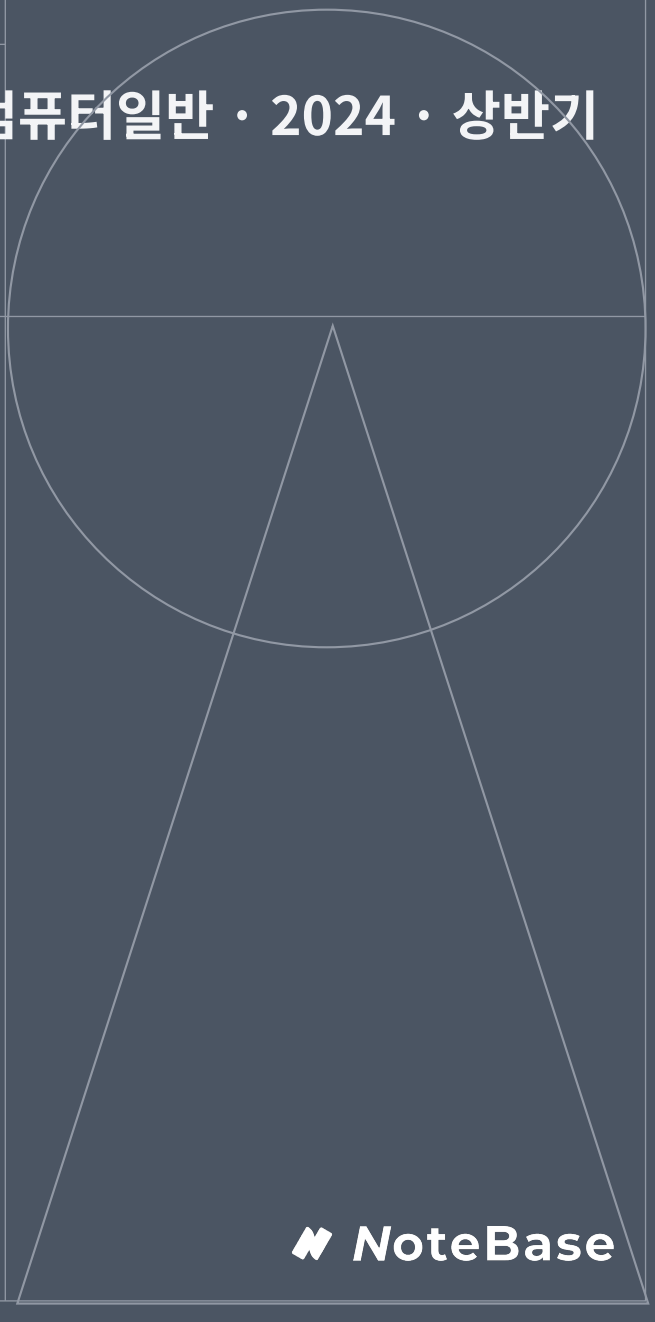
기출OX 해설집

공무원/해경/일반직 9급 · 컴퓨터일반 · 2024 · 상반기



NoteBase

기출OX 해설집

공무원/해경/일반직 9급 · 컴퓨터일반 · 2024 · 상반기

공무원/해경/일반직 9급 · 컴퓨터일반 · 2024 · 상반기

문 1. 다음 중 관계형 데이터베이스의 키(Key)에 대한 설명으로 가장 옳지 않은 것은?

① 기본키(Primary Key)는 후보키 중에서 기본적으로 사용하기 위해 선택한 키이다.

② 슈퍼키(Super Key)는 유일성과 최소성을 만족하는 속성 또는 속성들의 집합이다.

③ 대체키(Alternate Key)는 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키들을 말한다.

④ 외래키(Foreign Key)는 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합이다.

정답 2번

출제 포인트

관계형 데이터베이스의 키를 유일성과 최소성의 충족 여부 및 기본키와의 관계로 구분하는 문제이다.

핵심 개념

슈퍼키는 튜플을 유일하게 식별할 수 있는 하나 이상의 속성 집합으로 유일성은 충족하지만 최소성은 요구되지 않는다. 후보키는 슈퍼키 중 불필요한 속성을 제거할 수 없는 키로 유일성과 최소성을 모두 충족한다. 기본키는 후보키 중 대표로 선택한 키이고, 대체키는 선택되지 않은 나머지 후보키이다. 외래키는 다른 릴레이션의 기본키를 참조하여 릴레이션 사이의 관계를 나타내는 속성 집합이다.

선지별 해설

① O — 기본키는 여러 후보키 가운데 해당 릴레이션에서 대표로 사용할 키로 선택한 후보키이다.
② X — 틀린 표현은 '유일성과 최소성을 만족'한다는 부분이다. 슈퍼키는 유일성만 충족하면 되며 최소성까지 충족하는 슈퍼키가 후보키이다.
③ O — 후보키가 둘 이상이면 기본키로 선택되지 않은 나머지 후보키가 대체키가 된다.
④ O — 외래키는 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합으로, 릴레이션 사이의 참조 관계를 형성한다.

오답 포인트

②는 후보키의 조건인 '유일성+최소성'을 슈퍼키의 조건으로 바꾸어 놓았다. 슈퍼키에 불필요한 속성이 포함되어 있어도 튜플을 유일하게 식별할 수 있으면 슈퍼키가 된다.

NoteBase

다시 나오면 이렇게 판단

먼저 유일성만 요구하는지, 유일성과 최소성을 모두 요구하는지를 확인한다. 유일성만 있으면 슈퍼키이고, 여기에 최소성까지 있으면 후보키이다. 그다음 후보키 중 선택된 것은 기본키, 선택되지 않은 것은 대체키로 구분한다.

암기 포인트

슈퍼키는 유일성, 후보키는 유일성+최소성, 후보키 중 대표는 기본키, 나머지는 대체키이다.

NoteBase

문 2. 다음 중 데이터베이스에서 사용하는 뷰(View)에 대한 설명으로 가장 옳지 않은 것은?

① 독자적인 인덱스를 가질 수 있다.

② 사용자의 데이터 관리를 간단하게 한다.

③ 뷰는 삽입, 갱신, 삭제 연산에 제약이 있다.

④ 여러 사용자의 상이한 응용이나 요구를 지원한다.

정답 1번

출제 포인트

이 문제는 뷰가 실제 데이터를 별도로 저장하는 기본 테이블이 아니라, 하나 이상의 기본 테이블에서 필요한 내용을 보여 주는 논리적·가상 테이블이라는 점을 묻는다.

핵심 개념

뷰는 기본 테이블에 대한 질의 정의를 저장하여 사용자에게 필요한 데이터만 논리적 테이블처럼 제공한다. 일반적인 뷰 자체에는 데이터가 독립적으로 저장되지 않으므로 독자적인 인덱스를 두지 않으며, 필요한 인덱스는 기본 테이블에 생성한다.

선지별 해설

① X — 틀린 표현은 '독자적인 인덱스를 가질 수 있다'이다. 일반적인 뷰는 실제 데이터를 독립적으로 저장하지 않는 가상 테이블이므로 자체 인덱스를 갖지 않는다. 올바르게는 '뷰의 처리 성능을 높이기 위한 인덱스는 기본 테이블에 생성한다'이다.

② O — 뷰는 복잡한 질의를 미리 정의하고 필요한 열과 행만 보여 줄 수 있으므로 사용자가 데이터를 조회하고 관리하는 과정을 단순화한다.

③ O — 모든 뷰가 갱신 가능한 것은 아니다. 조인, 집계 함수, GROUP BY, DISTINCT 등을 포함한 뷰는 원본 행을 하나로 특정하기 어려워 삽입·갱신·삭제 연산이 제한될 수 있다.

④ O — 같은 기본 데이터를 대상으로 사용자나 업무별로 서로 다른 뷰를 정의할 수 있으므로 여러 사용자의 상이한 응용과 요구를 지원한다.

오답 포인트

③의 '제약이 있다'를 '뷰에서는 변경 연산이 전혀 불가능하다'는 뜻으로 오해하기 쉽다. 단순한 뷰는 갱신될 수 있지만 복잡한 뷰에는 제약이 있으므로 ③은 옳다. 반면 ①은 일반적인 뷰와 데이터를 실제로 저장하는 물리적 테이블의 성질을 뒤바꾼 표현이다.

NoteBase

다시 나오면 이렇게 판단

뷰 관련 선지는 먼저 '논리적·가상 테이블'이라는 정의에 대입한다. 데이터의 독립 저장이나 자체 인덱스는 일반적인 뷰의 성질이 아니고, 사용자별 표현 제공·복잡성 은폐·접근 범위 제한은 뷰의 장점이며, 변경 연산에는 제약이 있을 수 있다.

암기 포인트

뷰는 보여 주는 가상 테이블이다. 자체 데이터·자체 인덱스는 없고, 사용자별 화면 제공과 변경 제약은 있다.

NoteBase

문 3. 다음 <보기>에서 전위(Prefix) 표기식의 계산 결과로 가장 옳은 것은?

<보기>

+ − 7 4 × 3 8

① 17

② 11

③ −13

④ 27

정답 4번

출제 포인트

전위 표기식에서 연산자별 피연산자 범위를 구분하고, 안쪽 하위 식부터 계산하여 전체 값을 구하는 문제이다.

조건 정리

주어진 식: + − 74 × 38
구해야 할 값: 전위 표기식 전체의 계산 결과
주의할 조건: 전위 표기법에서는 연산자가 두 피연산자보다 앞에 놓인다. 따라서 첫 번째 +의 두 피연산자는 (−74)와 (×38)이다.

사용할 원리

전위식은 연산자 뒤에 필요한 피연산자를 차례로 묶어 하위 식을 만든 뒤 계산한다. 이 식의 구조는 (7 − 4) + (3 × 8)이다.

계산 과정

1. 왼쪽 하위 식 (−74)를 계산하면 7 − 4 = 3이다.
2. 오른쪽 하위 식 (×38)을 계산하면 3 × 8 = 24이다.
3. 맨 앞의 +를 적용하면 3 + 24 = 27이다.

선지 판단

① X — 17은 하위 식을 올바르게 묶어 계산한 값이 아니다. 올바르게는 (7 − 4) + (3 × 8) = 27이다.
② X — 11은 전위식 전체의 계산 결과가 아니다. 두 하위 식의 값 3과 24를 더하면 올바른 결과는 27이다.
③ X — -13은 연산자의 피연산자 범위나 부호를 잘못 적용한 값이다. 올바르게 묶은 식은 (7 − 4) + (3 × 8)이며 결과는 27이다.
④ O — 7 − 4 = 3, 3 × 8 = 24이므로 전체 값은 3 + 24 = 27이다.

NoteBase

오답 포인트

기호를 왼쪽부터 순서대로 계산하면 연산자별 피연산자 범위를 잘못 잡게 된다. 맨 앞의 +에 숫자 7과 4가 곧바로 대응하는 것이 아니라, 완성된 두 하위 식 (−74)와 (×38)이 대응한다.

다시 나오면 이렇게 판단

전위 표기식은 각 연산자에 필요한 피연산자 수를 확인하고, 완성되는 하위 식을 괄호로 묶은 뒤 안쪽에서 바깥쪽 순서로 계산한다.

암기 포인트

전위 표기법은 '연산자 먼저, 피연산자 나중'이며, 계산은 완성된 하위 식부터 진행한다.

NoteBase

문 4. 다음 <보기> 중 정렬 알고리즘의 수행시간을 Big-O 표기법으로 나타냈을 때 최악의 경우에 수행시간이 같은 것들로만 고른 것은?

<보기>
㉠ 선택 정렬(Selection Sort)
㉡ 삽입 정렬(Insertion Sort)
㉢ 쉘 정렬(Shell Sort)
㉣ 힙 정렬(Heap Sort)

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

정답 1번

출제 포인트

선택 정렬, 삽입 정렬, 쉘 정렬, 힙 정렬의 최악 시간 복잡도를 비교하는 문제이다.

핵심 개념

선택 정렬: 매 단계에서 최솟값을 찾아 자리를 확정하는 정렬로, 최악 시간 복잡도는 $O(n^2)$이다.
삽입 정렬: 원소를 앞쪽의 정렬된 부분에 알맞게 삽입하는 정렬로, 역순 입력 등 최악의 경우 $O(n^2)$이다.
쉘 정렬: 일정 간격으로 떨어진 원소들을 부분 정렬한 뒤 간격을 줄이는 정렬로, 일반적인 시험 기준의 최악 시간 복잡도는 $O(n^2)$이다.
힙 정렬: 힙에서 최댓값 또는 최솟값을 반복해 꺼내 정렬하며, 최악 시간 복잡도는 $O(n\log n)$이다.

선지별 해설

① O — ㉠ 선택 정렬, ㉡ 삽입 정렬, ㉢ 쉘 정렬은 모두 최악 시간 복잡도가 $O(n^2)$이므로 같은 것들로만 묶였다.
② X — ㉣ 힙 정렬을 ㉡·㉢과 같은 복잡도로 묶은 부분이 틀렸다. 힙 정렬은 최악에도 $O(n\log n)$이며, 올바른 묶음은 ㉠·㉡·㉢이다.
③ X — ㉣ 힙 정렬을 ㉠·㉢과 같은 복잡도로 묶은 부분이 틀렸다. 힙 정렬 대신 삽입 정렬 ㉡이 들어가야 하므로 올바른 묶음은 ㉠·㉡·㉢이다.
④ X — ㉣ 힙 정렬을 ㉠·㉡과 같은 복잡도로 묶은 부분이 틀렸다. 힙 정렬 대신 쉘 정렬 ㉢이 들어가야 하므로 올바른 묶음은 ㉠·㉡·㉢이다.

NoteBase

오답 포인트

힙 정렬도 원소를 반복해서 교환하므로 $O(n^2)$이라고 오인하기 쉽다. 그러나 힙의 높이가 $O(\log n)$이고 이 연산을 n번 수행하므로 최악 시간도 $O(n \log n)$이다.

다시 나오면 이렇게 판단

먼저 최악 시간 복잡도가 $O(n \log n)$인 힙 정렬을 분리한다. 남은 선택 정렬·삽입 정렬·쉘 정렬은 이 문항의 기준에서 모두 $O(n^2)$인 묶음으로 판단한다.

암기 포인트

최악 기준으로 선택·삽입·쉘은 $O(n^2)$, 힙은 $O(n \log n)$이다.

NoteBase

문 5. 다음 소프트웨어의 개발 생명주기 모형 중 나선형 모형에 대한 설명으로 가장 옳은 것은?

① 시스템 개발 초기에 아키텍처와 전체적인 구조를 확정하고, 전체를 통합해서 반복적이고 점증적으로 개발한다.

② 각 단계가 끝난 후 결과물이 명확히 나와야 하며, 중간 산출물이 명확하여 관리하기 좋다.

③ 폭포수 모델의 제어와 프로토타입 모델의 반복적 특성을 수용하고, 새로운 요소인 위험 분석을 추가하였다.

④ 잘 쓴 문서보다는 실행되는 소프트웨어에 더 가치를 두며, 계획을 따라 하는 것보다 고객의 요구사항 변화에 잘 대응하는 것을 중요하게 여긴다.

정답 3번

출제 포인트

나선형 모형의 핵심인 반복 개발과 위험 분석을 다른 생명주기 모형의 특징과 구별하는 문제이다.

핵심 개념

나선형 모형은 계획 수립, 위험 분석, 개발 및 검증, 고객 평가를 반복하면서 시스템을 점진적으로 완성하는 모형이다. 폭포수 모형의 체계적인 단계 제어와 프로토타입 모형의 반복성을 결합하고, 매 반복 주기마다 위험을 식별하고 분석한다.

통합 프로세스 모형은 아키텍처 중심으로 반복적·점증적으로 개발하고, 폭포수 모형은 단계를 순차적으로 진행하며 단계별 산출물을 중시한다. 애자일 개발은 문서나 계획보다 실행되는 소프트웨어와 변화 대응을 더 중시한다.

NoteBase

선지별 해설

① X
틀린 표현은 '나선형 모형'의 특징으로 본 부분이다. 개발 초기에 아키텍처와 전체 구조를 확립하고 반복적·점증적으로 개발하는 것은 통합 프로세스 모형의 특징이다. 나선형 모형은 반복 주기마다 위험 분석을 수행하는 것이 핵심이다.

② X
틀린 표현은 단계별 결과물과 명확한 중간 산출물을 나선형 모형의 대표 특징으로 본 부분이다. 이는 각 단계를 순차적으로 완료하고 단계별 산출물을 명확히 하는 폭포수 모형의 특징이다. 나선형 모형은 위험 분석을 중심으로 여러 주기를 반복한다.

③ O
나선형 모형은 폭포수 모형의 체계적인 제어와 프로토타입 모형의 반복적 개발을 수용하고 위험 분석을 추가한 모형이다. 위험이 큰 대규모 시스템을 반복적으로 개발하는 데 적합하므로 옳다.

④ X
틀린 표현은 실행되는 소프트웨어와 요구사항 변화 대응을 나선형 모형의 가치로 본 부분이다. 이는 애자일 선언의 핵심 가치이다. 나선형 모형의 구별 기준은 계획보다 변화 대응을 중시하는지가 아니라 반복 주기마다 위험을 분석하는지이다.

오답 포인트

①의 '반복적이고 점증적', ④의 '변화 대응'도 반복 개발과 관련되어 보여 나선형 모형과 혼동하기 쉽다. 그러나 나선형 모형을 직접 판별하는 결정적 표현은 ③의 '위험 분석'이다.

다시 나오면 이렇게 판단

선지에서 '위험 분석'을 먼저 찾으면 나선형 모형을 판별할 수 있다. '단계별 산출물·순차 진행'은 폭포수, '아키텍처 중심·반복적·점증적 개발'은 통합 프로세스, '실행되는 소프트웨어·변화 대응'은 애자일로 구별한다.

암기 포인트

나선형 모형은 반복과 위험 분석, 폭포수 모형은 순차와 산출물, 통합 프로세스는 아키텍처 중심의 반복·점증, 애자일은 실행 소프트웨어와 변화 대응이다.

NoteBase

문 6. 다음 중 소프트웨어 모듈 결합도(Coupling)에 대한 설명으로 가장 옳지 않은 것은?

① 모듈간 결합도는 약할수록 바람직하다.

② 데이터(Data) 결합은 모델들이 매개변수를 통해 데이터만 주고 받는다.

③ 레코드나 배열같은 데이터 구조는 스탬프(Stamp) 결합에 해당된다.

④ 외부(External) 결합보다는 제어(Control) 결합이 상호 의존도가 강하다.

정답 4번

출제 포인트

모듈 결합도의 의미와 데이터 결합, 스탬프 결합, 제어 결합, 외부 결합의 상대적 강도를 구별하는 문제이다.

핵심 개념

결합도는 서로 다른 모듈이 얼마나 강하게 의존하는지를 나타내며, 낮을수록 모듈의 독립성과 유지보수성이 높다.
약한 결합에서 강한 결합의 순서는 데이터 결합 → 스탬프 결합 → 제어 결합 → 외부 결합 → 공통 결합 → 내용 결합이다.
데이터 결합은 필요한 단순 데이터만 매개변수로 전달하고, 스탬프 결합은 레코드나 배열 같은 자료구조 전체를 전달한다. 제어 결합은 다른 모듈의 처리 흐름을 정하는 제어 정보를 전달하며, 외부 결합은 장치 인터페이스·통신 프로토콜·외부 파일 형식 같은 외부 환경을 여러 모듈이 함께 의존하는 형태이다.

선지별 해설

① O — 모듈 간 결합도는 낮을수록 각 모듈을 독립적으로 이해·수정하기 쉬우므로 바람직하다.
② O — 데이터 결합은 모듈들이 매개변수를 통해 처리에 필요한 데이터 항목만 주고받는 결합이다.
③ O — 레코드나 배열처럼 여러 항목이 묶인 데이터 구조 전체를 매개변수로 전달하면 스탬프 결합에 해당한다.
④ X — 틀린 표현은 '외부 결합보다는 제어 결합이 상호 의존도가 강하다'이다. 실제로는 제어 결합보다 외부 결합의 상호 의존도가 더 강하다.

오답 포인트

④는 제어 결합과 외부 결합의 강도 순서를 뒤집었다. 제어 정보를 넘겨 상대 모듈의 내부 흐름에 관여하는 제어 결합도 강한 편이지만, 외부 환경을 공동으로 의존하는 외부 결합이 그보다 더 강하다.

다시 나오면 이렇게 판단

먼저 결합도는 약할수록 좋다는 원칙을 확인하고, 전달 대상이 단순 데이터이면 데이터 결합, 자료구조 전체이면 스탬프 결합, 처리 방향을 정하는 값이면 제어 결합으로 판단한다. 강도 비교에서는 데이터 → 스탬프 → 제어 → 외부 → 공통 → 내용의 순서로 대조한다.

NoteBase

암기 포인트

결합도는 데이터 < 스탬프 < 제어 < 외부 < 공통 < 내용이며, 오른쪽으로 갈수록 강하고 바람직하지 않다.

NoteBase

문 7. 다음 중 TCP/IP 프로토콜 중 물리적인 주소(Mac Address)에 해당하는 논리적인 주소(IP Address)를 찾아와 주는 프로토콜로 가장 옳은 것은?

① SNMP(Simple Network Management Protocol)

② RARP(Reverse Address Resolution Protocol)

③ ICMP(Internet Control Message Protocol)

④ ARP(Address Resolution Protocol)

정답 2번

출제 포인트

물리 주소인 MAC 주소를 바탕으로 논리 주소인 IP 주소를 알아내는 RARP와, 반대 방향으로 주소를 변환하는 ARP를 구별하는 문제이다.

핵심 개념

RARP는 호스트가 자신의 MAC 주소에 대응하는 IP 주소를 서버에 요청하여 알아내는 프로토콜이다.
ARP는 알고 있는 IP 주소에 대응하는 MAC 주소를 알아내는 프로토콜이다.
즉, RARP는 MAC → IP이고 ARP는 IP → MAC이다.

선지별 해설

① X — SNMP는 네트워크 장비의 상태를 감시하고 설정 정보를 관리하는 네트워크 관리 프로토콜이므로 MAC 주소로 IP 주소를 찾는 프로토콜이 아니다. 틀린 선택은 SNMP이며, 올바른 프로토콜은 RARP이다.
② O — RARP는 Reverse Address Resolution Protocol로, 주어진 MAC 주소에 대응하는 IP 주소를 알아내는 역주소 변환 프로토콜이다.
③ X — ICMP는 IP 통신 중 발생한 오류를 알리고 진단·제어 메시지를 전달하는 프로토콜이므로 주소 변환을 수행하지 않는다. 틀린 선택은 ICMP이며, 올바른 프로토콜은 RARP이다.
④ X — ARP는 IP 주소를 이용하여 대응하는 MAC 주소를 찾으므로 문제에서 제시한 변환 방향과 반대이다. 틀린 선택은 ARP이며, MAC 주소로 IP 주소를 찾는 프로토콜은 RARP이다.

오답 포인트

④ ARP도 IP 주소와 MAC 주소를 대응시키므로 혼동하기 쉽지만, 변환 방향이 반대이다. 문제의 출발점이 MAC 주소이면 RARP, IP 주소이면 ARP이다.

다시 나오면 이렇게 판단

문장에서 이미 알고 있는 주소와 찾아야 하는 주소를 구분한다. MAC을 알고 IP를 찾으면 RARP이고, IP를 알고 MAC을 찾으면 ARP이다.

NoteBase

암기 포인트

ARP는 IP → MAC, Reverse가 붙은 RARP는 MAC → IP이다.

NoteBase

문 8. 다음 중 CPU의 명령어 처리 성능을 높이기 위한 파이프라이닝(Pipelining)에 관한 설명으로 가장 옳지 않은 것은?

① 제어 해저드(Control Hazard)는 산술논리 연산 명령으로 발생한다.

② 데이터 해저드(Data Hazard)는 액세스 데이터의 의존성에 의해 발생한다.

③ 구조적 해저드(Structural Hazard)는 내부 자원에 대한 충돌에 의해 발생한다.

④ 성능 개선 정도는 해저드에 따라 달라질 수 있다.

정답 1번

출제 포인트

파이프라이닝에서 발생하는 제어 해저드, 데이터 해저드, 구조적 해저드의 원인과 해저드가 성능에 미치는 영향을 구별하는 문제이다.

핵심 개념

파이프라이닝은 여러 명령어의 처리 단계를 겹쳐 실행하여 처리량을 높이는 방식이다.
제어 해저드는 분기·점프 등으로 다음에 실행할 명령어가 확정되지 않을 때 발생한다.
데이터 해저드는 앞 명령어의 결과를 뒤 명령어가 필요로 하는 등 명령어 사이에 데이터 의존성이 있을 때 발생한다.
구조적 해저드는 같은 시점에 여러 명령어가 동일한 하드웨어 자원을 요구할 때 발생한다.

선지별 해설

① X — 틀린 표현은 '제어 해저드는 산술논리 연산 명령으로 발생한다'이다. 제어 해저드는 주로 분기·점프와 같은 제어 흐름 변경 명령 때문에 다음 명령어의 위치가 확정되지 않아 발생한다. 산술논리 연산의 결과를 뒤 명령어가 이용한다면 이는 제어 해저드가 아니라 데이터 해저드와 관련된다.
② O — 데이터 해저드는 명령어 사이의 데이터 의존성 때문에 발생한다. 앞 명령어가 기록할 값을 뒤 명령어가 읽어야 하는 경우처럼 필요한 데이터가 아직 준비되지 않으면 파이프라인이 지연될 수 있다.
③ O — 구조적 해저드는 두 개 이상의 명령어가 같은 클록에 메모리나 연산 장치 등 동일한 내부 자원을 동시에 사용하려 할 때 발생하는 자원 충돌이다.
④ O — 해저드가 발생하면 정지, 버블 삽입, 파이프라인 비우기 등이 필요할 수 있으므로 실제 성능 개선 정도는 해저드의 종류와 발생 빈도 및 처리 방식에 따라 달라진다.

오답 포인트

①은 '산술논리 연산'이라는 명령어 종류와 '제어'라는 이름을 잘못 연결한 선지이다. 해저드는 명령어 이름보다 충돌 원인으로 구분하며, 다음 명령어 주소의 불확실성은 제어 해저드, 값의 선후 의존성은 데이터 해저드이다.

NoteBase

다시 나오면 이렇게 판단

원인이 다음 실행 주소의 불확실성이면 제어 해저드, 명령어 사이의 값 의존성이면 데이터 해저드, 하드웨어 자원의 동시 사용 충돌이면 구조적 해저드로 판단한다.

암기 포인트

분기·점프는 제어, 값의 의존은 데이터, 자원의 충돌은 구조적 해저드이다.

NoteBase

문 9. 다음 중 10진수 −17을 2의 보수(2's Complement) 형식의 8비트 2진수로 나타낸 것으로 옳은 것은?

① 10010100

② 10010011

③ 11101110

④ 11101111

정답 4번

출제 포인트

이 문제는 음의 정수를 8비트 2의 보수로 표현하는 과정을 묻는다.

조건 정리

주어진 값: 10진수 -17
구해야 할 값: -17의 8비트 2의 보수 표현
주의할 조건: 먼저 +17을 8비트로 맞춘 뒤 모든 비트를 반전하고 1을 더한다.

사용할 공식

음수의 2의 보수 표현은 양수의 이진 표현에서 모든 비트를 반전한 뒤 1을 더해 구한다. 같은 결과를 $2^8 - 17 = 239$로 계산한 뒤 239를 8비트 이진수로 바꾸어도 얻을 수 있다.

이 원리를 쓰는 이유: 문제에서 -17을 8비트 2의 보수 형식으로 나타내라고 했기 때문이다.

계산 과정

1. +17의 8비트 이진 표현은 00010001이다.
2. 모든 비트를 반전하면 11101110이다.
3. 여기에 1을 더하면 11101111이다.

따라서 -17의 8비트 2의 보수 표현은 11101111이다.

NoteBase

선지 판단

① X — 10010100은 최상위 비트가 1인 8비트 2의 보수로 해석하면 -108이다. -17의 표현이 아니며, 올바른 값은 11101111이다.
② X — 10010011은 8비트 2의 보수로 해석하면 -109이다. -17의 표현이 아니며, 올바른 값은 11101111이다.
③ X — 11101110은 +17의 비트를 반전하기만 한 값으로, 1을 더하지 않은 1의 보수 표현이다. 2의 보수에서는 여기에 1을 더하여 11101111로 고쳐야 한다.
④ O — 00010001의 비트를 반전한 11101110에 1을 더하면 11101111이 되므로 -17의 8비트 2의 보수 표현과 일치한다.

오답 포인트

③의 11101110은 비트 반전까지는 했지만 마지막에 1을 더하지 않은 값이다. 1의 보수와 2의 보수를 혼동하면 선택하기 쉬운 함정이다.

다시 나오면 이렇게 판단

음수를 2의 보수로 바꾸는 문제는 비트 수를 맞춘 양수 표현 → 모든 비트 반전 → 1 더하기의 순서로 판단한다.

암기 포인트

2의 보수는 비트 반전 후 1을 더한다.

NoteBase

문 10. 컴퓨터 시스템의 기억장치는 여러 계층으로 구성되어 성능을 높인다. 다음 중 접근 속도가 빠른 것부터 느린 순서대로 나열된 것으로 가장 옳은 것은?

① 캐시 메모리, CPU 내부 레지스터, 주기억장치, 보조기억장치

② CPU 내부 레지스터, 캐시 메모리, 주기억장치, 보조기억장치

③ CPU 내부 레지스터, 주기억장치, 캐시 메모리, 보조기억장치

④ 캐시 메모리, CPU 내부 레지스터, 보조기억장치, 주기억장치

정답 2번

출제 포인트

기억장치 계층에서 CPU와의 거리, 용도, 구현 방식에 따른 접근 속도의 순서를 묻는 문제이다.

올바른 순서

1. CPU 내부 레지스터 — CPU가 연산 중인 데이터와 명령을 직접 보관하므로 가장 빠르다.
2. 캐시 메모리 — 자주 사용하는 데이터와 명령을 임시로 저장하여 주기억장치 접근을 줄인다.
3. 주기억장치 — 실행 중인 프로그램과 데이터를 저장하지만 레지스터와 캐시보다 접근 속도가 느리다.
4. 보조기억장치 — 데이터를 장기간 보관하며 네 계층 중 접근 속도가 가장 느리다.

왜 이 순서: CPU가 즉시 사용하는 레지스터가 가장 빠르고, CPU와 주기억장치의 속도 차이를 보완하는 캐시가 그다음이며, 주기억장치와 보조기억장치가 차례로 이어진다.

NoteBase

선지별 해설

① X
틀린 부분: 캐시 메모리를 CPU 내부 레지스터보다 먼저 둔 부분이 틀리다.
올바르게는: CPU 내부 레지스터 → 캐시 메모리 → 주기억장치 → 보조기억장치 순이다.

② O
CPU 내부 레지스터 → 캐시 메모리 → 주기억장치 → 보조기억장치로, 접근 속도가 빠른 것부터 느린 순서가 정확하다.

③ X
틀린 부분: 주기억장치를 캐시 메모리보다 먼저 둔 부분이 틀리다.
올바르게는: 캐시 메모리가 주기억장치보다 빠르므로 CPU 내부 레지스터 → 캐시 메모리 → 주기억장치 → 보조기억장치 순이다.

④ X
틀린 부분: 캐시 메모리를 CPU 내부 레지스터보다 먼저 두고, 보조기억장치를 주기억장치보다 먼저 둔 부분이 틀리다.
올바르게는: CPU 내부 레지스터가 캐시 메모리보다 빠르고 주기억장치가 보조기억장치보다 빠르므로 CPU 내부 레지스터 → 캐시 메모리 → 주기억장치 → 보조기억장치 순이다.

오답 포인트

CPU 가까이에 있는 캐시를 가장 빠르다고 착각하기 쉽지만, 연산에 직접 사용되는 CPU 내부 레지스터가 캐시보다 빠르다. 또한 보조기억장치는 비휘발성·대용량 저장에 유리할 뿐, 주기억장치보다 빠르지 않다.

다시 나오면 이렇게 판단

접근 속도는 CPU가 직접 사용하는 저장공간에서 장기 보관 장치로 멀어지는 순서인 레지스터 → 캐시 → 주기억장치 → 보조기억장치로 판단한다. 일반적으로 속도가 느려질수록 용량은 커지고 비트당 가격은 낮아진다.

암기 포인트

속도 순서는 레지스터 → 캐시 → 주기억 → 보조기억이다.

NoteBase

문 11. 다음 중 DMA(Direct Memory Access) 제어기에 관련된 설명으로 가장 옳지 않은 것은?

① 대용량의 데이터를 전송하기 위한 장치이다.

② 시스템 버스에 연결되어 있는 장치이다.

③ CPU가 직접 워드 단위 데이터로 메모리를 읽거나 쓰는 사이클을 생성한다.

④ 시스템 버스를 이용한 입출력 전송을 처리한다.

정답 3번

출제 포인트

DMA 제어기와 CPU 중 누가 입출력 데이터 전송과 메모리 접근 사이클을 담당하는지 구별하는 문제이다.

핵심 개념

DMA는 CPU가 데이터를 하나씩 옮기지 않아도 DMA 제어기가 입출력장치와 주기억장치 사이의 데이터를 직접 전송하는 방식이다. DMA 제어기는 시스템 버스의 제어권을 얻어 메모리 읽기·쓰기 사이클을 생성하고, 전송이 끝나면 CPU에 완료를 알린다.

선지별 해설

① O — DMA 제어기는 CPU의 반복적인 데이터 이동을 대신하므로 대용량 데이터를 고속으로 전송하는 데 적합하다.
② O — DMA 제어기는 메모리 및 입출력장치와 데이터를 주고받기 위해 시스템 버스에 연결된다.
③ X — 틀린 표현은 ‘CPU가 직접’ 메모리 읽기·쓰기 사이클을 생성한다는 부분이다. DMA 전송 중에는 DMA 제어기가 버스 제어권을 얻어 워드 단위 등의 메모리 읽기·쓰기 사이클을 생성한다.
④ O — DMA 제어기는 시스템 버스를 통해 입출력장치와 주기억장치 사이의 데이터 전송을 처리한다.

오답 포인트

③은 메모리 접근이 이루어진다는 사실 때문에 CPU가 그 사이클도 직접 생성한다고 착각하게 만든다. DMA의 핵심은 전송 중 버스 제어와 메모리 접근 사이클의 주체가 CPU가 아니라 DMA 제어기라는 점이다.

다시 나오면 이렇게 판단

선지에서 데이터 전송이나 메모리 읽기·쓰기 사이클의 주체를 확인한다. DMA 방식에서는 CPU가 전송을 시작하도록 설정한 뒤, 실제 데이터 이동과 해당 버스 사이클은 DMA 제어기가 담당한다.

암기 포인트

CPU는 DMA 전송을 지시하고, DMA 제어기는 버스를 장악해 데이터를 직접 옮긴다.

NoteBase

문 12. 다음 중 프로그래밍 언어 분류 상 인터프리터 언어로 가장 옳은 것은?

① 베이직(Basic)

② 어셈블리어(Assembly Language)

③ 코볼(Cobol)

④ 포트란(Fortran)

정답 1번

출제 포인트

프로그래밍 언어를 번역·실행하는 대표 방식에 따라 인터프리터 언어와 컴파일 언어, 어셈블리어로 구분할 수 있는지를 묻는 문제이다.

핵심 개념

인터프리터 방식은 원시 프로그램을 실행하면서 문장 단위로 해석하고 수행하는 방식이다. 전통적인 언어 분류에서 베이직은 대표적인 인터프리터 언어로 다룬다. 반면 코볼과 포트란은 대표적인 컴파일 언어이며, 어셈블리어는 어셈블러가 기계어로 번역하는 저급 언어이다.

선지별 해설

① O — 베이직(BASIC)은 전통적인 분류에서 인터프리터 방식으로 실행되는 대표 언어이므로 옳다.
② X — 어셈블리어를 인터프리터 언어로 본 부분이 틀리다. 올바르게는 어셈블리어는 기계어 명령을 기호로 표현한 저급 언어이며 어셈블러가 기계어로 번역한다.
③ X — 코볼(COBOL)을 인터프리터 언어로 본 부분이 틀리다. 올바르게는 코볼은 사무·업무 처리용으로 개발된 고급 언어이며 전통적인 분류에서는 컴파일 언어에 해당한다.
④ X — 포트란(Fortran)을 인터프리터 언어로 본 부분이 틀리다. 올바르게는 포트란은 과학·기술 계산용으로 개발된 고급 언어이며 전통적인 분류에서는 컴파일 언어에 해당한다.

오답 포인트

모든 프로그래밍 언어가 번역 과정을 거친다는 점만 보고 번역 프로그램의 종류를 혼동하기 쉽다. 특히 ②는 번역기가 필요하지만 그 번역기는 인터프리터가 아니라 어셈블러라는 점이 함정이다.

다시 나오면 이렇게 판단

실행하면서 문장 단위로 해석하는 대표 언어이면 인터프리터 언어, 프로그램 전체를 목적 프로그램으로 번역하는 대표 언어이면 컴파일 언어, 기호 명령을 기계어로 바꾸면 어셈블리어로 판단한다.

NoteBase

암기 포인트

BASIC은 인터프리터, COBOL·Fortran은 컴파일, Assembly Language는 어셈블러로 연결한다.

NoteBase

문 13. 다음 중 스레드(Thread)에 관련된 내용으로 가장 옳지 않은 것은?

① 다중 스레드는 처리 성능을 높이는 데 효과적이다.

② 한 프로세스 내 여러 스레드가 포함될 수 있다.

③ 여러 프로세스가 한 스레드를 공유할 수 있다.

④ 한 프로세스 내의 여러 스레드는 데이터를 공유할 수 있다.

정답 3번

출제 포인트

스레드가 프로세스에 속하는 실행 단위라는 점과 같은 프로세스 안의 스레드들이 자원을 공유한다는 점을 구별하는 문제이다.

핵심 개념

프로세스는 실행 중인 프로그램이 자원과 주소 공간을 소유하는 단위이고, 스레드는 그 프로세스 안에서 실행 흐름을 담당하는 단위이다. 한 프로세스에는 여러 스레드가 존재할 수 있으며, 이들은 코드·데이터·힙 등의 프로세스 자원을 공유하지만 각자 스택과 레지스터 상태를 가진다. 하나의 스레드가 여러 프로세스에 공동으로 소속되지는 않는다.

선지별 해설

① O — 다중 스레드는 작업을 나누어 동시에 진행하거나 입출력 대기 시간에 다른 작업을 수행하게 하므로 처리 성능과 응답성을 높이는 데 효과적이다.
② O — 하나의 프로세스는 하나 이상의 스레드를 포함할 수 있으며, 여러 실행 흐름을 두는 형태가 다중 스레드이다.
③ X — 틀린 표현은 '여러 프로세스가 한 스레드를 공유할 수 있다'이다. 하나의 스레드는 특정한 하나의 프로세스에 속한다. 올바르게는 '한 프로세스가 여러 스레드를 포함할 수 있다'이다. 프로세스 사이의 정보 교환은 프로세스 간 통신 등의 별도 방법을 사용한다.
④ O — 같은 프로세스에 속한 스레드들은 그 프로세스의 코드·데이터·힙과 열린 파일 등의 자원을 공유할 수 있다. 다만 각 스레드는 독립적인 스택과 레지스터 상태를 가진다.

오답 포인트

③은 '프로세스가 여러 스레드를 가진다'는 관계의 방향을 뒤집어 놓았다. 프로세스들이 데이터를 주고받거나 공유 메모리를 사용할 수 있다는 사실과 하나의 스레드를 여러 프로세스가 공유한다는 주장은 서로 다르다.

NoteBase

다시 나오면 이렇게 판단

스레드와 프로세스의 관계를 묻는 선지는 먼저 소속 방향을 확인한다. 스레드는 프로세스 안에 속하고, 같은 프로세스의 스레드끼리는 프로세스 자원을 공유하며, 각 스레드의 스택과 레지스터 상태는 독립적이다.

암기 포인트

프로세스는 자원 소유 단위, 스레드는 프로세스 안의 실행 단위이다. 한 프로세스에 여러 스레드는 가능하지만 한 스레드가 여러 프로세스에 속하지는 않는다.

NoteBase

문 14. 다음 중 OSI 구성에서 동일 계층에 해당하는 프로토콜만으로 구성된 것으로 가장 옳은 것은?

① TCP, UDP, PPP

② HTTP, FTP, Telnet

③ UDP, IP, ATM

④ SMB, TCP, RPC

정답 2번

출제 포인트

OSI 참조 모델에서 각 프로토콜이 어느 계층에서 동작하는지를 구분하는 문제이다.

핵심 개념

응용 계층은 사용자의 네트워크 응용 서비스가 이용하는 프로토콜을 다루며 HTTP, FTP, Telnet, SMB, RPC 등이 해당한다.
전송 계층은 종단 간 데이터 전달을 담당하며 TCP와 UDP가 해당한다.
네트워크 계층은 패킷의 주소 지정과 경로 선택을 담당하며 IP가 해당한다.
데이터 링크 계층은 인접한 노드 사이의 프레임 전송을 담당하며 PPP가 해당하고, ATM도 일반적으로 이 계층의 교환 기술로 분류한다.

선지별 해설

① X — TCP와 UDP는 전송 계층이지만 PPP는 데이터 링크 계층이다. 틀린 부분은 PPP를 TCP·UDP와 같은 계층에 묶은 것이며, 같은 계층으로 구성하려면 PPP 대신 SCTP와 같은 전송 계층 프로토콜이 와야 한다.
② O — HTTP, FTP, Telnet은 모두 사용자에게 웹, 파일 전송, 원격 접속 서비스를 제공하는 응용 계층 프로토콜이다.
③ X — UDP는 전송 계층, IP는 네트워크 계층, ATM은 일반적으로 데이터 링크 계층의 교환 기술이므로 서로 다른 계층이 섞여 있다. 같은 계층으로 구성하려면 UDP·IP 대신 PPP나 Ethernet과 같은 데이터 링크 계층 프로토콜·기술이 와야 한다.
④ X — SMB와 RPC는 응용 계층에서 사용되지만 TCP는 전송 계층이다. 틀린 부분은 TCP를 SMB·RPC와 같은 계층에 묶은 것이며, 같은 계층으로 구성하려면 TCP 대신 HTTP와 같은 응용 계층 프로토콜이 와야 한다.

오답 포인트

①은 TCP와 UDP가 같은 전송 계층이라서 그대로 고르기 쉽지만 PPP가 데이터 링크 계층이라는 점에서 틀린다.
④도 SMB와 RPC가 응용 계층이라는 공통점에 TCP를 끼워 넣은 함정이다.

NoteBase

다시 나오면 이렇게 판단

각 선지를 서비스 프로토콜(응용), 종단 간 전달(TCP·UDP), 주소 지정·경로 선택(IP), 링크·프레임 전송(PPP)으로 나누어 하나라도 다른 계층이 섞였는지 확인한다.

암기 포인트

HTTP·FTP·Telnet은 응용, TCP·UDP는 전송, IP는 네트워크, PPP는 데이터 링크 계층이다.

NoteBase

문 15. 다음 중 패킷 전송 방식에 대한 설명으로 가장 옳지 않은 것은?

① 가상 회선 방식은 송신 노드와 수신 노드 사이에 패킷 전송 경로가 설정된다.

② 가상 회선 방식은 수신 노드에서 패킷 유실이 탐지된다.

③ 데이터그램 방식은 수신 노드로 다양한 경로를 통해 패킷이 전송된다.

④ 데이터그램 방식은 패킷 전송 순서로 수신된다.

정답 4번

출제 포인트

이 문제는 패킷 교환의 가상 회선 방식과 데이터그램 방식이 경로 설정 및 도착 순서를 어떻게 다르게 처리하는지를 묻는 문제이다.

핵심 개념

가상 회선 방식은 데이터를 보내기 전에 송신 노드와 수신 노드 사이에 논리적인 전송 경로를 설정하고, 모든 패킷을 그 경로로 보내는 연결형 방식이다.
데이터그램 방식은 각 패킷을 독립적으로 취급하여 미리 정해진 전송 경로 없이 보내는 비연결형 방식이다.
가상 회선은 동일 경로와 순서 유지가 핵심이고, 데이터그램은 패킷별 경로 선택과 도착 순서 불일치 가능성이 핵심이다.

선지별 해설

① O — 가상 회선 방식은 실제 데이터를 전송하기 전에 송신 노드와 수신 노드 사이에 논리적 경로인 가상 회선을 설정한다.
② O — 가상 회선에서는 설정된 하나의 경로를 따라 패킷이 순서대로 전달되므로, 수신 노드가 패킷 흐름의 연속성을 확인하여 유실을 탐지할 수 있다.
③ O — 데이터그램 방식은 각 패킷이 독립적으로 라우팅되므로 같은 수신 노드로 가는 패킷도 서로 다른 경로를 거칠 수 있다.
④ X — 틀린 표현은 '패킷 전송 순서로 수신된다'이다. 데이터그램 방식에서는 각 패킷이 서로 다른 경로를 거칠 수 있으므로 전송한 순서와 다른 순서로 도착할 수 있다. 올바르게는 '데이터그램 방식은 패킷의 도착 순서가 전송 순서와 달라질 수 있다'이다.

오답 포인트

④는 가상 회선 방식의 '순서대로 도착'하는 특징을 데이터그램 방식의 특징처럼 바꾸어 놓았다.
데이터그램에서도 우연히 전송 순서대로 도착할 수는 있지만, 그 순서를 보장하지 않는다는 점이 핵심이다.

NoteBase

다시 나오면 이렇게 판단

먼저 경로를 설정하고 모든 패킷이 같은 경로를 지나면 가상 회선 방식으로 판단한다. 패킷마다 독립적으로 경로가 정해져 서로 다른 길을 갈 수 있으면 데이터그램 방식이며, 이때 도착 순서는 보장되지 않는다.

암기 포인트

가상 회선은 같은 길·같은 순서, 데이터그램은 각자 길·순서 불보장이다.

NoteBase

문 16. 다음 중 IPv4와 IPv6에 관련된 내용으로 가장 옳지 않은 것은?

① IPv4는 4바이트, IPv6는 16바이트 주소로 구성된다.

② IPv4는 NAT를 이용, LAN에서 사설 주소를 이용할 수 있다.

③ IPv4와 IPv6는 동일한 LAN에서 사용이 불가능하다.

④ IPv4는 브로드캐스트를 지원하나, IPv6는 지원하지 않는다.

정답 3번

출제 포인트

IPv4와 IPv6의 주소 길이, 주소 변환, 공존 가능성, 브로드캐스트 지원 여부를 구분하는 문제이다.

핵심 개념

IPv4는 32비트 주소를 사용하는 인터넷 프로토콜이며, 사설 주소와 NAT 및 브로드캐스트를 사용할 수 있다.
IPv6는 128비트 주소를 사용하는 인터넷 프로토콜이며, 브로드캐스트 대신 멀티캐스트와 애니캐스트를 사용한다.
IPv4와 IPv6는 주소 체계가 다르지만 듀얼 스택 등의 전환 기술을 이용하면 같은 LAN에서 함께 운용할 수 있다.

선지별 해설

① O — IPv4 주소는 32비트, 즉 4바이트이고 IPv6 주소는 128비트, 즉 16바이트이므로 옳다.
② O — LAN 내부의 호스트에는 IPv4 사설 주소를 할당할 수 있으며, NAT는 사설 주소와 공인 주소를 변환해 외부 네트워크와 통신하게 하므로 옳다.
③ X — 틀린 표현은 '동일한 LAN에서 사용이 불가능하다'이다. 실제로는 듀얼 스택을 구성하면 같은 LAN의 장비에서 IPv4와 IPv6를 함께 사용할 수 있다.
④ O — IPv4는 브로드캐스트를 지원하지만 IPv6에는 브로드캐스트 주소가 없으며, 필요한 일대다 통신은 주로 멀티캐스트로 처리하므로 옳다.

오답 포인트

IPv4와 IPv6의 주소 형식이 서로 다르다는 사실을 두 프로토콜이 같은 네트워크에서 공존할 수 없다는 뜻으로 오해하기 쉽다. ③의 '사용이 불가능하다'가 함정이며, 듀얼 스택은 두 프로토콜을 동시에 운용하는 대표적인 방식이다.

다시 나오면 이렇게 판단

주소 길이는 IPv4 32비트·IPv6 128비트로 구분하고, 공존 여부는 듀얼 스택을 떠올린다. 브로드캐스트는 IPv4에만 있고 IPv6는 멀티캐스트를 사용한다는 기준으로 판단한다.

NoteBase

암기 포인트

IPv4는 32비트·브로드캐스트, IPv6는 128비트·멀티캐스트, 둘의 공존은 듀얼 스택이다.

NoteBase

문 17. 다음 중 시간 내 처리가 요구된 경우에 적용하는 시스템으로 가장 옳은 것은?

① 실시간 시스템

② 다단계 우선순위 시스템

③ 다중 프로세서 시스템

④ 시분할 시스템

정답 1번

출제 포인트

정해진 시간 안에 반드시 처리를 끝내야 하는 요구와 실시간 시스템의 특징을 연결하는 문제이다.

핵심 개념

실시간 시스템은 외부 사건이나 입력에 대해 정해진 제한 시간 안에 결과를 내도록 설계된 시스템이다. 핵심 판단 기준은 단순히 처리 속도가 빠른지가 아니라 마감 시간의 충족이 요구되는지이다.
다단계 우선순위 시스템은 작업을 여러 우선순위 단계로 나누어 처리 순서를 정하고, 다중 프로세서 시스템은 둘 이상의 처리기를 이용하며, 시분할 시스템은 CPU 시간을 짧게 나누어 여러 사용자가 대화식으로 이용하게 한다.

선지별 해설

① O — 실시간 시스템은 정해진 시간 제한 안에 처리가 완료되어야 하는 업무에 적용하므로 문제의 조건에 정확히 부합한다.
② X — 틀린 부분은 '다단계 우선순위 시스템'이다. 이는 작업을 여러 우선순위 단계로 구분하여 처리 순서를 정하는 방식이며, 올바르게는 제한 시간 안의 처리가 핵심인 '실시간 시스템'이다.
③ X — 틀린 부분은 '다중 프로세서 시스템'이다. 이는 둘 이상의 처리기가 작업을 수행하는 시스템을 뜻하며, 올바르게는 처리기의 수가 아니라 정해진 시간 안의 완료를 보장하도록 설계된 '실시간 시스템'이다.
④ X — 틀린 부분은 '시분할 시스템'이다. 이는 CPU 시간을 짧은 시간 단위로 나누어 여러 사용자가 번갈아 이용하도록 하는 시스템이며, 올바르게는 마감 시간 내 처리가 요구될 때 적용하는 '실시간 시스템'이다.

오답 포인트

'신속한 응답'이라는 인상 때문에 시분할 시스템을 고르기 쉽다. 그러나 시분할 시스템은 여러 사용자의 대화식 이용과 빠른 체감 응답이 목적이고, 실시간 시스템은 정해진 제한 시간의 충족이 핵심이다.

다시 나오면 이렇게 판단

문제에 '정해진 시간 내', '즉시 제어', '마감 시간 준수'가 제시되면 실시간 시스템을 판단한다. '여러 사용자가 CPU 시간을 나누어 사용'하면 시분할 시스템, '여러 처리기'이면 다중 프로세서 시스템으로 구분한다.

NoteBase

암기 포인트

시간 제한 준수는 실시간, 시간 조각 공유는 시분할이다.

NoteBase

문 18. 다음 중 프로세스 제어 블록(Process Control Block)에 관련된 내용으로 가장 옳은 것은?

① 프로그램 소스 컴파일 시 생성되며, 메모리 사용량이 명세된다.

② 프로세스 실행 상태와 스케줄링 정보가 기록된다.

③ 프로세스 종료 시 운영체제 관리 스토리지에 저장된다.

④ 프로세스간 문맥교환 시 해당 프로세스 사이에 공유된다.

정답 2번

출제 포인트

프로세스 제어 블록(PCB)의 생성 시점, 저장 정보, 소멸 시점과 문맥 교환에서의 역할을 묻는 문제이다.

핵심 개념

PCB는 운영체제가 각 프로세스를 관리하기 위해 만드는 자료 구조이다. 프로세스 식별자, 프로세스 상태, 프로그램 카운터와 레지스터 값, CPU 스케줄링 정보, 메모리 관리 정보, 입출력 상태 등을 담는다. 문맥 교환이 일어나면 운영체제는 실행을 멈추는 프로세스의 문맥을 그 프로세스의 PCB에 저장하고, 실행할 프로세스의 PCB에서 문맥을 복원한다.

선지별 해설

① X — 틀린 표현은 '프로그램 소스 컴파일 시 생성'이다. PCB는 컴파일러가 만드는 정보가 아니라, 운영체제가 프로세스를 생성할 때 해당 프로세스를 관리하기 위해 생성한다. 메모리 관리 정보가 포함될 수 있지만 단순히 메모리 사용량만 명세하는 자료 구조는 아니다.
② O — PCB에는 준비·실행·대기와 같은 프로세스 실행 상태와 우선순위, 스케줄링 큐 관련 정보 등 CPU 스케줄링에 필요한 정보가 기록된다.
③ X — 틀린 표현은 '프로세스 종료 시 운영체제 관리 스토리지에 저장'이다. 프로세스가 종료되면 운영체제는 해당 PCB와 프로세스에 할당된 자원을 회수한다.
④ X — 틀린 표현은 '해당 프로세스 사이에 공유'이다. 문맥 교환 시 각 프로세스의 실행 문맥은 그 프로세스에 대응하는 PCB에 따로 저장되며, 두 프로세스가 하나의 PCB를 공유하는 것이 아니다.

오답 포인트

①은 프로그램 번역 과정인 컴파일과 운영체제의 프로세스 생성 과정을 섞었고, ④는 문맥 교환에 두 프로세스가 관여한다는 사실을 PCB 공유로 바꾸었다. PCB는 프로세스별 운영체제 관리 정보라는 점이 핵심이다.

다시 나오면 이렇게 판단

PCB 관련 선지는 '운영체제가 프로세스별로 생성·관리하는가', '실행 문맥과 스케줄링 정보를 담는가', '종료 시 회수되는가'를 기준으로 판단한다.

NoteBase

암기 포인트

PCB는 프로세스별 관리 장부이다. 생성 때 만들고, 문맥 교환 때 저장·복원하며, 종료 때 회수한다.

NoteBase

문 19. 다음 <보기>의 카르노맵으로 나타나는 논리식의 결과 F로 가장 옳지 않은 것은?
(단, ~은 NOT, ⊕은 XOR, ⊙는 XNOR 연산을 의미한다.)

< 보 기 >

A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

① $F = A \oplus B \oplus C$

② $F = A \odot B \odot C$

③ $F = \sim (A \oplus B \odot C)$

④ $F = A \odot B \oplus C$

정답 4번

출제 포인트

카르노맵의 출력 패턴을 홀수 패리티인 XOR 식으로 나타내고, XOR와 XNOR의 관계를 이용해 각 논리식의 동치 여부를 판단하는 문제이다.

조건 정리

카르노맵에서 $F = 1$인 입력은 $(A, B, C) = (0, 0, 1), (0, 1, 0), (1, 0, 0), (1, 1, 1)$이다. 즉 세 입력 중 1의 개수가 홀수일 때만 출력이 1이므로 $F = A \oplus B \oplus C$이다.

사용할 공식

XOR는 두 입력이 다를 때 1이고, XNOR는 두 입력이 같을 때 1이다. 따라서 $X \odot Y = \sim (X \oplus Y) = 1 \oplus X \oplus Y$ 이다. 이 항등식을 쓰면 각 선지에 포함된 XNOR를 XOR 식으로 바꾸어 카르노맵에서 얻은 F와 비교할 수 있다.

NoteBase

계산 과정

1. 카르노맵의 1이 놓인 네 입력을 확인하면 모두 1의 개수가 홀수이다.
2. 그러므로 $F = A \oplus B \oplus C$이다.
3. XNOR 하나는 XOR 결과를 한 번 반전하므로, 식 전체에 포함된 반전 횟수가 짝수이면 F, 홀수이면 $\sim F$가 된다.

선지 판단

① O — $A \oplus B \oplus C$는 세 입력 중 1의 개수가 홀수일 때 1이 되므로 카르노맵과 정확히 일치한다.
② O — $A \odot B \odot C = (1 \oplus A \oplus B) \odot C = A \oplus B \oplus C$이므로 F와 같다.
③ O — 괄호 안의 $A \oplus B \odot C$에는 XNOR에 따른 반전이 한 번 포함되어 $\sim F$가 되고, 바깥의 NOT이 이를 다시 반전하므로 결과는 F이다.
④ X — $A \odot B \oplus C = 1 \oplus A \oplus B \oplus C = \sim F$이므로 카르노맵과 반대 출력이다. 틀린 표현 $F = A \odot B \oplus C$는 올바르게는 $F = \sim (A \odot B \oplus C)$, 즉 $F = A \oplus B \oplus C$이다.

오답 포인트

②처럼 XNOR가 두 번 연속 나타나면 두 번의 반전이 상쇄되어 세 입력 XOR과 같아진다. 반면 ④는 XNOR가 한 번뿐이므로 전체 XOR의 반대가 된다는 점이 함정이다.

다시 나오면 이렇게 판단

카르노맵에서 1이 되는 입력의 1 개수가 모두 홀수인지 짝수인지 먼저 확인한다. 그다음 XNOR를 $1 \oplus X \oplus Y$로 바꾸어 반전 횟수의 홀짝을 세면 동치식을 빠르게 판별할 수 있다.

암기 포인트

홀수 개의 1이면 XOR, XNOR 한 번이면 XOR의 반대, 반전 두 번이면 다시 XOR이다.

NoteBase