값에 x를 곱하고 다음 계수를 더한다'이며, 이 문제의 흐름은 $0 \to 4 \to 11 \to 24 \to 49$이다.

NoteBase

문 11. (가), (나)에서 설명하는 악성 프로그램의 용어를 바르게 짝지은 것은?

(가) 사용자 컴퓨터의 데이터를 암호화시켜 파일을 사용할 수 없도록 한 후 암호화를 풀어주는 대가로 금전을 요구하는 악성 프로그램
(나) '○○○초대장' 등의 내용을 담은 문자 메시지 내에 링크된 인터넷 주소를 클릭하면 악성 코드가 설치되어 사용자의 정보를 빼가거나 소액결제를 진행하는 악성 프로그램

	(가)	(나)
①	스파이웨어	트로이목마
②	랜섬웨어	파밍(Pharming)
③	스파이웨어	피싱(Phishing)
④	랜섬웨어	스미싱(Smishing)

정답 4번

출제 포인트

이 문제는 악성 프로그램의 작동 방식과 유포 수단을 보고 랜섬웨어와 스미싱을 구별하는 문제이다.

핵심 개념

랜섬웨어: 사용자의 파일이나 시스템을 암호화하거나 접근하지 못하게 한 뒤, 복구를 대가로 금전을 요구하는 악성 프로그램이다. 핵심 키워드는 '암호화·사용 불가·금전 요구'이다.
스미싱: 문자메시지와 피싱의 합성어로, 문자메시지의 악성 링크를 누르게 하여 악성 코드를 설치하고 개인정보를 탈취하거나 소액결제를 유도하는 수법이다. 핵심 키워드는 '문자메시지·링크·악성 앱 설치'이다.
스파이웨어는 사용자 몰래 정보를 수집하는 악성 프로그램이고, 트로이목마는 정상 프로그램으로 위장하여 침투하는 악성 프로그램이다. 파밍은 사용자를 위조 사이트로 유도하는 수법이며, 피싱은 거짓 전자우편이나 웹사이트 등으로 개인정보를 속여 빼내는 수법이다.

NoteBase

선지별 해설

① X — (가)의 '스파이웨어'와 (나)의 '트로이목마'가 모두 틀렸다. (가)는 파일을 암호화하고 복구 대가를 요구하므로 랜섬웨어이고, (나)는 문자메시지의 링크로 악성 코드를 설치하게 하므로 스미싱이다. 올바르게는 '(가) 랜섬웨어, (나) 스미싱'이다.
② X — (가)의 '랜섬웨어'는 맞지만 (나)의 '파밍'이 틀렸다. 파밍은 인터넷 주소 입력이나 접속 과정 등을 조작하여 위조 사이트로 유도하는 수법이고, 제시문처럼 문자메시지의 링크를 이용하는 수법은 스미싱이다. 올바르게는 '(가) 랜섬웨어, (나) 스미싱'이다.
③ X — (가)의 '스파이웨어'와 (나)의 '피싱'이 모두 틀렸다. (가)는 단순한 정보 수집이 아니라 데이터를 암호화하고 금전을 요구하므로 랜섬웨어이다. (나)는 넓게는 피싱 계열이지만, 문자메시지의 링크를 이용한다는 구체적 수단에 따라 스미싱이라고 한다. 올바르게는 '(가) 랜섬웨어, (나) 스미싱'이다.
④ O — (가)는 데이터를 암호화한 뒤 복구 대가를 요구하는 랜섬웨어이고, (나)는 문자메시지의 악성 링크를 통해 정보 탈취나 소액결제를 유도하는 스미싱이므로 두 용어가 모두 바르게 연결되었다.

오답 포인트

(나)의 정보 탈취 행위만 보고 피싱을 고르기 쉽지만, '문자메시지 내 링크'라는 유포 수단이 제시되면 더 구체적인 용어인 스미싱으로 판단해야 한다. 또한 (가)는 정보 수집보다 '파일 암호화 후 금전 요구'가 결정적 단서이다.

다시 나오면 이렇게 판단

악성 행위의 결과와 전달 수단을 나누어 본다. 파일을 잠그고 돈을 요구하면 랜섬웨어이고, 문자메시지의 링크로 악성 코드 설치나 결제를 유도하면 스미싱이다.

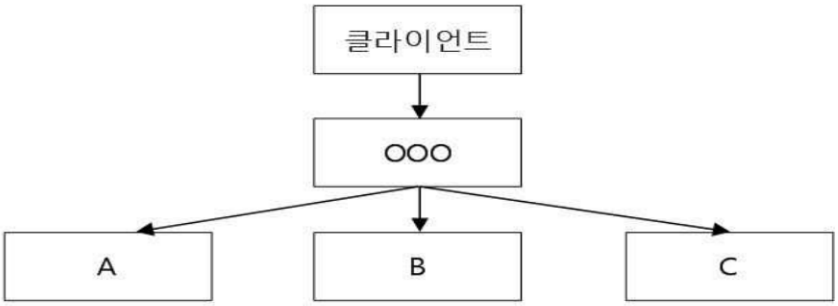
암기 포인트

암호화 후 금전 요구는 랜섬웨어, 문자 링크를 통한 사기는 스미싱이다.

NoteBase

문 12. 다음에서 설명하는 디자인 패턴으로 옳은 것은?

클라이언트와 서브시스템 사이에 ○○○ 객체를 세워놓음으로써 복잡한 관계를 구조화한 디자인 패턴이다. ○○○ 패턴을 사용하면 서브시스템의 복잡한 구조를 의식하지 않고, ○○○에서 제공하는 단순화된 하나의 인터페이스만 사용하므로 클래스 간의 의존관계가 줄어들고 복잡성 또한 낮아지는 효과를 가져 온다.



① MVC pattern
② facade pattern
③ mediator pattern
④ bridge pattern

정답 2번

출제 포인트

복잡한 서브시스템 앞에 단순화된 하나의 인터페이스를 두어 클라이언트와 서브시스템 사이의 의존성을 줄이는 디자인 패턴을 구별하는 문제이다.

핵심 개념

퍼사드 패턴(facade pattern)은 복잡한 서브시스템의 여러 기능을 하나의 단순한 통합 인터페이스로 제공하는 구조 패턴이다. 클라이언트는 서브시스템의 내부 구조를 직접 다루지 않고 퍼사드를 통해 필요한 기능을 사용하므로 결합도와 사용 복잡성이 낮아진다.

NoteBase

선지별 해설

① X — MVC 패턴은 애플리케이션을 모델(Model), 뷰(View), 컨트롤러(Controller)로 분리하는 패턴이다. 틀린 표현은 'MVC pattern'이며, 제시문의 설명에 맞는 표현은 'facade pattern'이다.
② O — facade pattern은 클라이언트와 복잡한 서브시스템 사이에 단순화된 하나의 인터페이스를 제공한다. 제시문의 설명과 도식처럼 클라이언트가 중간 객체를 통해 A, B, C로 이루어진 서브시스템을 이용하는 구조에 해당한다.
③ X — mediator pattern은 여러 객체 사이의 상호작용을 중재자 객체에 집중하여 객체들이 서로 직접 참조하지 않게 하는 행위 패턴이다. 틀린 표현은 'mediator pattern'이며, 복잡한 서브시스템에 단일 인터페이스를 제공하는 패턴은 'facade pattern'이다.
④ X — bridge pattern은 추상화와 구현을 분리하여 두 부분을 독립적으로 변경·확장할 수 있게 하는 구조 패턴이다. 틀린 표현은 'bridge pattern'이며, 클라이언트가 서브시스템의 내부 복잡성을 의식하지 않게 하는 패턴은 'facade pattern'이다.

오답 포인트

도식에서 하나의 객체가 여러 객체와 연결되어 있어 ③ mediator pattern으로 혼동하기 쉽다. 그러나 제시문의 결정적 표현은 '서브시스템에서 제공하는 단순화된 하나의 인터페이스'이며, 이는 객체 간 상호작용의 중재가 아니라 서브시스템 사용 창구의 단순화를 뜻한다.

다시 나오면 이렇게 판단

'복잡한 서브시스템', '단순화된 통합 인터페이스', '클라이언트가 내부 구조를 몰라도 됨'이 함께 제시되면 facade pattern으로 판단한다. 객체들의 상호작용을 한곳에서 조정하면 mediator, 추상화와 구현을 분리하면 bridge, 모델·뷰·컨트롤러로 역할을 나누면 MVC이다.

암기 포인트

Facade는 복잡한 서브시스템 앞에 세우는 하나의 간단한 출입구이다.

NoteBase

문 13. SQL의 명령을 DDL, DML, DCL로 구분할 경우, 이를 바르게 짝지은 것은?

	DDL	DML	DCL
①	RENAME	SELECT	COMMIT
②	UPDATE	SELECT	GRANT
③	RENAME	ALTER	COMMIT
④	UPDATE	ALTER	GRANT

정답 1번

출제 포인트

SQL 명령어를 데이터 정의어(DDL), 데이터 조작어(DML), 데이터 제어어(DCL)로 올바르게 분류할 수 있는지를 묻는 문제이다.

핵심 개념

DDL은 데이터베이스 객체의 구조를 정의하거나 변경하는 명령어로 CREATE, ALTER, DROP, RENAME 등이 해당한다.
DML은 테이블의 데이터를 조회하거나 삽입·수정·삭제하는 명령어로 SELECT, INSERT, UPDATE, DELETE 등이 해당한다.
DCL은 데이터 접근 권한이나 작업의 확정·취소를 제어하는 명령어로 GRANT, REVOKE, COMMIT, ROLLBACK 등이 해당한다.
핵심 대비는 구조 정의·변경은 DDL, 데이터 조회·변경은 DML, 권한·트랜잭션 제어는 DCL이라는 점이다.

선지별 해설

① O — RENAME은 객체의 이름을 바꾸는 DDL, SELECT는 데이터를 조회하는 DML, COMMIT은 작업 결과를 확정하는 DCL이므로 세 명령어가 모두 바르게 분류되어 있다.
② X — DDL 칸의 UPDATE가 틀린 표현이다. UPDATE는 테이블의 데이터를 수정하는 DML이며, DDL에는 RENAME이나 ALTER 등이 들어가야 한다. SELECT는 DML, GRANT는 권한을 부여하는 DCL로 올바르다.
③ X — DML 칸의 ALTER가 틀린 표현이다. ALTER는 테이블 등 데이터베이스 객체의 구조를 변경하는 DDL이며, DML에는 SELECT나 UPDATE 등이 들어가야 한다. RENAME은 DDL, COMMIT은 DCL로 올바르다.
④ X — DDL 칸의 UPDATE와 DML 칸의 ALTER가 모두 틀린 표현이다. 올바르게는 UPDATE가 DML이고 ALTER가 DDL이다. GRANT는 권한을 부여하는 DCL로 올바르다.

NoteBase

오답 포인트

UPDATE와 ALTER는 모두 '변경'한다는 뜻처럼 보여 혼동하기 쉽다. UPDATE는 데이터 값인 행의 내용을 수정하고, ALTER는 테이블 같은 객체의 구조를 변경한다는 차이가 핵심이다.

다시 나오면 이렇게 판단

명령어가 바꾸는 대상을 확인한다. 테이블·객체의 구조나 이름을 바꾸면 DDL, 행의 값을 조회·변경하면 DML, 권한을 부여하거나 작업 결과를 확정·취소하면 DCL로 판단한다.

암기 포인트

구조는 DDL, 데이터는 DML, 권한·확정·취소는 DCL이다. RENAME·ALTER는 DDL, SELECT·UPDATE는 DML, GRANT·COMMIT은 DCL로 묶는다.

NoteBase

문 14. ㉠과 ㉡에 들어갈 용어로 바르게 짝지은 것은?

> (㉠)은/는 구글에서 개발해서 공개한 인공지능 응용프로그램 개발용 오픈소스 프레임워크이다. 이 프레임워크를 사용할 때 인공지능 소프트웨어가 이미지 및 음성을 인식하기 위해서는 신경망의 (㉡) 모델을 주로 사용한다.

	㉠	㉡
①	텐서플로우	논리곱 신경망
②	알파고	퍼셉트론
③	노드레드	인공 신경망
④	텐서플로우	합성곱 신경망

정답 4번

출제 포인트

구글이 공개한 인공지능 개발 프레임워크와 이미지·음성 인식에 주로 활용되는 신경망 모델을 올바르게 연결하는 문제이다.

핵심 개념

텐서플로: 구글이 개발하여 공개한 오픈소스 머신러닝·인공지능 개발 프레임워크이다.
합성곱 신경망(CNN): 입력 데이터의 국소적 특징을 추출하는 합성곱 연산을 이용하며, 이미지와 음성의 특징을 인식하는 데 널리 쓰이는 신경망 모델이다.
알파고는 바둑 인공지능 시스템이고, 노드레드는 흐름 기반 프로그래밍 도구이므로 텐서플로와 같은 답이 아니다.

NoteBase

선지별 해설

① X — (ㄱ)의 텐서플로는 맞지만, (ㄴ)의 '논리곱 신경망'이 틀렸다. 이미지와 음성의 특징을 인식할 때 주로 사용하는 모델은 합성곱 신경망이다. 따라서 '텐서플로ㅡ합성곱 신경망'으로 고쳐야 한다.
② X — 알파고는 구글 딥마인드가 개발한 바둑 인공지능 시스템이지, 제시문에서 말한 범용 오픈소스 개발 프레임워크가 아니다. 퍼셉트론도 기본적인 인공 신경망 단위·모델로서 제시문의 (ㄴ)에 해당하지 않는다. '텐서플로ㅡ합성곱 신경망'으로 고쳐야 한다.
③ X — 노드레드는 흐름 기반 프로그래밍 도구이므로 구글이 공개한 해당 인공지능 개발 프레임워크가 아니다. 또한 '인공 신경망'은 넓은 상위 개념이고, 제시문이 요구하는 구체적인 모델은 합성곱 신경망이다. '텐서플로ㅡ합성곱 신경망'으로 고쳐야 한다.
④ O — 텐서플로는 구글이 개발하여 공개한 오픈소스 인공지능 개발 프레임워크이고, 합성곱 신경망은 이미지와 음성의 특징 인식에 주로 활용되는 모델이므로 두 빈칸에 모두 알맞다.

오답 포인트

①의 '논리곱'과 ④의 '합성곱'은 글자 모양이 비슷하지만 전혀 다른 표현이다. 또한 ②의 알파고는 인공지능의 한 응용 시스템이고, ③의 인공 신경망은 넓은 개념이므로 각각 개발 프레임워크인 텐서플로와 구체적 모델인 합성곱 신경망을 대신할 수 없다.

다시 나오면 이렇게 판단

'구글 공개·오픈소스·인공지능 개발 프레임워크'라는 단서에서는 텐서플로를 찾고, '이미지·음성의 특징 인식'이라는 단서에서는 합성곱 신경망을 찾는다. 도구·응용 시스템·신경망의 상위 개념을 서로 구분하면 된다.

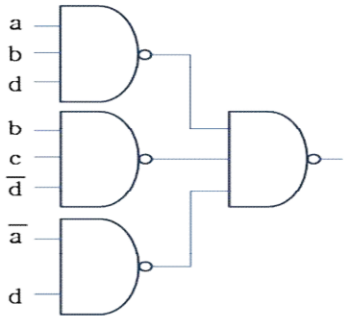
암기 포인트

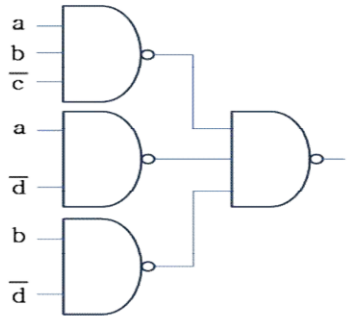
구글의 오픈소스 프레임워크는 텐서플로, 이미지·음성 특징 인식 모델은 합성곱 신경망이다.

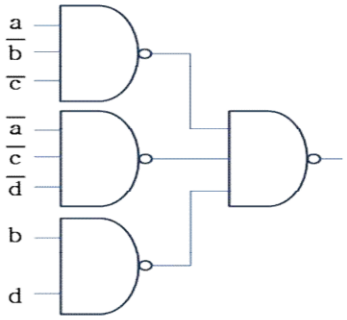
NoteBase

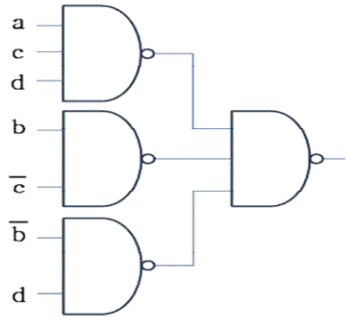
문 15. 아래에 제시된 K-map(카르노 맵)을 NAND 게이트들로만 구성한 것으로 옳은 것은?

ab \ cd	00	01	11	10
00	1	0	0	0
01	1	1	1	0
11	0	1	1	0
10	1	1	0	0

① 

② 

③ 

④ 

정답 3번

출제 포인트

K-map에서 인접한 1을 묶어 최소 논리식을 구하고, NAND-NAND 회로가 그 곱의 합을 구현하는지 판단하는 문제이다.

조건 정리

K-map의 행과 열은 각각 그레이 코드 순서인 00, 01, 11, 10으로 배열되어 있다.
서로 인접한 1은 2의 거듭제곱 개수로 묶으며, 묶음 안에서 값이 변하지 않는 변수만 곱항에 남긴다.
NAND 게이트의 출력에 붙은 작은 원은 해당 곱항 전체의 부정을 뜻한다.

NoteBase

사용할 공식

드모르간 법칙에 따라 최종 NAND의 출력은

$F = \overline{\overline{P_1 P_2 P_3}} = P_1 + P_2 + P_3$

가 된다. 따라서 앞단 NAND들의 입력으로 만든 곱항이 K-map의 최소 곱항과 같은지 확인한다.

계산 과정

1. 행 01·11과 열 01·11의 네 칸을 묶으면 b와 d가 고정되므로 곱항은 bd이다.
2. 행 10의 열 00·01 두 칸을 묶으면 $a = 1, b = 0, c = 0$이 고정되므로 곱항은 $a\bar{b}\bar{c}$이다.
3. 열 00의 행 00·01 두 칸을 묶으면 $a = 0, c = 0, d = 0$이 고정되므로 곱항은 $\bar{a}\bar{c}\bar{d}$이다.
4. 따라서 $F = a\bar{b}\bar{c} + \bar{a}\bar{c}\bar{d} + bd$이다. 이 세 곱항을 각각 앞단 NAND에 넣고 그 출력을 마지막 NAND로 결합한 회로가 ③이다.

선지 판단

① X — 이 회로는 $F = abd + bc\bar{d} + \bar{a}d$를 구현한다. K-map에서 1인 $abcd = 0000$도 출력하지 못하므로 틀리다. 올바르게는 앞단 곱항이 $a\bar{b}\bar{c}$, $\bar{a}\bar{c}\bar{d}$, bd가 되어야 한다.

② X — 이 회로는 $F = ab\bar{c} + a\bar{d} + b\bar{d}$를 구현한다. 이 식도 K-map의 0000 위치의 1을 포함하지 못한다. 올바르게는 $F = a\bar{b}\bar{c} + \bar{a}\bar{c}\bar{d} + bd$를 구현해야 한다.

③ O — 세 앞단 NAND의 곱항이 각각 $a\bar{b}\bar{c}$, $\bar{a}\bar{c}\bar{d}$, bd이다. 마지막 NAND에 의해 이 곱항들이 OR로 결합되어 K-map의 최소식과 정확히 일치한다.

④ X — 이 회로는 $F = acd + b\bar{c} + \bar{b}d$를 구현한다. 이 식 역시 K-map의 0000 위치의 1을 포함하지 못한다. 올바르게는 세 곱항을 $a\bar{b}\bar{c}$, $\bar{a}\bar{c}\bar{d}$, bd로 바꾸어야 한다.

오답 포인트

K-map의 맨 위 행과 맨 아래 행은 서로 인접하지만, 0이 포함된 칸까지 한 묶음으로 처리할 수는 없다. 또한 NAND 기호의 출력 원을 개별 입력의 보수로 오해하면 선지의 곱항을 잘못 읽게 된다.

다시 나오면 이렇게 판단

K-map의 1을 가능한 큰 묶음으로 묶어 곱의 합을 먼저 만든 뒤, 앞단 NAND 하나당 곱항 하나가 대응하는지 확인한다. 마지막 NAND는 드모르간 법칙에 의해 이 곱항들을 OR로 결합한다.

암기 포인트

NAND-NAND 2단 회로는 곱의 합 $P_1 + P_2 + P_3$을 구현한다.

NoteBase

문 16. 다음은 숫자를 처리하는 C 프로그램이다. 프로그램에서 ㉠과 ㉡에 들어갈 내용과 3 2 1 4를 입력하였을 때의 출력결과를 바르게 짝지은 것은? (단, 다음 프로그램에 문법적 오류는 없다고 가정한다.)

```
#include <stdio.h>
#include <stdlib.h>

void a (int n, int *num) {
   for (int i = 0; i < n; i++)
      scanf("%d", &(num[i]));
}
void c(int *a, int *b) {
   int t;
   t = *a; *a = *b; *b = t;
}
void b(int n, int *lt) {
   int a, b;
   for (a = 0; a < n-1; a++)
      for (b = a + 1; b < n; b++)
               if (lt[a] > lt[b]) c ( ㉠ , ㉡ ) ;
}
int main() {
   int n;
   int *num;
   printf("How many numbers?");
   scanf("%d", &n);
   num = (int *)malloc(sizeof(int) * n);
   a(n, num);
   b(n, num);
   for (int i = 0; i < n; i++)
      printf("%d ", num[i]);
}
```

㉠	㉡	출력 결과
① lt+a	lt+b	1 2 3 4
② lt+a	lt+b	1 2 4
③ lt[a]	lt[b]	4 3 2 1
④ lt[a]	lt[b]	4 2 1

정답 2번

NoteBase

출제 포인트

입력값의 역할을 구분하고, 두 배열 원소를 교환하도록 포인터 인수를 전달한 뒤 중첩 반복문의 실행 결과를 추적하는 문제이다.

조건 정리

입력 3 2 1 4에서 첫 번째 값 3은 n에 저장된다. 따라서 a 함수가 배열 num에 입력하는 실제 자료는 2, 1, 4의 세 개이다. c 함수의 매개변수는 int * 형식이므로 배열 원소의 주소인 lt+a와 lt+b를 전달해야 한다.

사용할 원리

lt+a는 &lt[a]와 같고 lt+b는 &lt[b]와 같다. c 함수는 전달받은 두 주소가 가리키는 값을 서로 바꾼다. b 함수는 앞쪽 원소 lt[a]가 뒤쪽 원소 lt[b]보다 클 때 두 원소를 교환하므로 배열을 오름차순으로 정렬한다.

실행 과정

초기 배열은 2, 1, 4이다. a=0, b=1일 때 2>1이므로 두 값을 교환하여 1, 2, 4가 된다. a=0, b=2와 a=1, b=2에서는 앞의 값이 더 크지 않으므로 교환하지 않는다. 최종 출력은 1 2 4이다.

선지 판단

① X — lt+a와 lt+b는 올바른 인수이지만 출력 결과 1 2 3 4가 틀렸다. 입력의 첫 값 3은 배열 원소가 아니라 n이므로, 올바른 출력은 1 2 4이다.
② O — lt+a와 lt+b는 각각 lt[a]와 lt[b]의 주소이므로 c 함수가 두 원소를 정상적으로 교환한다. 세 원소 2, 1, 4가 오름차순으로 정렬되어 1 2 4가 출력된다.
③ X — lt[a]와 lt[b]는 int 값인데 c 함수는 int * 형식의 주소를 요구하므로 인수 형식이 맞지 않는다. 올바르게는 c(lt+a, lt+b)로 호출해야 하며, 그때 출력은 1 2 4이다.
④ X — lt[a]와 lt[b]를 넘기면 원소의 주소가 아니라 값이 전달되어 c 함수의 포인터 매개변수와 맞지 않으며, 4 2 1이라는 출력도 나오지 않는다. 올바르게는 c(lt+a, lt+b)로 호출하고 결과는 1 2 4이다.

오답 포인트

①의 1 2 3 4는 입력열의 첫 값 3까지 정렬 대상이라고 착각한 결과이다. scanf가 먼저 n을 읽고, 이어서 a 함수가 정확히 n개의 수만 배열에 읽는다는 실행 순서를 구분해야 한다. 또한 배열 원소의 값 lt[a]와 그 주소 lt+a를 혼동하기 쉽다.

다시 나오면 이렇게 판단

먼저 각 scanf가 입력열의 어느 값을 소비하는지 순서대로 나눈다. 다음으로 함수의 포인터 매개변수에는 값이 아니라 주소가 전달되는지 확인하고, 조건이 참일 때 배열 상태가 어떻게 바뀌는지를 반복문 순서대로 추적한다.

NoteBase

암기 포인트

배열에서 lt+a는 &lt[a]와 같아 원소의 주소이고, lt[a]는 원소의 값이다. 첫 입력 3은 개수 n이며 정렬 자료는 2, 1, 4이다.

NoteBase

문 17. 엑셀 시트를 이용해 수식을 실행한 결과, 값이 나머지와 다른 것은?

	A
1	3
2	7
3	5
4	3
5	0
6	1

① =GCD(A1,A6)

② =MEDIAN(A1:A6)

③ =MODE(A1:A6)

④ =POWER(A1,A6)

정답 1번

출제 포인트

이 문제는 GCD, MEDIAN, MODE, POWER 함수의 뜻을 알고 각 수식의 결과를 비교하는 문제이다.

조건 정리

주어진 값: A1부터 A6까지 차례로 3, 7, 5, 3, 0, 1이다.
구해야 할 값: 나머지 세 수식과 결과가 다른 수식이다.
주의할 조건: '다른 것'을 묻는 부정형 문항이므로 각 수식의 값을 모두 계산해 비교한다.

사용할 공식

GCD는 주어진 수들의 최대공약수, MEDIAN은 자료를 크기순으로 배열했을 때의 중앙값, MODE는 가장 자주 나타나는 최빈값, POWER는 밑을 지수만큼 거듭제곱한 값이다.
이 함수들을 쓰는 이유: 각 선지가 서로 다른 엑셀 함수를 사용하므로 함수의 정의에 따라 반환값을 구해야 결과가 다른 하나를 찾을 수 있다.

계산 과정

1. ①은 GCD(3, 1) = 1이다.
2. ②는 자료를 0, 1, 3, 3, 5, 7로 정렬하면 가운데 두 값이 3과 3이므로 중앙값은 $\frac{3+3}{2} = 3$이다.
3. ③은 3이 두 번으로 가장 많이 나타나므로 최빈값은 3이다.
4. ④는 POWER(3, 1) = $3^1 = 3$이다.

따라서 ①의 결과만 1이고 ②, ③, ④의 결과는 모두 3이다.

NoteBase

선지 판단

① O

- '=GCD(A1,A6)'은 3과 1의 최대공약수인 1을 반환한다. 나머지 세 수식의 결과 3과 다르므로 옳은 선지이다.

② X

- '=MEDIAN(A1:A6)'은 정렬한 자료 0, 1, 3, 3, 5, 7의 가운데 두 값의 평균인 3을 반환한다. '나머지와 다른 값'이 아니라 올바르게는 ③, ④와 같은 값 3을 내는 수식이다.

③ X

- '=MODE(A1:A6)'은 가장 자주 나타나는 값 3을 반환한다. '나머지와 다른 값'이 아니라 올바르게는 ②, ④와 같은 값 3을 내는 수식이다.

④ X

- '=POWER(A1,A6)'은 $3^1 = 3$을 반환한다. '나머지와 다른 값'이 아니라 올바르게는 ②, ③과 같은 값 3을 내는 수식이다.

오답 포인트

①의 GCD를 범위 A1:A6 전체의 최대공약수로 오해하기 쉽지만, 수식의 인수는 A1과 A6 두 셀뿐이다. 또한 ②의 MEDIAN은 평균이 아니라 정렬 후 가운데 값이며, 자료의 개수가 짝수이면 가운데 두 값의 평균을 구한다.

다시 나오면 이렇게 판단

먼저 함수의 인수가 개별 셀인지 범위인지 확인하고, 각 함수의 정의에 따라 반환값을 하나씩 계산한 뒤 문항의 '같은 것' 또는 '다른 것'이라는 극성을 마지막에 적용한다.

암기 포인트

GCD=최대공약수, MEDIAN=중앙값, MODE=최빈값, POWER=거듭제곱이다.

NoteBase

문 18. RISC(Reduced Instruction Set Computer)에 대한 설명으로 옳은 것의 총 개수는?

ㄱ. 칩 제작을 위한 R&D 비용이 감소한다.
ㄴ. 개별 명령어 디코딩 시간이 CISC(Complex Instruction Set Computer)보다 많이 소요된다.
ㄷ. 동일한 기능을 구현할 경우, CISC보다 적은 수의 레지스터가 필요하다.
ㄹ. 복잡한 연산을 수행하려면 명령어를 반복수행하여야 하므로 CISC의 경우보다 프로그램이 복잡해진다.
ㅁ. 각 명령어는 한 클럭에 실행하도록 고정되어 있어 파이프라인 성능을 향상시킬 수 있다.
ㅂ. 마이크로코드 설계가 어렵다.
ㅅ. 고정된 명령어이므로 명령어 디코딩 속도가 빠르다.

① 2개 ② 3개 ③ 4개 ④ 5개

정답 3번

출제 포인트

이 문제는 RISC와 CISC의 명령어 구조, 제어 방식, 레지스터 수, 파이프라이닝 특성을 비교하여 옳은 설명의 개수를 판단하는 문제이다.

핵심 개념

RISC는 자주 쓰는 단순한 명령어를 중심으로 명령어 형식과 길이를 규칙화하고, 하드와이어드 제어와 많은 범용 레지스터를 활용하는 구조이다. 명령어 해독이 빠르고 파이프라이닝에 유리하지만, 복잡한 연산은 여러 단순 명령어로 나누어 실행하므로 명령어 수가 늘어날 수 있다.
CISC는 하나의 명령어가 복잡한 기능을 수행하도록 다양한 명령어와 주소 지정 방식을 제공하며, 전통적으로 마이크로프로그램 제어를 많이 사용한다.
핵심 대비는 'RISC = 단순·고정 형식·빠른 해독·하드와이어드·많은 레지스터', 'CISC = 복잡·가변 형식·마이크로프로그램 제어'이다.

NoteBase

보기 판단

ㄱ O: RISC는 명령어 집합과 제어 구조가 단순하여 칩 설계 및 제작을 위한 연구·개발 비용을 줄일 수 있다.
ㄴ X: 틀린 표현은 '명령어 디코딩 시간이 CISC보다 많이 소요된다'이다. RISC는 단순하고 고정된 명령어 형식을 사용하므로 일반적으로 CISC보다 디코딩이 빠르다.
ㄷ X: 틀린 표현은 'CISC보다 적은 수의 레지스터가 필요하다'이다. RISC는 메모리 접근을 줄이고 단순 명령어를 효율적으로 처리하기 위해 일반적으로 많은 범용 레지스터를 사용한다.
ㄹ O: RISC에서는 복잡한 연산을 여러 단순 명령어로 나누어 반복 수행하므로 CISC보다 프로그램의 명령어 수가 늘고 코드가 복잡해질 수 있다.
ㅁ O: RISC는 명령어 길이와 실행 형식이 규칙적이고 명령어를 한 클럭에 실행하도록 설계하기 쉬워 파이프라인 처리에 유리하다.
ㅂ X: 틀린 표현은 '마이크로코드 설계가 어렵다'이다. RISC는 주로 하드와이어드 제어 방식을 사용하므로 복잡한 마이크로코드 설계 부담이 작다.
ㅅ O: RISC의 고정된 명령어 형식은 명령어 구조를 단순하게 하므로 디코딩 속도를 빠르게 한다.
따라서 옳은 것은 ㄱ, ㄹ, ㅁ, ㅅ의 4개이다.

선지별 해설

① X: 옳은 설명은 2개가 아니라 ㄱ, ㄹ, ㅁ, ㅅ의 4개이다. 올바른 선택지 내용은 '4개'이다.
② X: 옳은 설명은 3개가 아니라 ㄱ, ㄹ, ㅁ, ㅅ의 4개이다. 올바른 선택지 내용은 '4개'이다.
③ O: 옳은 설명이 ㄱ, ㄹ, ㅁ, ㅅ으로 모두 4개이므로 정답이다.
④ X: 옳은 설명은 5개가 아니라 ㄱ, ㄹ, ㅁ, ㅅ의 4개이다. 올바른 선택지 내용은 '4개'이다.

오답 포인트

ㄴ과 ㅅ은 모두 디코딩 속도를 묻지만 방향이 반대이다. RISC의 고정되고 단순한 명령어 형식은 디코딩 시간을 늘리는 것이 아니라 줄인다. 또한 마이크로코드는 CISC의 전형적인 제어 방식이고, 많은 범용 레지스터와 하드와이어드 제어는 RISC의 대표적 특징이라는 점을 혼동하기 쉽다.

다시 나오면 이렇게 판단

각 문장을 '명령어 형식', '제어 방식', '레지스터', '파이프라이닝'의 네 기준으로 나눈다. 단순·고정 형식·빠른 디코딩·하드와이어드 제어·많은 레지스터·파이프라이닝 유리는 RISC에, 복잡한 명령어·마이크로프로그램 제어는 CISC에 연결한 뒤 옳은 문장만 센다.

암기 포인트

RISC는 단순·고정·빠른 해독·많은 레지스터·파이프라인, CISC는 복잡한 명령어·마이크로코드이다.

NoteBase

문 19. 참조 무결성에 대한 설명으로 옳지 않은 것은?

① 검색 연산의 수행 결과는 어떠한 참조 무결성 제약조건도 위배하지 않는다.

② 참조하는 릴레이션에서 튜플이 삭제되는 경우, 참조 무결성 제약조건이 위배될 수 있다.

③ 외래 키 값은 참조되는 릴레이션의 어떤 튜플의 기본 키 값과 같거나 널(NULL) 값일 수 있다.

④ 참조 무결성 제약조건은 DBMS에 의하여 유지된다.

정답 2번

출제 포인트

참조 무결성의 의미와 검색·삭제 연산이 외래 키와 참조 대상 키의 대응 관계에 미치는 영향을 판단하는 문제이다.

핵심 개념

참조 무결성은 참조하는 릴레이션의 외래 키 값이 참조되는 릴레이션에 실제로 존재하는 키 값과 일치하거나, 허용된 경우 널(NULL)이 되도록 유지하는 제약조건이다. 참조하는 쪽의 투플을 삭제하면 참조 관계 자체가 없어지므로 위반이 발생하지 않지만, 참조되는 쪽에서 아직 참조 중인 투플을 삭제하면 외래 키가 존재하지 않는 대상을 가리키게 되어 위반될 수 있다.

선지별 해설

① O — 검색 연산은 데이터 값을 변경하거나 삭제하지 않으므로 외래 키와 참조 대상 키의 대응 관계를 훼손하지 않는다.

② X — 틀린 부분은 '참조하는 릴레이션에서 투플이 삭제되는 경우'이다. 참조하는 쪽의 투플을 삭제하면 그 투플이 갖고 있던 참조가 사라질 뿐이므로 참조 무결성을 위반하지 않는다. 올바르게는 '참조되는 릴레이션에서 다른 투플이 참조하고 있는 투플을 삭제하는 경우 참조 무결성 제약조건이 위반될 수 있다'이다.

③ O — 외래 키 값은 참조되는 릴레이션에 존재하는 기본 키 값과 같아야 하며, 해당 속성에 널이 허용되는 경우에는 널(NULL)일 수도 있다.

④ O — 참조 무결성 제약조건은 데이터의 삽입·수정·삭제 때 DBMS가 검사하고 유지한다.

NoteBase

오답 포인트

②는 '참조하는 릴레이션'과 '참조되는 릴레이션'을 바꾸어 놓은 선지이다. 자식에 해당하는 참조 투플의 삭제는 안전하지만, 부모에 해당하는 참조 대상 투플의 삭제는 남아 있는 외래 키를 고아 값으로 만들 수 있다.

다시 나오면 이렇게 판단

연산 뒤 외래 키가 가리킬 실제 대상이 남아 있는지를 확인한다. 검색은 데이터를 바꾸지 않고, 참조하는 쪽의 삭제는 참조를 없애므로 안전하다. 반면 참조되는 쪽의 삭제는 남은 외래 키의 대상이 사라질 수 있으므로 위반 가능성이 있다.

암기 포인트

자식 삭제는 참조 제거, 부모 삭제는 참조 대상 제거이다. 참조 중인 부모 투플의 삭제가 참조 무결성을 위협한다.

NoteBase

문 20. 프로세스(Process)와 쓰레드(Thread)에 대한 설명으로 옳지 않은 것은?

① 프로세스 내 쓰레드 간 통신은 커널 개입을 필요로 하지 않기 때문에 프로세스 간 통신보다 더 효율적으로 이루어진다.

② 멀티프로세서는 탑재 프로세서마다 쓰레드를 실행시킬 수 있기 때문에 프로세스의 처리율을 향상시킬 수 있다.

③ 한 프로세스 내의 모든 쓰레드들은 정적 영역(Static Area)을 공유한다.

④ 한 프로세스의 어떤 쓰레드가 스택 영역(Stack Area)에 있는 데이터 내용을 변경하면 해당 프로세스의 다른 쓰레드가 변경된 내용을 확인할 수 있다.

정답 4번

출제 포인트

이 문제는 같은 프로세스에 속한 스레드들이 공유하는 자원과 각 스레드가 독립적으로 가지는 자원을 구분하는 문제이다.

핵심 개념

프로세스는 실행 중인 프로그램의 자원 소유 단위이며, 서로 다른 프로세스는 원칙적으로 독립된 주소 공간을 가진다.
스레드는 프로세스 안에서 실행 흐름을 이루는 단위이다. 같은 프로세스의 스레드들은 코드·정적 데이터·힙 등의 영역을 공유하지만, 실행 상태를 보존하는 스택과 레지스터 등은 스레드마다 독립적으로 가진다.

선지별 해설

① O — 같은 프로세스의 스레드들은 주소 공간과 자원을 공유하므로, 프로세스 간 통신처럼 별도의 커널 기반 통신 절차를 거치는 경우보다 통신과 문맥 교환의 부담이 작다.
② O — 멀티프로세서에서는 여러 스레드를 서로 다른 프로세서에서 동시에 실행할 수 있으므로 병렬 처리를 통해 처리율을 높일 수 있다.
③ O — 정적 영역은 프로세스의 주소 공간에 속하므로 같은 프로세스 안의 모든 스레드가 공유한다.
④ X — 틀린 표현은 한 스레드의 스택 데이터 변경 내용을 다른 스레드가 확인할 수 있다는 부분이다. 스택 영역은 스레드마다 독립적으로 할당되므로 다른 스레드가 그 변경 내용을 공유 데이터처럼 직접 확인할 수 없다.
올바르게는 같은 프로세스의 스레드들이 정적 영역이나 힙 영역의 공유 데이터를 변경하면 다른 스레드도 그 내용을 확인할 수 있다고 해야 한다.

NoteBase

오답 포인트

같은 프로세스의 스레드들이 주소 공간을 공유한다는 사실을 모든 메모리 영역을 공유한다는 뜻으로 확대하면 ④를 옳다고 판단하기 쉽다. 공유되는 정적·힙 영역과 스레드별로 독립적인 스택 영역을 구별해야 한다.

다시 나오면 이렇게 판단

스레드 자원 문제에서는 먼저 해당 자원이 프로세스 공용인지 스레드 전용인지 판단한다. 코드·정적 데이터·힙은 공유하고, 스택과 레지스터는 각 스레드가 독립적으로 가진다는 기준을 적용한다.

암기 포인트

코드·정적·힙은 함께 쓰고, 스택·레지스터는 따로 쓴다.

NoteBase