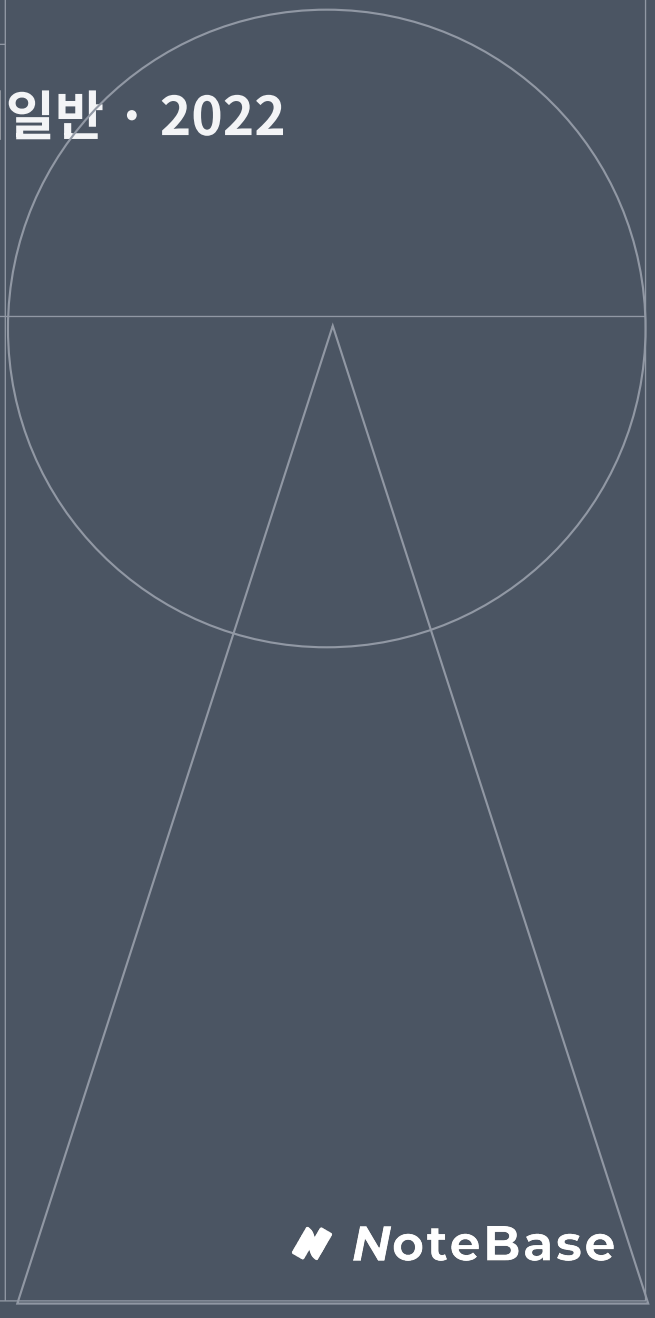
기출OX 해설집

공무원/군무원/9급 · 컴퓨터일반 · 2022



NoteBase

기출OX 해설집

공무원/군무원/9급 · 컴퓨터일반 · 2022

공무원/군무원/9급 · 컴퓨터일반 · 2022

1. 다음 중 2진수 11010010_2의 1의 보수(one's complement)를 16진수로 표현한 것으로 옳은 것은?

① $\square_6$ ② $D2_{16}$

③ $D3_{16}$ ④ $2E_{16}$

[복원 표기]
선지 ①: 2D₁☒ (16진수 2D).

정답 1번

출제 포인트

이 문제는 1의 보수를 구한 뒤, 2진수를 4비트씩 묶어 16진수로 변환하는 문제이다.

조건 정리

주어진 값: 11010010_2
구해야 할 값: 주어진 8비트 2진수의 1의 보수를 16진수로 표현한 값
주의할 조건: 1의 보수는 자릿수를 유지한 채 각 비트의 0과 1을 서로 반전한다.

사용할 공식

1의 보수: $0 \leftrightarrow 1$
16진수 변환: 오른쪽부터 4비트씩 묶어 각 묶음을 하나의 16진수 숫자로 바꾼다.

8비트 수의 1의 보수는 모든 비트를 반전하여 구하며, 16진수 한 자리는 2진수 4비트에 대응하므로 이 방법을 사용한다.

NoteBase

계산 과정

1. 각 비트를 반전한다.
$11010010_2 \rightarrow 00101101_2$
2. 4비트씩 묶는다.
00101101
3. 각 묶음을 16진수로 바꾼다. $0010_2 = 2_{16}$, $1101_2 = D_{16}$이므로
$00101101_2 = 2D_{16}$

선지 판단

① O — $2D_{16}$은 00101101_2과 같으므로 계산 결과와 일치한다.
② X — $D2_{16}$는 11010010_2를 그대로 16진수로 바꾼 값이므로 1의 보수 연산을 하지 않은 결과이다. 올바르게는 $2D_{16}$이다.
③ X — $D3_{16}$는 계산 결과와 일치하지 않는다. 특히 원래 수의 마지막 비트 0은 1로 반전되어야 하므로 하위 4비트는 $1101_2 = D_{16}$가 된다. 올바르게는 $2D_{16}$이다.
④ X — $2E_{16}$의 하위 16진수 숫자 E는 1110_2이므로, 반전 결과의 하위 4비트 1101_2과 다르다. 올바르게는 $2D_{16}$이다.

계산 결과가 $2D_{16}$이므로 정답은 1번이다.

오답 포인트

②의 $D2_{16}$는 원래 2진수를 반전하지 않고 바로 16진수로 변환할 때 나오는 값이다. 또한 1의 보수는 비트 순서를 뒤집는 것이 아니라 각 자리의 0과 1을 바꾸는 연산이라는 점을 구분해야 한다.

다시 나오면 이렇게 판단

먼저 모든 비트를 빠짐없이 반전하고, 반전된 수를 오른쪽부터 4비트씩 묶어 16진수로 변환한다. 마지막으로 각 4비트 묶음을 다시 펼쳐 원래 반전 결과와 같은지 검산한다.

암기 포인트

1의 보수는 각 비트의 0 ↔ 1 반전이고, 16진수 한 자리는 2진수 4비트이다.

NoteBase

2. 논리회로의 조합회로(combinational circuit)와 순차회로(sequential circuit)에 대한 다음의 설명 중 가장 올바르지 않은 것은?
   ① 조합회로는 현재의 입력에 의해서만 출력값이 결정된다.
   ② 순차회로는 현재의 입력과 저장된 값에 의해 출력값이 결정된다.
   ③ 복호기(decoder)와 부호기(encoder)는 조합회로로 구성되는 것이 일반적이다.
   ④ 멀티플렉서(multiplexer)와 디멀티플렉서(demultiplexer)는 순차회로로 구성되는 것이 일반적이다.

정답 4번

출제 포인트

조합회로와 순차회로를 출력 결정 요소와 기억 기능의 유무로 구분하고, 대표 회로를 올바르게 분류하는 문제이다.

핵심 개념

조합회로: 기억 기능이 없으며 출력이 현재 입력의 조합으로만 결정되는 회로이다. 키워드는 현재 입력이다.
순차회로: 기억 소자를 가지며 출력이 현재 입력과 저장된 이전 상태에 의해 결정되는 회로이다. 키워드는 현재 입력과 저장 상태이다.
대표적인 조합회로에는 복호기, 부호기, 멀티플렉서, 디멀티플렉서가 있다.

선지별 해설

① O — 조합회로에는 이전 상태를 저장하는 기억 기능이 없으므로 출력값은 현재 입력에 의해서만 결정된다.
② O — 순차회로는 플립플롭 등의 기억 소자에 저장된 상태를 사용하므로 출력값이 현재 입력과 저장된 값에 의해 결정된다.
③ O — 복호기는 입력 코드를 대응하는 출력으로 변환하고, 부호기는 입력 정보를 코드로 변환하는 회로로서 모두 일반적으로 조합회로로 구성된다.
④ X — 틀린 부분은 '순차회로로 구성되는 것이 일반적이다'이다. 멀티플렉서는 여러 입력 중 하나를 선택하여 출력하고, 디멀티플렉서는 하나의 입력을 선택된 출력선으로 전달하며, 둘 다 일반적으로 조합회로로 구성된다. 올바르게는 '멀티플렉서와 디멀티플렉서는 조합회로로 구성되는 것이 일반적이다'이다.

NoteBase

오답 포인트

멀티플렉서와 디멀티플렉서가 선택선에 따라 신호의 경로를 바꾸므로 상태를 기억하는 순차회로처럼 보일 수 있다. 그러나 선택 결과는 현재 데이터 입력과 현재 선택 입력으로 정해지며 이전 상태를 저장하지 않는다. 또한 문제의 극성이 '가장 올바르지 않은 것'이므로 틀린 설명인 ④를 고른다.

다시 나오면 이렇게 판단

회로가 이전 상태를 저장하는지를 먼저 확인한다. 출력이 현재 입력만의 함수이면 조합회로이고, 현재 입력과 저장된 이전 상태에 함께 의존하면 순차회로이다.

암기 포인트

현재 입력만 사용하면 조합회로, 현재 입력과 기억된 상태를 함께 사용하면 순차회로이다.
복호기·부호기·멀티플렉서·디멀티플렉서는 대표적인 조합회로이다.

NoteBase

3. 숫자 0부터 n까지, n을 포함한 합을 구하는 함수를 C 언어로 구현하고자 한다. 이때 n은 0보다 크거나 같은 경우만 고려한다.
다음에 주어진 프로그램에서 /* 공란 */ 으로 표시된 곳에 채워져야 할 코드로 가장 옳은 것은?

```
int sum(int n) {
  int result = 0;
  /* 공란 */

  return result;
}
```

① for (int i = 0; i++; i < n) result = result + i;
② for (int i = 0; i++; i <= n) result = result + i;
③ for (int i = 0; i <= n; i++) result = result + i;
④ for (int i = 0; i < n; i++) result = result + i;

정답 3번

출제 포인트

C 언어 for 문의 세 식이 놓이는 순서와 0부터 n까지 n을 포함해 반복하는 조건을 묻는 문제이다.

핵심 개념

for 문은 for (초기식; 조건식; 증감식)의 순서로 작성한다. 초기식은 반복을 시작하기 전에 한 번 실행되고, 조건식이 참인 동안 본문을 실행한 뒤 증감식을 실행한다. 0부터 n까지 모두 더하려면 i를 0으로 초기화하고, 조건식을 i <= n으로 두며, 매 반복 뒤 i++로 i를 1씩 증가시켜야 한다.

선지별 해설

① X — 두 번째 식 i++가 조건식 자리에 있다. 최초 검사에서 i++는 증가 전 값 0을 조건값으로 내므로 거짓이 되어 본문이 한 번도 실행되지 않는다. 또한 세 번째 식 i < n은 비교만 할 뿐 i를 증가시키지 않는다. 올바르게는 for (int i = 0; i <= n; i++) result = result + i;이다.
② X — 두 번째 식 i++가 조건식 자리에 있어 최초 검사 결과가 0, 즉 거짓이므로 반복이 시작되지 않는다. 세 번째 식 i <= n도 증감식이 아니라 단순 비교식이다. 올바르게는 for (int i = 0; i <= n; i++) result = result + i;이다.
③ O — i를 0으로 초기화하고, i <= n인 동안 result에 i를 더한 뒤 i++로 증가시킨다. 따라서 0, 1, ..., n이 모두 한 번씩 더해진다.
④ X — 조건식 i < n은 i가 n이 되면 반복을 끝내므로 0부터 n - 1까지만 더하고 n은 빠진다. 올바르게는 조건식을 i <= n으로 바꾼 for (int i = 0; i <= n; i++) result = result + i;이다.

NoteBase

오답 포인트

①과 ②는 i++가 보인다는 이유만으로 증가식이라고 착각하게 하지만, for 문에서는 세미콜론 사이의 위치가 역할을 결정한다. ④는 반복문 형식은 맞지만 <를 사용해 합에 n을 포함하지 않는 함정이다.

다시 나오면 이렇게 판단

먼저 세미콜론을 기준으로 초기식·조건식·증감식의 위치를 확인한다. 그다음 범위의 끝값을 포함하면 <=, 포함하지 않으면 <인지 확인한다.

암기 포인트

for 문은 초기화-조건-증가의 순서이며, n까지 포함하는 조건은 i <= n이다.

NoteBase

4. 다음에 주어진 Java 프로그램의 Example 클래스 내부의 내용 중에서 컴파일 오류를 일으키는 잘못된 코드는?

```
class C {}
class CS extends C {}
interface I {}
class CI extends C implements I {}

class Example {
      static I  i  = new CI();
      static C  ca = new CI();
      static CS cs = new C();
      static C  cb = new CS();
}
```

① static I i = new CI();
② static C ca = new CI();
③ static CS cs = new C();
④ static C cb = new CS();

정답 3번

출제 포인트

참조 변수의 선언 자료형과 생성 객체의 자료형 사이에 대입 호환성이 있는지를 묻는 문제이다.

핵심 개념

Java에서 상위 클래스형 변수는 하위 클래스 객체를 참조할 수 있고, 인터페이스형 변수는 그 인터페이스를 구현한 클래스의 객체를 참조할 수 있다. 반대로 하위 클래스형 변수에 상위 클래스 객체를 그대로 대입할 수는 없다.
CI는 C의 하위 클래스이면서 I의 구현 클래스이고, CS는 C의 하위 클래스이다.

선지별 해설

① O — CI는 I를 구현하므로 I형 변수 i에 CI 객체를 대입할 수 있다.
② O — CI는 C를 상속하므로 C형 변수 ca에 하위 클래스인 CI 객체를 대입할 수 있다.
③ X — 틀린 부분은 CS형 변수 cs에 상위 클래스 C의 객체를 대입한 것이다. C 객체가 반드시 CS 객체인 것은 아니므로 컴파일 오류가 발생한다. 올바르게는 객체도 CS로 생성하여 static CS cs = new CS();로 쓰거나, 변수의 선언 자료형을 C로 바꾸어 static C cs = new C();로 써야 한다.
④ O — CS는 C를 상속하므로 C형 변수 cb에 하위 클래스인 CS 객체를 대입할 수 있다.

NoteBase

오답 포인트

③은 상속 관계가 있다는 사실만 보고 양방향 대입이 가능하다고 착각하게 만든다. 상위 타입 변수에 하위 객체를 담는 업캐스팅은 가능하지만, 하위 타입 변수에 상위 객체를 그대로 담는 대입은 불가능하다.

다시 나오면 이렇게 판단

대입문의 오른쪽 객체가 왼쪽 선언 자료형과 같거나 그 하위 타입인지 확인한다. 인터페이스형 변수라면 오른쪽 객체의 클래스가 그 인터페이스를 구현했는지 확인한다.

암기 포인트

상위 타입 변수에는 하위 객체를 담을 수 있지만, 하위 타입 변수에는 상위 객체를 그대로 담을 수 없다.

NoteBase

5. 무선 LAN의 종류 중에 가장 전송속도가 느린 것은?

① 802.11b ② 802.11a
③ 802.11g ④ 802.11n

정답 1번

출제 포인트

무선 LAN 표준인 IEEE 802.11a/b/g/n의 대표적인 최대 전송속도를 비교하는 문제이다.

핵심 개념

802.11b는 최대 11 Mbps, 802.11a와 802.11g는 최대 54 Mbps, 802.11n은 이론상 최대 600 Mbps를 지원한다. 따라서 네 규격 중 전송속도가 가장 느린 것은 802.11b이다.

선지별 해설

① O — 802.11b의 최대 전송속도는 11 Mbps로, 제시된 네 규격 중 가장 느리다.
② X — 802.11a의 최대 전송속도는 54 Mbps이므로 802.11b보다 빠르다. '가장 느린 것은 802.11a'가 아니라 '가장 느린 것은 802.11b'가 올바른 표현이다.
③ X — 802.11g의 최대 전송속도는 54 Mbps이므로 11 Mbps인 802.11b보다 빠르다. '가장 느린 것은 802.11g'가 아니라 '가장 느린 것은 802.11b'가 올바른 표현이다.
④ X — 802.11n은 다중 안테나 기술 등을 사용하여 이론상 최대 600 Mbps를 지원하므로 가장 느린 규격이 아니다. '가장 느린 것은 802.11n'이 아니라 '가장 느린 것은 802.11b'가 올바른 표현이다.

오답 포인트

802.11a는 알파벳 순서상 b보다 먼저 등장해 더 느릴 것처럼 보이지만, 대표 최대 전송속도는 54 Mbps이다. 이 문제에서는 표준명의 알파벳 순서가 아니라 각 규격의 최대 전송속도를 비교해야 한다.

다시 나오면 이렇게 판단

802.11a/b/g/n의 속도를 각각 54/11/54/600 Mbps로 대응한 뒤 최솟값을 찾는다. 최솟값 11 Mbps에 해당하는 규격은 802.11b이다.

암기 포인트

b는 11 Mbps, a·g는 54 Mbps, n은 최대 600 Mbps이다.

NoteBase

6. 인터넷 주소로 사용되는 IPv4 주소에 대한 설명 중 가장 옳지 않은 것은?
① IPv4는 32-bit로 구성되어 있으며 네트워크 주소와 호스트 주소로 구성된다.
② 255.255.255.0는 A클래스를 지정하기 위한 서브넷 마스크로 사용된다.
③ 192.168.1.100은 C클래스 사설 IP(내부 IP) 주소로 사용될 수 있다.
④ 공인 IP(외부 IP) 주소는 인터넷에서 유일한 주소로써 일반적으로 인터넷 서비스 공급자(ISP)가 제공한다.

정답 2번

출제 포인트

IPv4 주소의 구성, 클래스별 기본 서브넷 마스크, 사설 IP와 공인 IP의 차이를 구별하는 문제이다.

핵심 개념

IPv4 주소: 32비트 주소이며 서브넷 마스크를 기준으로 네트워크 부분과 호스트 부분을 구분한다.
클래스별 기본 마스크: 전통적인 클래스 기반 주소 체계에서 A 클래스는 255.0.0.0, B 클래스는 255.255.0.0, C 클래스는 255.255.255.0을 기본 마스크로 사용한다.
사설 IP와 공인 IP: 사설 IP는 내부 네트워크에서 사용하며 인터넷에서 직접 라우팅되지 않고, 공인 IP는 인터넷에서 식별되는 전역적으로 고유한 주소이다.

선지별 해설

① O — IPv4 주소는 32비트로 구성되며, 서브넷 마스크에 따라 네트워크 주소 부분과 호스트 주소 부분으로 나뉜다.
② X — 틀린 표현은 '255.255.255.0은 A 클래스를 지정하기 위한 서브넷 마스크'이다. 전통적인 클래스 기반 주소 체계에서 255.255.255.0은 C 클래스의 기본 서브넷 마스크이고, A 클래스의 기본 서브넷 마스크는 255.0.0.0이다.
③ O — 192.168.1.100은 사설 IPv4 대역인 192.168.0.0~192.168.255.255에 포함되며, 전통적인 클래스 구분으로는 C 클래스 범위에 속한다. 따라서 내부 IP 주소로 사용할 수 있다.
④ O — 공인 IP 주소는 인터넷에서 중복되지 않도록 전역적으로 고유하게 할당되는 외부 IP 주소이며, 일반적으로 인터넷 서비스 공급자(ISP)를 통해 제공받는다.

NoteBase

오답 포인트

②의 255.255.255.0을 자주 보았다는 이유로 A 클래스의 마스크라고 판단하기 쉽다. 클래스 기반 기본 마스크에서는 첫 번째 옥텟만 네트워크 부분인 A 클래스가 255.0.0.0이고, 앞의 세 옥텟이 네트워크 부분인 C 클래스가 255.255.255.0이다.

다시 나오면 이렇게 판단

클래스별 기본 마스크는 255로 채워진 옥텟의 수를 기준으로 판단한다. A 클래스는 1개, B 클래스는 2개, C 클래스는 3개이므로 각각 255.0.0.0, 255.255.0.0, 255.255.255.0이다. 사설 IP 여부는 10.0.0.0 대역, 172.16.0.0~172.31.255.255 대역, 192.168.0.0~192.168.255.255 대역에 포함되는지 확인한다.

암기 포인트

기본 마스크는 A=255.0.0.0, B=255.255.0.0, C=255.255.255.0이며, 192.168 대역은 대표적인 사설 IP 대역이다.

NoteBase

7. 인공지능 학습 데이터 등에 포함된 개인정보의 비식별화에 관한 다음의 설명 중 가장 옳지 않은 것은?
   ① k-익명성은 주어진 데이터 집합에서 같은 값이 적어도 k개 이상 존재하도록 하는 것을 말한다.
   ② ℓ-다양성은 비식별 조치 과정에서 ℓ개 이상 서로 다른 민감한 정보를 갖도록 동질 집합을 구성하는 것을 말한다.
   ③ t-근접성은 동질 집합에서 특정 정보의 분포와 전체 데이터 집합에서 정보의 분포가 t 이하의 차이를 보이게 하는 것을 말한다.
   ④ 정보주체의 동의 없이는 과학적 연구, 통계 작성, 공익적 기록보존을 목적으로라도 가명 정보를 활용할 수 없다.

정답 4번

출제 포인트

개인정보 비식별화 모델인 k-익명성, ℓ-다양성, t-근접성의 판단 기준과 가명정보의 활용 범위를 구별하는 문제이다.

핵심 개념

k-익명성: 준식별자 속성이 같은 레코드가 적어도 k개 존재하도록 하여 특정 개인을 구별하기 어렵게 하는 원칙이다.
ℓ-다양성: 각 동질 집합 안에 서로 다른 민감한 정보가 적어도 ℓ개 존재하도록 하는 원칙이다.
t-근접성: 동질 집합의 민감정보 분포와 전체 데이터의 민감정보 분포 사이의 거리가 t 이하가 되도록 하는 원칙이다.
가명정보: 추가 정보 없이는 특정 개인을 알아볼 수 없도록 처리한 개인정보이며, 정해진 목적에서는 정보주체의 동의 없이 처리할 수 있다.

NoteBase

선지별 해설

① O — k-익명성은 같은 준식별자 값을 가진 레코드가 적어도 k개가 되게 하여 한 개인이 k명 미만으로 좁혀지지 않도록 하는 개념이다.
② O — ℓ-다양성은 하나의 동질 집합 안에 서로 다른 민감정보 값이 적어도 ℓ개 존재하게 하여 민감정보가 쉽게 추론되지 않도록 하는 개념이다.
③ O — t-근접성은 동질 집합의 민감정보 분포가 전체 데이터의 민감정보 분포에서 지나치게 벗어나지 않도록 두 분포의 거리를 t 이하로 제한하는 개념이다.
④ X — 틀린 표현은 '정보주체의 동의 없이는 … 가명정보를 활용할 수 없다'이다. 올바르게는 가명정보를 과학적 연구, 통계작성, 공익적 기록보존 등의 목적으로 처리하는 경우 정보주체의 동의 없이 활용할 수 있다.

오답 포인트

'가명정보도 개인정보이므로 언제나 동의가 필요하다'고 단정하면 ④를 놓치기 쉽다. 가명정보는 개인정보에 해당하지만, 과학적 연구·통계작성·공익적 기록보존이라는 허용 목적에서는 동의 없는 처리가 가능하다는 예외가 핵심이다.

다시 나오면 이렇게 판단

비식별화 모델은 집단의 크기인지, 민감정보 값의 다양성인지, 전체 분포와의 거리인지로 구분한다. 가명정보 활용 선지는 '항상 동의가 필요한가'와 '허용된 목적에 해당하는가'를 나누어 판단한다.

암기 포인트

k는 집단 크기, ℓ은 민감값 다양성, t는 분포 거리이며, 가명정보는 연구·통계·공익 기록 목적이면 동의 없이 처리할 수 있다.

NoteBase

8. 소프트웨어 디자인 패턴(design pattern)에 대한 설명 중 가장 옳지 않은 것은?
① 생성(creational)패턴, 구조(structural)패턴, 행위(behavioral)패턴 등으로 분류할 수 있다.
② 각기 다른 소프트웨어 모듈이나 기능 간의 설계 또는 해결책 간의 공통되는 요소를 재사용할 수 있게 해준다.
③ 소스나 기계 코드로 바로 전환될 수 있는 완성된 디자인을 제공함으로써 소프트웨어 개발 효율을 향상하게 한다.
④ 어댑터 패턴(adapter pattern)은 클래스의 인터페이스를 다른 인터페이스로 변환하는 데 활용되며, 비호환적 인터페이스를 갖춘 클래스들이 함께 작동하도록 지원한다.

정답 3번

출제 포인트

디자인 패턴의 분류와 목적, 그리고 디자인 패턴이 완성된 코드가 아니라 반복되는 설계 문제에 대한 재사용 가능한 해결 구조라는 점을 구별하는 문제이다.

핵심 개념

디자인 패턴은 소프트웨어 설계에서 반복적으로 나타나는 문제에 적용할 수 있도록 검증된 해결 방법의 구조와 지침을 정리한 것이다.
생성 패턴은 객체 생성 방식을, 구조 패턴은 클래스와 객체의 조합 방식을, 행위 패턴은 객체 사이의 책임 분배와 상호작용 방식을 다룬다.
디자인 패턴은 재사용 가능한 설계 지식이지, 그대로 소스나 기계 코드로 변환되는 완성 프로그램은 아니다.

선지별 해설

① O — 디자인 패턴은 해결 대상에 따라 생성 패턴, 구조 패턴, 행위 패턴으로 분류할 수 있다.
② O — 반복되는 설계 문제의 공통 구조와 해결 방법을 패턴으로 정리하므로 서로 다른 모듈이나 기능을 설계할 때에도 해당 설계 지식을 재사용할 수 있다.
③ X — 틀린 표현은 '소스나 기계 코드로 바로 전환될 수 있는 완성된 디자인'이다. 디자인 패턴은 특정 언어의 완성 코드가 아니라 상황에 맞게 구체화하여 구현해야 하는 재사용 가능한 설계상의 해결 구조이다.
④ O — 어댑터 패턴은 한 클래스의 인터페이스를 클라이언트가 기대하는 다른 인터페이스로 변환하여, 원래 인터페이스가 호환되지 않는 클래스들이 함께 작동하게 하는 구조 패턴이다.

NoteBase

오답 포인트

③의 '재사용 가능'이라는 취지는 맞아 보이지만, 디자인 패턴 자체를 '소스나 기계 코드로 바로 전환되는 완성된 디자인'이라고 한 부분이 함정이다. 재사용되는 것은 완성 코드가 아니라 문제 해결을 위한 설계 구조와 아이디어이다.

다시 나오면 이렇게 판단

디자인 패턴을 설명하는 선지에서는 '반복 문제에 대한 일반화된 설계 해법'인지 확인한다. 그대로 실행하거나 즉시 코드로 변환되는 완성품이라고 하면 틀리고, 상황에 맞게 구현해야 하는 설계 지침이라고 하면 옳다.

암기 포인트

디자인 패턴은 완성 코드가 아니라 재사용하는 설계 해법이며, 분류는 생성·구조·행위이다.

NoteBase

9. 객체지향 프로그래밍(object-oriented programming)과 관련된 다음의 설명 중 가장 옳지 않은 것은?
① 클래스 내의 세부 구현에 대한 접근 제한을 통해 정보 은닉(information hiding) 효과를 달성할 수 있다.
② 클래스 간의 상속(inheritance) 기능을 써서 부모 클래스의 특성과 기능을 물려받을 수 있으며, 오버라이딩(overriding)을 통하여 일부만을 변경할 수도 있다.
③ 다형성(polymorphism)을 활용하면 하나의 변수나 함수의 의미를 상황에 따라 다르게 해석할 수 있게 된다.
④ 객체지향 프로그래밍은 실행 속도가 빨라지는 장점이 있으나 코드의 재사용성이 절차적 프로그래밍 방식에 비하여 저하하는 단점이 있다.

정답 4번

출제 포인트

객체지향 프로그래밍의 핵심 특성인 정보 은닉, 상속, 다형성과 객체지향 방식의 일반적인 장단점을 구분하는 문제이다.

핵심 개념

정보 은닉: 객체 내부의 구현 세부 사항을 외부에서 직접 접근하지 못하게 제한하여 데이터를 보호하는 특성이다. / 키워드: 접근 제한
상속: 기존 클래스의 속성과 기능을 새로운 클래스가 물려받아 확장하거나 재사용하는 특성이다. / 키워드: 재사용과 확장
다형성: 같은 이름이나 인터페이스가 객체나 상황에 따라 서로 다른 동작을 나타내는 특성이다. / 키워드: 하나의 형식, 여러 동작
객체지향 방식: 모듈화와 상속 등을 통해 코드의 재사용성과 유지보수성을 높일 수 있지만, 추상화와 동적 바인딩 등에 따른 부가 비용 때문에 실행 속도가 반드시 빨라지는 것은 아니다.

NoteBase

선지별 해설

① O — 클래스 내부의 세부 구현에 대한 외부 접근을 제한하면 데이터와 구현을 감출 수 있으므로 정보 은닉 효과를 얻을 수 있다.
② O — 상속을 이용하면 부모 클래스의 특성과 기능을 물려받을 수 있고, 오버라이딩을 통해 상속받은 메서드의 동작을 자식 클래스에 맞게 변경할 수 있다.
③ O — 다형성을 활용하면 같은 변수의 참조 대상이나 같은 함수·메서드 호출이 실제 객체와 상황에 따라 서로 다른 의미와 동작을 나타낼 수 있다.
④ X — 틀린 표현은 '실행 속도가 빨라지는 장점'과 '코드의 재사용성이 절차적 프로그래밍 방식에 비하여 저하'된다는 부분이다. 올바르게는 객체지향 프로그래밍은 일반적으로 코드의 재사용성이 높지만, 추상화·동적 바인딩 등의 부가 비용으로 실행 속도가 절차적 프로그래밍보다 반드시 빠른 것은 아니라고 해야 한다.

오답 포인트

④는 객체지향의 장단점을 서로 뒤바꾸어 제시한다. 상속과 모듈화는 코드 재사용성을 높이는 요소이며, 객체지향이라는 이유만으로 실행 속도가 빨라지는 것은 아니다.

다시 나오면 이렇게 판단

접근 제한은 정보 은닉, 부모의 특성 승계는 상속, 같은 호출의 다양한 동작은 다형성으로 판단한다. 성능 비교에서는 객체지향의 대표 장점을 재사용성과 유지보수성으로 보고, 실행 속도 향상을 일반적 장점으로 단정한 표현은 경계한다.

암기 포인트

객체지향은 '숨기고, 물려받고, 다르게 동작하며, 재사용한다'로 정리한다.

NoteBase

10. 다음 중 DBMS의 필수 기능으로 가장 옳지 않은 것은?
① 이식(migration) 기능
② 정의(definition) 기능
③ 조작(manipulation) 기능
④ 제어(control) 기능

정답 1번

출제 포인트

DBMS의 세 가지 필수 기능인 정의, 조작, 제어를 구별하는 문제이다.

핵심 개념

정의 기능: 데이터베이스의 구조, 데이터 형식, 제약 조건 등을 정의하는 기능이다.
조작 기능: 데이터를 검색·삽입·삭제·수정하는 기능이다.
제어 기능: 데이터의 무결성·보안·권한·병행 처리·회복 등을 관리하는 기능이다.
DBMS의 필수 기능은 정의·조작·제어이며, 이식은 이 세 가지에 포함되지 않는다.

선지별 해설

① X — '이식(migration) 기능'은 DBMS의 세 가지 필수 기능에 해당하지 않는다. 틀린 표현은 '이식'이며, 올바르게는 DBMS의 필수 기능을 '정의·조작·제어 기능'으로 구분한다.
② O — 정의(definition) 기능은 데이터베이스의 논리적·물리적 구조와 제약 조건 등을 명세하므로 DBMS의 필수 기능이다.
③ O — 조작(manipulation) 기능은 사용자가 데이터를 검색·삽입·삭제·수정할 수 있게 하므로 DBMS의 필수 기능이다.
④ O — 제어(control) 기능은 데이터의 정확성과 안전성을 유지하고 여러 사용자의 접근을 관리하므로 DBMS의 필수 기능이다.

오답 포인트

①의 '이식'은 시스템이나 데이터를 다른 환경으로 옮긴다는 의미로 쓰일 수 있어 DBMS 기능처럼 보이지만, 교과서에서 분류하는 DBMS의 필수 기능은 아니다. 또한 문제의 부정형 표현인 '옳지 않은 것'을 놓치면 ②~④ 중 하나를 고르기 쉽다.

다시 나오면 이렇게 판단

선지의 용어가 정의·조작·제어 중 하나인지 확인한다. 구조를 정하면 정의, 데이터를 처리하면 조작, 무결성·보안·권한 등을 관리하면 제어로 판단한다.

NoteBase

암기 포인트

DBMS 필수 기능은 '정·조·제', 즉 정의·조작·제어이다.

NoteBase

11. 데이터베이스에서 데이터의 중복성으로 발생하는 다음의 문제와 설명이 가장 옳지 않은 것은?
① 일관성: 중복된 데이터 간에 내용이 일치하지 않는 상황의 발생
② 보안성: 중복된 모든 데이터에 동등한 보안 수준 유지에 어려움
③ 경제성: 저장 공간의 낭비와 반복 작업으로 비용의 증가
④ 무결성: 제어의 분산으로 연계된 응용프로그램 간의 불일치

정답 4번

출제 포인트

데이터 중복이 일관성·보안성·경제성·무결성에 미치는 문제를 구분하는 문항이다.

핵심 개념

데이터 중복은 같은 데이터를 여러 곳에 반복 저장하는 상태이다. 중복 데이터 가운데 일부만 수정되면 값이 서로 달라져 일관성이 깨지고, 보안 통제와 무결성 제약의 적용 범위가 늘어나며, 저장 공간과 처리 비용도 증가한다. 일관성은 여러 곳의 동일 데이터가 서로 모순되지 않는 성질이고, 무결성은 데이터가 정해진 제약조건을 만족하며 정확성과 유효성을 유지하는 성질이다.

선지별 해설

① O — 같은 데이터가 중복 저장되면 한쪽만 갱신되는 경우가 생겨 중복된 데이터의 내용이 서로 일치하지 않을 수 있으므로 일관성 문제가 발생한다.
② O — 동일한 데이터가 여러 위치에 존재하면 모든 사본에 같은 접근 통제와 보안 수준을 적용하기 어려워 보안 관리 부담이 커진다.
③ O — 중복 저장은 저장 공간을 낭비하고 동일한 입력·갱신·관리 작업을 반복하게 하므로 비용을 증가시킨다.
④ X — 틀린 표현은 '제어의 분산으로 연계된 응용프로그램 간의 불일치'이다. 이는 데이터 무결성의 설명이 아니다. 올바르게는 중복된 데이터의 일부가 잘못 갱신되거나 제약조건이 일관되게 적용되지 않아 데이터의 정확성과 유효성을 유지하기 어려워진다고 설명해야 한다.

오답 포인트

①의 '내용이 일치하지 않음'과 ④의 '불일치'라는 표현이 비슷해 보이는 점이 함정이다. 일관성 문제는 중복된 데이터 값 사이의 불일치이고, 무결성 문제는 데이터가 제약조건을 만족하며 정확성과 유효성을 유지하는지에 관한 문제이다. ④는 불일치의 대상을 데이터가 아니라 응용프로그램으로 잡았으므로 가장 옳지 않다.

NoteBase

다시 나오면 이렇게 판단

중복성의 영향을 판단할 때 불일치의 대상부터 확인한다. 데이터 사본끼리 값이 다르면 일관성, 데이터가 제약조건을 위반하면 무결성, 통제 범위가 늘어나면 보안성, 공간과 작업이 늘어나면 경제성 문제로 판단한다.

암기 포인트

중복은 값의 불일치, 통제의 확대, 비용의 증가, 제약 유지의 어려움을 낳는다. 무결성은 응용프로그램의 일치가 아니라 데이터의 정확성·유효성이다.

NoteBase

12. DMA(Direct Memory Access) 입출력 기능에 대한 다음의 설명 중 가장 옳지 않은 것은?
① 하드웨어 하위시스템이 CPU와 독립적으로 메인시스템 메모리에 접근할 수 있게 해주는 기능이다.
② 장치 컨트롤러가 데이터의 한 블록을 이동시키는데 이 과정에서 CPU의 개입은 불필요하다.
③ 데이터가 전송되는 동안 CPU는 다른 작업을 수행할 수 있게 되어 CPU 효율성이 향상된다.
④ 메모리 간 복사 또는 데이터 이동에는 쓰일 수 없는 한계가 있다.

정답 4번

출제 포인트

DMA가 CPU 대신 데이터 전송을 담당하는 원리와 전송 가능한 범위를 묻는 문제이다.

핵심 개념

DMA는 CPU가 매번 데이터를 옮기지 않아도 DMA 컨트롤러가 시스템 버스를 사용하여 주기억장치와 입출력장치 사이의 블록 데이터를 직접 전송하게 하는 방식이다. CPU는 전송 조건을 설정하고 완료 결과를 확인하지만, 실제 블록 전송이 진행되는 동안에는 다른 작업을 수행할 수 있다. DMA 컨트롤러가 지원하면 메모리의 한 영역에서 다른 영역으로 데이터를 옮기는 메모리 간 전송도 가능하다.

선지별 해설

① O — DMA 하드웨어는 버스 사용 권한을 얻어 CPU가 데이터를 하나씩 중계하지 않아도 주기억장치에 직접 접근할 수 있다.
② O — DMA 컨트롤러는 설정된 데이터 블록을 전송하는 동안 CPU의 지속적인 개입을 필요로 하지 않는다. 다만 전송 전의 설정과 전송 완료 후의 확인까지 CPU가 전혀 관여하지 않는다는 뜻은 아니다.
③ O — DMA가 실제 전송을 담당하는 동안 CPU는 다른 명령을 처리할 수 있으므로 CPU의 부담이 줄고 시스템 효율성이 향상된다.
④ X — 틀린 표현은 '메모리 간 복사 또는 데이터 이동에는 쓸 수 없는 한계가 있다'이다. 올바르게는 DMA 컨트롤러가 해당 기능을 지원하면 메모리의 한 영역에서 다른 영역으로 데이터를 복사하거나 이동하는 데에도 DMA를 사용할 수 있다.

NoteBase

오답 포인트

②의 ‘CPU의 개입은 불필요하다’를 DMA 동작 전체에서 CPU가 전혀 필요 없다는 뜻으로 확대하면 오답이 된다. CPU는 전송을 설정하고 완료를 확인하며, 개별 데이터 전송 과정만 DMA 컨트롤러가 맡는다. 반대로 ④는 메모리 간 전송이 절대 불가능하다고 단정한 표현이 틀렸다.

다시 나오면 이렇게 판단

DMA 문항에서는 CPU의 역할을 ‘전송 설정·완료 확인’, DMA 컨트롤러의 역할을 ‘실제 블록 전송’으로 나누어 판단한다. 또한 ‘전혀 불가능하다’, ‘쓸 수 없다’와 같은 절대 표현은 DMA 컨트롤러가 지원하는 전송 방식이 있는지 확인한다.

암기 포인트

DMA는 CPU가 설정하고, DMA 컨트롤러가 블록을 직접 전송하며, CPU는 그동안 다른 작업을 수행한다.

NoteBase

13. 웹에 대한 다음 설명 중 가장 옳지 않은 것은?
① HTTP는 네트워크 계층의 프로토콜로서 확장이 가능하다.
② HTTP는 브라우저가 웹 서버와 통신하기 위해 많이 사용된다.
③ HTTP 클라이언트와 서버 간에 연결 상태를 유지하지 않는 프로토콜이다.
④ URL은 웹 페이지, 이미지, 비디오 등 리소스의 위치를 가리키는 문자열이다.

정답 1번

출제 포인트

HTTP의 계층상 위치와 무상태성, URL의 의미를 정확히 구분할 수 있는지를 묻는 문제이다.

핵심 개념

HTTP: 웹에서 클라이언트와 서버가 요청과 응답을 주고받을 때 사용하는 응용 계층 프로토콜이다. 메시지 구조와 헤더를 확장할 수 있으며, 기본적으로 각 요청을 독립적으로 처리하는 무상태 프로토콜이다.
URL: 웹 페이지, 이미지, 비디오 등 인터넷 자원의 위치를 나타내는 문자열이다.

선지별 해설

① X — 틀린 표현은 '네트워크 계층의 프로토콜'이다. HTTP는 응용 계층 프로토콜이며, 헤더 필드나 메서드 등을 통해 확장 가능한 구조를 가진다. 따라서 'HTTP는 응용 계층의 프로토콜로서 확장이 가능하다'로 고쳐야 한다.
② O — HTTP는 웹 브라우저와 웹 서버가 요청과 응답을 주고받을 때 널리 사용되는 프로토콜이다.
③ O — HTTP는 기본적으로 서버가 이전 요청의 연결 상태를 계속 유지하지 않는 무상태 프로토콜이다. 필요한 상태 정보는 쿠키나 세션 등의 별도 방법으로 관리한다.
④ O — URL은 웹 페이지, 이미지, 비디오 등 자원이 있는 위치를 가리키는 문자열이다.

출제자가 노린 함정

①은 '확장이 가능하다'는 뒷부분이 맞기 때문에 전체 문장을 옳다고 판단하기 쉽다. 그러나 HTTP를 '네트워크 계층'으로 분류한 부분이 틀렸으며, HTTP는 응용 계층에 속한다.

다시 나오면 이렇게 판단

프로토콜 설명에서는 먼저 계층을 확인한다. HTTP는 응용 계층, IP는 네트워크 계층, TCP는 전송 계층으로 구분하고, HTTP의 상태 특성은 '기본적으로 무상태'라고 판단한다.

NoteBase

암기 포인트

HTTP는 응용 계층·요청과 응답·무상태, URL은 자원의 위치이다.

NoteBase

14. 어떤 프로그램이 단일 프로세서를 이용하여 실행하면 100초가 소요되며 이 프로그램의 40%는 반드시 순차적으로 실행되어야 한다고 가정했을 때, 4개의 프로세서를 이용하여 실행할 경우 이론적으로 가능한 최단 시간에 가장 가까운 것으로 가장 옳은 것은?

① 25초 ② 55초
③ 75초 ④ 100초

정답 2번

출제 포인트

이 문제는 프로그램의 순차 실행 부분과 병렬 실행 가능 부분을 구분하여 4개 프로세서를 사용할 때의 이론적 최단 실행 시간을 구하는 문제이다.

조건 정리

주어진 값: 단일 프로세서 실행 시간은 100초이고, 순차 실행 비율은 40%이며, 프로세서 수는 4개이다.
구해야 할 값: 4개 프로세서를 사용할 때 이론적으로 가능한 최단 실행 시간이다.
주의할 조건: 순차 실행 부분은 프로세서가 늘어도 줄어들지 않으며, 나머지 부분만 4개 프로세서에 균등하게 나뉜다고 가정한다.

사용할 공식

순차 실행 시간을 T_s, 병렬 실행 가능 시간을 T_p, 프로세서 수를 N이라 하면 전체 실행 시간은 다음과 같다.
$T = T_s + \frac{T_p}{N}$
순차 부분은 병렬화할 수 없고 병렬 실행 가능 부분만 4개 프로세서가 동시에 나누어 처리하므로 이 공식을 사용한다.

계산 과정

1. 순차 실행 시간은 100 × 0.4 = 40초이다.
2. 병렬 실행 가능 시간은 100 × 0.6 = 60초이다.
3. 이 부분을 4개 프로세서로 나누면 60 ÷ 4 = 15초이다.
4. 따라서 이론적 최단 실행 시간은 다음과 같다.

40+15=55초

NoteBase

선지 판단

① X — 25초는 전체 100초를 단순히 4로 나눈 값으로, 병렬화할 수 없는 40초를 무시한 결과이다. 올바르게는 순차 부분 40초와 병렬 부분 15초를 더한 55초이다.
② O — 계산 결과가 55초이므로 이론적으로 가능한 최단 시간과 일치한다.
③ X — 75초는 순차 부분과 병렬 부분을 조건에 맞게 계산한 값이 아니다. 올바르게는 40 + 60 ÷ 4 = 55초이다.
④ X — 100초는 4개 프로세서에 의한 병렬 실행 시간 단축을 전혀 반영하지 않은 값이다. 올바르게는 병렬 실행 가능한 60초가 15초로 줄어들므로 총 55초이다.

오답 포인트

①의 25초처럼 전체 실행 시간을 프로세서 수로 곧바로 나누는 것이 핵심 함정이다. 반드시 순차 실행 부분 40초는 그대로 두고, 병렬 실행 가능한 나머지 60초만 4로 나누어야 한다.

다시 나오면 이렇게 판단

전체 시간을 순차 부분과 병렬 가능 부분으로 먼저 분리한 뒤, 병렬 가능 부분만 프로세서 수로 나누고 마지막에 순차 부분을 더한다.

암기 포인트

병렬 처리 시간은 '순차 시간 + 병렬 가능 시간 ÷ 프로세서 수'이다.

NoteBase

15. 캐시 메모리에 대한 다음의 설명 중 가장 옳지 않은 것은?
① CPU 캐시 메모리는 주기억 장치와 CPU 사이에 위치하여 두 장치 간의 속도 차에 의한 데이터 처리 지연을 해소한다.
② 주기억 장치에 비하여 용량은 적으나 속도는 빠른 메모리를 사용한다.
③ 멀티 코어 프로세서의 경우 각 코어별로 독립된 캐시메모리를 가질 수 없다.
④ 캐시 메모리는 데이터 지역성의 원리를 이용하여 작동한다.

정답 3번

출제 포인트

캐시 메모리의 위치·속도·용량·지역성 원리와 멀티코어 프로세서의 캐시 구성 방식을 구별하는 문제이다.

핵심 개념

캐시 메모리는 CPU가 자주 사용할 데이터와 명령어를 임시로 저장하여 주기억 장치 접근에 따른 지연을 줄이는 고속·소용량 메모리이다.
멀티코어 프로세서에서는 코어마다 전용 캐시를 둘 수 있고, 여러 코어가 공유하는 캐시를 함께 구성할 수도 있다.
캐시는 최근 사용한 데이터가 다시 사용될 가능성이 높은 시간적 지역성과 가까운 주소의 데이터가 사용될 가능성이 높은 공간적 지역성을 이용한다.

선지별 해설

① O — CPU 캐시 메모리는 CPU와 주기억 장치 사이의 속도 차이로 생기는 메모리 접근 지연을 줄인다.
② O — 캐시는 주기억 장치보다 용량은 작지만 접근 속도는 빠른 메모리를 사용한다.
③ X — 틀린 표현은 '각 코어별로 독립된 캐시메모리를 가질 수 없다'이다. 올바르게는 멀티코어 프로세서의 각 코어가 독립된 전용 캐시를 가질 수 있으며, 캐시 계층에 따라 코어들이 공유하는 캐시도 둘 수 있다.
④ O — 캐시 메모리는 시간적 지역성과 공간적 지역성을 포함하는 데이터 지역성의 원리를 이용하여 작동한다.

오답 포인트

멀티코어라는 이유로 모든 캐시가 반드시 공유된다고 생각하기 쉽다. 그러나 캐시는 코어별 전용 캐시와 코어 간 공유 캐시를 함께 사용하는 구조가 가능하므로, ③의 '가질 수 없다'라는 단정이 함정이다.

NoteBase

다시 나오면 이렇게 판단

캐시 관련 선지는 ‘CPU와 주기억 장치 사이의 속도 차이 완화’, ‘고속·소용량’, ‘지역성 활용’을 기본 특징으로 판단한다. 멀티코어 캐시 구성은 코어별 전용과 코어 간 공유가 모두 가능하므로 ‘반드시 공유’ 또는 ‘전용 캐시 불가’ 같은 단정은 틀린 표현으로 판단한다.

암기 포인트

캐시는 고속·소용량·지역성, 멀티코어 캐시는 전용과 공유가 모두 가능하다.

NoteBase

16. 다중 프로그램 실행 환경에서 일련의 프로세스들이 서로가 가진 자원을 무한정 기다리며 더 이상 진행이 될 수 없는 상태를 일컫는 용어로 가장 옳은 것은?
① deadlock
② pipelining
③ preemption
④ mutual exclusion

정답 1번

출제 포인트

여러 프로세스가 상대방이 가진 자원을 기다리면서 모두 더 이상 진행하지 못하는 교착 상태의 정의를 묻는 문제이다.

핵심 개념

데드록(deadlock): 둘 이상의 프로세스가 서로 점유한 자원을 기다리며 무한정 대기하는 상태이다.
파이프라이닝(pipelining): 작업을 여러 단계로 나누고 서로 다른 작업의 단계를 겹쳐 처리하여 처리량을 높이는 방식이다.
선점(preemption): 운영체제가 실행 중인 프로세스에서 CPU나 자원의 사용권을 강제로 회수하는 것이다.
상호 배제(mutual exclusion): 하나의 자원을 한 번에 하나의 프로세스만 사용하도록 하는 조건이다.

선지별 해설

① O — deadlock은 여러 프로세스가 서로 점유한 자원을 기다리며 무한정 대기하여 진행할 수 없는 상태이므로 제시된 설명과 정확히 일치한다.
② X — pipelining은 작업 단계를 겹쳐 처리하는 기법이지 프로세스들이 서로 자원을 기다리는 상태가 아니다. 틀린 표현인 ‘pipelining’은 ‘deadlock’으로 고쳐야 한다.
③ X — preemption은 운영체제가 프로세스의 CPU나 자원 사용권을 강제로 회수하는 행위이지 상호 자원 대기로 진행이 멈춘 상태가 아니다. 틀린 표현인 ‘preemption’은 ‘deadlock’으로 고쳐야 한다.
④ X — mutual exclusion은 자원을 한 번에 하나의 프로세스만 사용하게 하는 조건이며, 교착 상태가 발생할 수 있는 필요조건 중 하나일 뿐 교착 상태 자체를 뜻하지 않는다. 틀린 표현인 ‘mutual exclusion’은 ‘deadlock’으로 고쳐야 한다.

오답 포인트

④의 mutual exclusion은 데드록의 발생 조건과 관련되어 있어 정답처럼 보일 수 있다. 그러나 ‘서로 가진 자원을 무한정 기다려 더 이상 진행하지 못함’은 조건이 아니라 이미 발생한 상태를 설명하므로 deadlock이다.

NoteBase

다시 나오면 이렇게 판단

‘서로가 가진 자원을 기다림’, ‘무한정 대기’, ‘더 이상 진행 불가’가 함께 제시되면 deadlock으로 판단한다. 자원의 단독 사용 조건은 mutual exclusion, 실행권의 강제 회수는 preemption, 단계의 중첩 처리는 pipelining으로 구분한다.

암기 포인트

서로 자원을 쥔 채 상대 자원을 기다려 멈추면 deadlock이다.

NoteBase

17. 캐시 기억 장치에서 교체 기법이 불필요하고 캐시 효율이 낮아질 수 있는 사상 방식으로 가장 옳은 것은?

① 간접 ② 직접

③ 완전 – 연관 ④ 세트 – 연관

정답 2번

출제 포인트

캐시 기억 장치의 사상 방식별로 주기억 장치 블록이 들어갈 수 있는 캐시 라인의 범위와 교체 알고리즘의 필요 여부를 구분하는 문제이다.

핵심 개념

직접 사상: 각 주기억 장치 블록이 들어갈 캐시 라인이 하나로 고정되는 방식이다. 교체 위치를 선택할 필요가 없어 교체 알고리즘이 불필요하지만, 같은 라인에 대응하는 블록끼리 충돌이 자주 발생하여 캐시 효율이 낮아질 수 있다.

완전-연관 사상: 주기억 장치의 어느 블록이나 캐시의 어느 라인에도 저장할 수 있는 방식이다. 공간 활용은 유연하지만 캐시가 찼을 때 교체 대상을 정해야 한다.

세트-연관 사상: 캐시를 여러 세트로 나누고, 블록을 정해진 세트 안의 어느 라인에나 저장하는 방식이다. 직접 사상과 완전-연관 사상의 절충형이며 세트 안에서 교체 대상을 정해야 한다.

선지별 해설

① X — '간접'은 제시된 두 특징에 해당하지 않는다. 틀린 표현 '간접'은 '직접'으로 고쳐야 한다.

② O — 직접 사상은 블록마다 들어갈 캐시 라인이 고정되므로 교체 기법이 필요 없다. 다만 서로 다른 블록이 같은 라인에 대응하면 충돌 미스가 발생하여 캐시 효율이 낮아질 수 있다.

③ X — 완전-연관 사상은 블록을 어느 캐시 라인에나 저장할 수 있으므로 캐시가 가득 차면 교체할 라인을 선택하는 기법이 필요하다. 틀린 표현 '완전-연관'은 '직접'으로 고쳐야 한다.

④ X — 세트-연관 사상은 블록이 들어갈 세트는 정해지지만 그 세트 안에서는 여러 라인 중 하나를 사용하므로, 빈 라인이 없으면 교체 대상을 선택해야 한다. 틀린 표현 '세트-연관'은 '직접'으로 고쳐야 한다.

오답 포인트

'교체 기법이 불필요하다'는 말만 보고 저장 위치가 유연한 완전-연관 또는 세트-연관 사상을 고르기 쉽다. 저장 위치가 유연할수록 캐시가 찼을 때 어느 라인을 비울지 결정해야 하며, 저장 위치가 하나로 고정된 직접 사상만 교체 대상을 선택하지 않는다.

NoteBase

다시 나오면 이렇게 판단

주기억 장치 블록이 들어갈 수 있는 위치의 수를 확인한다. 위치가 한 곳으로 고정되면 직접 사상으로 판단하고, 교체 알고리즘은 불필요하지만 충돌 미스가 많아질 수 있다고 연결한다.

암기 포인트

직접 사상은 '한 블록당 한 자리'이므로 교체 선택은 없고 충돌 가능성은 크다.

NoteBase

18. 다음에서 설명하는 RAID 방식으로 옳은 것은?

- 모든 데이터를 실시간으로 복구가 가능해 디스크 오류가 발생해도 중요한 데이터를 즉시 사용할 수 있다.
- 시스템 소프트웨어나 중요한 파일을 저장하는 용도로만 사용된다.

① RAID 0 ② RAID 1
③ RAID 2 ④ RAID 5

정답 2번

출제 포인트

RAID 0, RAID 1, RAID 2, RAID 5를 데이터 저장 방식과 장애 복구 방식으로 구별하는 문제이다. 제시문의 핵심 단서는 디스크에 같은 데이터를 복제하여 장애가 발생해도 즉시 사용할 수 있다는 점이다.

핵심 개념

RAID 0은 데이터를 여러 디스크에 나누어 기록하는 스트라이핑 방식으로, 속도는 높지만 중복 저장 정보가 없어 장애 복구 기능이 없다.
RAID 1은 같은 데이터를 두 개 이상의 디스크에 복제하는 미러링 방식으로, 저장 공간의 효율은 낮지만 한 디스크가 고장 나도 복제본을 즉시 사용할 수 있다.
RAID 2는 비트 단위 스트라이핑과 해밍 코드 기반 오류 정정 정보를 사용하는 방식이다.
RAID 5는 블록 단위 스트라이핑과 분산 패리티를 사용하여 한 디스크의 장애를 복구할 수 있는 방식이다.

선지별 해설

① X — RAID 0은 데이터를 나누어 기록하는 스트라이핑 방식이므로 동일한 데이터의 복제본이 없고, 디스크 장애가 발생하면 자체적으로 데이터를 복구할 수 없다. 틀린 표현인 '실시간으로 복구가 가능'은 '중복 정보가 없어 장애 복구가 불가능'으로 고쳐야 한다.
② O — RAID 1은 동일한 데이터를 여러 디스크에 동시에 기록하는 미러링 방식이다. 한 디스크에 오류가 발생해도 다른 디스크의 복제본을 즉시 사용할 수 있어 시스템 파일이나 중요한 데이터처럼 신뢰성이 필요한 저장 용도에 적합하다.
③ X — RAID 2는 비트 단위로 데이터를 분산하고 해밍 코드로 오류를 정정하는 방식이지, 동일한 데이터를 복제하여 즉시 사용하는 미러링 방식이 아니다. 틀린 표현인 '중요한 데이터를 즉시 사용'하는 복제 방식은 RAID 1로 고쳐야 한다.
④ X — RAID 5는 데이터와 분산 패리티를 여러 디스크에 나누어 저장하여 한 디스크 장애 시 데이터를 재구성하는 방식이다. 동일 데이터의 완전한 복제본을 두는 방식이 아니므로, 틀린 표현인 '모든 데이터를 실시간으로 복구'하는 미러링 방식은 RAID 1로 고쳐야 한다.

NoteBase

오답 포인트

RAID 5도 한 디스크의 장애를 복구할 수 있어 ④와 혼동하기 쉽다. 그러나 제시문의 '중요한 데이터를 즉시 사용'이라는 표현은 별도의 복제본을 그대로 이용하는 RAID 1을 가리킨다. RAID 5는 패리티를 이용해 손실 데이터를 재구성한다는 점이 다르다.

다시 나오면 이렇게 판단

동일 데이터 복제·미러링·장애 즉시 사용이면 RAID 1, 속도 향상을 위한 단순 분산이면 RAID 0, 해밍 코드이면 RAID 2, 분산 패리티와 한 디스크 장애 복구이면 RAID 5로 판단한다.

암기 포인트

RAID 0은 나누기, RAID 1은 똑같이 복제하기, RAID 2는 해밍 코드, RAID 5는 분산 패리티이다.

NoteBase

19. 다음 중 파이썬 코드의 실행 결과로 옳은 것은?

```
list = [1, 2, 3, 4, 5, 6]
print(list[-2:-4:-1])
```

① [4, 3]　② [4, 3, 2]
③ [5, 4]　④ [5, 4, 3]

정답 3번

출제 포인트

이 문제는 파이썬 슬라이싱 list[start:stop:step]에서 음수 인덱스와 음수 간격을 적용해 출력 결과를 구하는 문제이다.

조건 정리

주어진 리스트: [1, 2, 3, 4, 5, 6]
슬라이스: list[-2:-4:-1]
주의할 조건: step이 -1이므로 뒤에서 앞으로 이동하며, stop 위치의 원소는 결과에 포함하지 않는다.

사용할 공식

슬라이싱은 list[start:stop:step] 형식이며, start부터 step만큼 이동하되 stop 직전에서 끝난다.
이 원리를 쓰는 이유: -2와 -4는 각각 시작 위치와 종료 경계를 나타내고, -1은 역방향 이동을 뜻하기 때문이다.

계산 과정

1. 길이가 6인 리스트에서 인덱스 -2는 양의 인덱스 4이므로 값은 5이다.
2. 인덱스 -4는 양의 인덱스 2이며 값은 3이지만, stop 위치이므로 포함하지 않는다.
3. 인덱스 4부터 -1씩 이동하여 stop 인덱스 2 직전까지 선택하면 인덱스 4와 3의 값인 5, 4가 남는다.
따라서 출력 결과는 [5, 4]이다.

NoteBase

선지 판단

① X — [4, 3]은 시작 원소를 4로 잘못 잡은 결과이다. 시작 인덱스 -2의 값은 5이므로 올바른 결과는 [5, 4]이다.
② X — [4, 3, 2]는 시작 위치와 역방향 범위를 잘못 해석한 결과이다. -2 위치의 값 5에서 시작하고 -4 위치의 값 3은 제외하므로 올바른 결과는 [5, 4]이다.
③ O — 인덱스 -2의 값 5에서 시작해 역방향으로 이동하고, stop인 -4 위치의 값 3 직전에서 끝나므로 [5, 4]가 출력된다.
④ X — [5, 4, 3]은 stop 위치의 값 3까지 포함한 결과이다. 슬라이싱은 stop을 포함하지 않으므로 올바른 결과는 [5, 4]이다.

오답 포인트

핵심 함정은 음수 인덱스를 원솟값과 혼동하거나 stop 위치까지 포함하는 것이다. 특히 ④는 역방향 슬라이싱에서도 끝 경계가 제외된다는 점을 놓치면 선택하게 된다.

다시 나오면 이렇게 판단

음수 인덱스를 실제 인덱스로 바꾼 뒤, start는 포함하고 stop은 제외하며 step의 부호에 따라 이동 방향을 정한다.

암기 포인트

슬라이싱은 시작 포함, 끝 제외이며 step이 음수이면 역방향으로 진행한다.

NoteBase

20. 운영체제의 발달과정 순서를 나열한 것으로 가장 옳은 것은?

① 일괄처리 시스템 → 다중모드 시스템 → 시분할 시스템 → 분산처리 시스템
② 일괄처리 시스템 → 시분할 시스템 → 다중모드 시스템 → 분산처리 시스템
③ 시분할 시스템 → 일괄처리 시스템 → 분산처리 시스템 → 다중모드 시스템
④ 시분할 시스템 → 분산처리 시스템 → 일괄처리 시스템 → 다중모드 시스템

정답 2번

출제 포인트

운영체제가 작업을 처리하는 방식이 일괄처리에서 대화형 처리와 복합 처리로 발전하고, 이후 네트워크 기반 분산처리로 이어지는 순서를 묻는 문제이다.

올바른 순서

1. 일괄처리 시스템 — 여러 작업을 모아 한꺼번에 순차적으로 처리하는 초기 방식이다.
2. 시분할 시스템 — CPU 시간을 짧게 나누어 여러 사용자가 대화식으로 시스템을 이용하게 한 방식이다.
3. 다중모드 시스템 — 일괄처리, 시분할, 실시간 처리 등 여러 처리 방식을 한 시스템에서 지원하는 단계이다.
4. 분산처리 시스템 — 네트워크로 연결된 여러 컴퓨터가 작업과 자원을 나누어 처리하는 방식이다.

이 순서인 이유: 먼저 단일 시스템의 처리 방식이 일괄처리에서 시분할과 다중모드로 발전한 뒤, 통신망을 바탕으로 여러 컴퓨터가 협력하는 분산처리 단계로 확대되었기 때문이다.

NoteBase

선지별 해설

① X — 시분할 시스템과 다중모드 시스템의 선후관계가 바뀌었다. 다중모드 시스템은 시분할을 포함한 여러 처리 방식을 함께 지원하는 발전 단계이므로 시분할 시스템보다 뒤에 온다. 올바르게는 '일괄처리 시스템 → 시분할 시스템 → 다중모드 시스템 → 분산처리 시스템'이다.

② O — 일괄처리에서 시분할로 대화성이 강화되고, 여러 처리 방식을 지원하는 다중모드를 거쳐 네트워크 기반의 분산처리로 발전하는 순서가 정확하다.

③ X — '시분할 시스템 → 일괄처리 시스템'과 '분산처리 시스템 → 다중모드 시스템'의 순서가 모두 거꾸로이다. 일괄처리가 시분할보다 먼저이고, 다중모드가 분산처리보다 먼저이다. 올바르게는 '일괄처리 시스템 → 시분할 시스템 → 다중모드 시스템 → 분산처리 시스템'이다.

④ X — 시분할 시스템을 시작 단계에 놓고, 가장 뒤에 오는 분산처리 시스템을 일괄처리 시스템보다 앞에 놓은 것이 잘못이다. 올바르게는 '일괄처리 시스템 → 시분할 시스템 → 다중모드 시스템 → 분산처리 시스템'이다.

오답 포인트

①은 '다중'이라는 말 때문에 다중모드 시스템을 시분할보다 앞선 단계로 착각하게 만든다. 그러나 시분할은 CPU 시간을 여러 사용자에게 나누는 처리 방식이고, 다중모드는 시분할을 비롯한 여러 처리 방식을 함께 지원하는 더 뒤의 단계이다. 또한 분산처리는 여러 컴퓨터의 네트워크 연결이 전제되므로 네 단계 중 마지막이다.

다시 나오면 이렇게 판단

처리 방식의 범위가 '모아 처리 → 시간을 나누어 대화식 처리 → 여러 처리 방식의 통합 → 여러 컴퓨터에 분산'으로 확대되는지를 확인하면 된다.

암기 포인트

일괄 → 시분할 → 다중모드 → 분산: 모아서, 시간을 나눠서, 방식을 합쳐서, 컴퓨터를 나눠서 처리한다.

NoteBase

21. 기억 장치 배치 전략 방법인 최초 적합 전략(first-fit), 최적 적합 전략(best-fit), 최악 적합 전략(worst-fit)으로 각각 5K의 프로그램을 할당할 때, 할당영역으로 가장 옳은 것은?

할당영역	운영체제
1	4K 공백
2	15K 공백
3	10K 공백
4	5K 공백
5	20K 공백

① 할당영역: 2, 3, 2
② 할당영역: 2, 3, 5
③ 할당영역: 2, 4, 5
④ 할당영역: 2, 4, 1

정답 3번

출제 포인트

이 문제는 크기가 5*K*인 프로그램을 최초 적합, 최적 적합, 최악 적합 전략으로 각각 독립적으로 배치할 때 선택되는 공백 영역을 판단하는 문제이다.

조건 정리

공백 영역의 크기는 1번 4*K*, 2번 15*K*, 3번 10*K*, 4번 5*K*, 5번 20*K*이다.
할당할 프로그램의 크기는 5*K*이다.
크기가 5*K*보다 작은 1번 영역에는 프로그램을 할당할 수 없다.

사용할 원리

최초 적합은 앞에서부터 검색하여 처음 발견한 충분한 공백을 선택한다. 최적 적합은 프로그램을 수용할 수 있는 공백 중 가장 작은 공백을 선택한다. 최악 적합은 프로그램을 수용할 수 있는 공백 중 가장 큰 공백을 선택한다.

계산 과정

1. 최초 적합: 1번 4*K*는 부족하고, 그다음 2번 15*K*가 처음으로 충분하므로 2번 영역을 선택한다.
2. 최적 적합: 수용 가능한 영역은 2번 15*K*, 3번 10*K*, 4번 5*K*, 5번 20*K*이며, 이 중 가장 작은 4번 영역을 선택한다.
3. 최악 적합: 수용 가능한 영역 중 가장 큰 5번 20*K* 영역을 선택한다.

따라서 배치 결과는 최초 적합 2번, 최적 적합 4번, 최악 적합 5번이다.

NoteBase

선지 판단

① X — ‘2, 3, 2’는 최적 적합과 최악 적합의 영역이 잘못되었다. 올바르게는 최초 적합 2번, 최적 적합 4번, 최악 적합 5번이다.
② X — ‘2, 3, 5’에서 최적 적합을 3번으로 고른 부분이 잘못되었다. 10*K*인 3번보다 5*K*인 4번이 프로그램 크기에 정확히 맞으므로 올바르게는 ‘2, 4, 5’이다.
③ O — 최초 적합은 2번, 최적 적합은 4번, 최악 적합은 5번을 선택하므로 ‘2, 4, 5’가 맞다.
④ X — ‘2, 4, 1’에서 최악 적합을 1번으로 고른 부분이 잘못되었다. 1번은 4*K*여서 5*K* 프로그램을 수용할 수 없으며, 올바르게는 가장 큰 공백인 5번을 선택한 ‘2, 4, 5’이다.

오답 포인트

최적 적합을 ‘검색 중 적당한 공백’으로 생각하면 3번을 고르기 쉽다. 최적 적합은 검색 순서가 아니라 수용 가능한 공백 중 남는 공간이 가장 작은 영역을 선택한다. 또한 최악 적합은 가장 작은 공백이 아니라 가장 큰 공백을 선택한다.

다시 나오면 이렇게 판단

먼저 프로그램이 들어갈 수 없는 공백을 제외한다. 그다음 최초 적합은 앞에서부터 첫 번째, 최적 적합은 가능한 것 중 최소, 최악 적합은 가능한 것 중 최대라는 기준으로 판단한다.

암기 포인트

최초는 첫 공백, 최적은 가장 작은 충분한 공백, 최악은 가장 큰 공백이다.

NoteBase

22. 데이터를 송신 측에서 수신 측으로 전송할 때, 한 링크에서 다른 링크로 데이터를 교환하는 방법 중에 회선 교환 방식의 설명으로 가장 옳지 않은 것은?

① 경로가 설정되면 교환 노드에서 처리 지연이 거의 없다.
② 처리속도가 다른 통신 기기 간에도 정보를 전송할 수 있다.
③ 음성이나 동영상 등 실시간 전송이 요구되는 미디어 전송에 적합하다.
④ 경로가 설정되면 사용자에게는 고정적인 전송률로 정보를 전송할 수 있다.

정답 2번

출제 포인트

회선 교환 방식에서 전용 경로를 설정한 뒤 데이터를 전송할 때 나타나는 특징을 묻는 문제이다.

핵심 개념

회선 교환은 통신을 시작하기 전에 송신 측과 수신 측 사이에 물리적·논리적 전용 경로를 설정하고, 통신이 끝날 때까지 그 경로의 자원을 독점하는 방식이다. 경로가 설정된 뒤에는 중간 노드의 저장·전달 과정이 거의 없어 지연과 전송률이 일정하므로 실시간 연속 전송에 적합하다. 반면 서로 다른 전송 속도 사이의 속도 변환은 데이터를 저장한 뒤 다음 링크로 전달하는 패킷 교환 방식의 특징이다.

선지별 해설

① O — 경로 설정이 끝나면 중간 교환 노드는 데이터를 저장한 뒤 재전송하지 않고 설정된 회선을 따라 전달하므로 교환 노드에서의 처리 지연이 거의 없다.
② X — 틀린 표현은 '처리속도가 다른 통신 기기 간에도 정보를 전송할 수 있다'이다. 회선 교환은 설정된 전용 회선에서 고정된 전송률을 사용하므로 서로 다른 속도 사이의 변환에 적합하지 않다. 올바르게는 '패킷 교환은 중간 노드의 저장·전달 기능을 이용하므로 처리속도가 다른 통신 기기 간에도 정보를 전송할 수 있다'이다.
③ O — 경로가 설정된 뒤에는 전송 지연이 작고 일정하며 데이터가 연속적으로 전달되므로 음성이나 동영상처럼 실시간성이 필요한 미디어 전송에 적합하다.
④ O — 회선 설정 시 일정한 대역폭이 통신 종료 때까지 전용으로 할당되므로 사용자는 고정적인 전송률로 정보를 전송할 수 있다.

NoteBase

오답 포인트

②의 '처리속도가 다른 통신 기기 간'이라는 표현이 함정이다. 전용 경로를 확보하는 회선 교환과, 저장·전달을 통해 링크별 속도 차이를 흡수할 수 있는 패킷 교환의 특징을 뒤바꾼 선지이다.

다시 나오면 이렇게 판단

'경로 사전 설정·자원 독점·고정 전송률·작고 일정한 지연·실시간 전송'은 회선 교환으로 판단한다.
'저장·전달·속도 변환·회선의 통계적 공유'는 패킷 교환으로 판단한다.

암기 포인트

회선 교환은 전용 회선과 고정 속도, 패킷 교환은 저장·전달과 속도 차이 수용이다.

NoteBase

23. IPv4 주소 체계에서 B 클래스에 대한 설명으로 가장 옳지 않은 것은?
① 가장 왼쪽 2비트가 10이다.
② 중형 기관 및 기업에서 사용된다.
③ 네트워크 주소는 128.0.0.0 ~ 191.255.0.0까지 표현된다.
④ 사설주소의 호스트 사용가능 범위는 172.15.0.1 ~ 172.30.255.254이다.

정답 4번

출제 포인트

IPv4의 클래스 기반 주소 체계에서 B 클래스의 식별 비트, 용도, 주소 범위와 사설 주소 대역을 구별하는 문제이다.

핵심 개념

B 클래스는 첫 번째 옥텟이 128~191이고, 가장 왼쪽 두 비트가 10인 클래스이다. 기본 네트워크 부분은 앞 16비트이며 중형 규모의 네트워크에 사용된다.
B 클래스에 포함되는 사설 주소 블록은 172.16.0.0~172.31.255.255이다. 전체 블록의 양 끝을 제외해 호스트 범위로 표현하면 172.16.0.1~172.31.255.254이다.

선지별 해설

① O — B 클래스는 가장 왼쪽 두 비트가 10으로 시작한다.

② O — B 클래스는 A 클래스보다 작고 C 클래스보다 큰 중형 규모의 기관이나 기업 네트워크를 대상으로 한 주소 체계이다.

③ O — B 클래스의 첫 번째 옥텟은 128~191이고 기본 네트워크 부분은 앞 16비트이므로, 클래스 기준 네트워크 주소는 128.0.0.0~191.255.0.0으로 표현된다.

④ X — 틀린 부분은 사설주소의 호스트 사용가능 범위를 '172.15.0.1~172.30.255.254'라고 한 것이다. B 클래스 영역에 속하는 사설 주소 블록은 172.16.0.0~172.31.255.255이며, 선지와 같은 방식으로 호스트 범위를 쓰면 172.16.0.1~172.31.255.254이다.

NoteBase

오답 포인트

④는 범위의 시작과 끝을 모두 한 단계씩 낮춰 제시했다. 172로 시작한다고 모두 사설 주소인 것은 아니며, 두 번째 옥텟이 16부터 31까지인 범위만 해당한다.

다시 나오면 이렇게 판단

B 클래스는 '앞 2비트 10, 첫 옥텟 128~191, 기본 /16'으로 확인한 뒤, 사설 주소는 별도로 '172.16~172.31'인지 판단한다.

암기 포인트

B 클래스는 10·128~191·기본 /16, 사설 주소는 172.16~172.31이다.

NoteBase

24. 약어의 표현으로 가장 옳지 않은 것은?
① NFC: Near Field Communication
② NFT: Non-Funded Token
③ OTP: One Time Password
④ OTT: Over The Top

정답 2번

출제 포인트

정보통신 분야에서 자주 쓰이는 약어와 영문 원어의 정확한 대응을 묻는 문제이다.

핵심 개념

NFC: 가까운 거리에서 기기 간 무선 통신을 가능하게 하는 Near Field Communication이다.
NFT: 블록체인 등에서 고유한 디지털 자산을 나타내는 대체 불가능 토큰인 Non-Fungible Token이다.
OTP: 한 번만 사용할 수 있는 인증 비밀번호인 One Time Password이다.
OTT: 인터넷을 통해 영상 등 미디어 콘텐츠를 제공하는 서비스인 Over The Top이다.

선지별 해설

① O — NFC는 Near Field Communication의 약어이므로 옳다.
② X — 틀린 표현은 'Non-Funded Token'이다. NFT의 올바른 원어는 'Non-Fungible Token'이며, 'Fungible' 앞의 'Non-'이 대체 불가능하다는 뜻을 만든다.
③ O — OTP는 One Time Password의 약어로, 일회용 비밀번호를 뜻하므로 옳다.
④ O — OTT는 Over The Top의 약어로, 인터넷을 통해 미디어 콘텐츠를 제공하는 서비스를 뜻하므로 옳다.

오답 포인트

②의 'Funded'가 자연스러운 영어 단어처럼 보여 'Fungible'과 혼동하기 쉽다. NFT에서 핵심 글자 F는 Funded가 아니라 Fungible이며, 문제의 부정형 표현인 '옳지 않은 것'도 함께 확인해야 한다.

다시 나오면 이렇게 판단

약어를 볼 때 각 글자를 원어와 순서대로 대응한다. 특히 NFT는 N=Non, F=Fungible, T=Token으로 나누어 판단한다.

암기 포인트

NFT의 F는 Funded가 아니라 Fungible이다.

NoteBase

25. 관계 데이터모델의 용어에 대한 설명으로 가장 옳은 것은?
① 도메인: 하나의 속성이 가질 수 있는 값들의 집합
② 차수: 하나의 릴레이션에서 튜플의 전체 개수
③ 카디널리티: 하나의 릴레이션에서 속성의 전체 개수
④ 튜플: 하나의 개체에 관한 데이터를 2차원 테이블의 구조로 저장한 것

정답 1번

출제 포인트

관계 데이터 모델에서 도메인, 차수, 카디널리티, 튜플의 정의를 정확히 구별하는 문제이다.

핵심 개념

도메인은 하나의 속성이 가질 수 있는 원자값들의 집합이다. 차수는 릴레이션을 구성하는 속성의 수이고, 카디널리티는 릴레이션에 들어 있는 튜플의 수이다. 튜플은 릴레이션의 한 행을 이루는 하나의 레코드이다.
핵심 대비: 열의 수는 차수, 행의 수는 카디널리티, 한 행은 튜플이다.

선지별 해설

① O — 도메인은 하나의 속성이 가질 수 있는 값들의 집합이므로 옳다. 예를 들어 성별 속성의 허용값 집합이나 점수 속성의 허용 범위가 도메인에 해당한다.
② X — 틀린 표현은 '튜플의 전체 개수'이다. 차수는 하나의 릴레이션에 있는 속성의 전체 개수, 즉 열의 수이다. '튜플의 전체 개수'를 '속성의 전체 개수'로 고쳐야 한다.
③ X — 틀린 표현은 '속성의 전체 개수'이다. 카디널리티는 하나의 릴레이션에 있는 튜플의 전체 개수, 즉 행의 수이다. '속성의 전체 개수'를 '튜플의 전체 개수'로 고쳐야 한다.
④ X — 틀린 표현은 '데이터를 2차원 테이블의 구조로 저장한 것'이다. 2차원 테이블 전체는 릴레이션이고, 튜플은 그 릴레이션에서 하나의 개체 인스턴스에 관한 값을 나타내는 한 행이다. 따라서 '릴레이션을 구성하는 하나의 행'으로 고쳐야 한다.

오답 포인트

②와 ③은 차수와 카디널리티의 대상을 서로 바꾸어 놓았고, ④는 릴레이션 전체와 릴레이션을 구성하는 한 행인 튜플을 혼동하게 만든다.

NoteBase

다시 나오면 이렇게 판단

속성이 가질 수 있는 값의 범위인지, 테이블의 열 개수인지, 행 개수인지, 한 행 자체인지 차례로 구분한다. 값의 집합은 도메인, 열의 수는 차수, 행의 수는 카디널리티, 한 행은 튜플이다.

암기 포인트

도메인=허용값 집합, 차수=열 수, 카디널리티=행 수, 튜플=한 행이다.

NoteBase