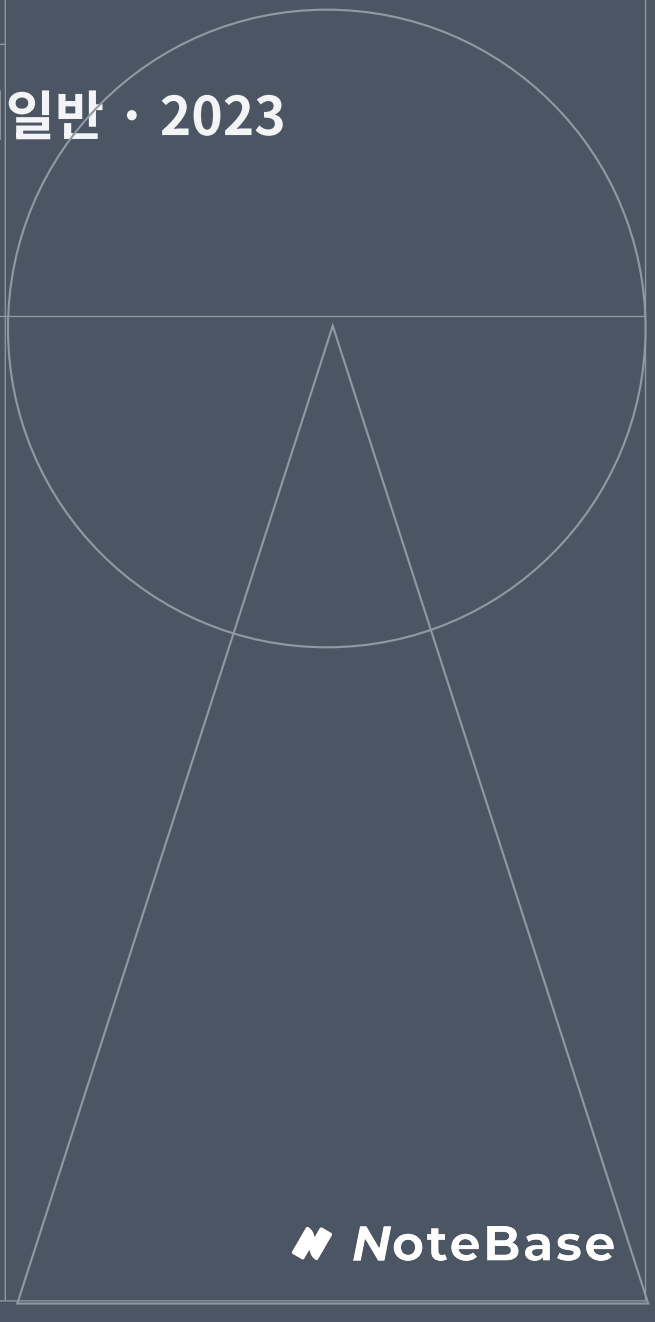
기출OX 해설집

공무원/지방직/9급 · 컴퓨터일반 · 2023



NoteBase

기출OX 해설집

공무원/지방직/9급 · 컴퓨터일반 · 2023

공무원/지방직/9급 · 컴퓨터일반 · 2023

문 1. 다음 중 문자 한 개를 표현하기 위해 필요한 비트 수가 가장 많은 문자 코드 체계는?

① ASCII

② BCD

③ EBCDIC

④ 유니코드(Unicode)

정답 4번

간단한 해설

초기 유니코드는 문자에 **16비트 코드값**을 부여하는 체계로 출발했다. 이 문항은 그 기준으로 비교하므로, 6~8비트를 사용하는 다른 문자 코드보다 비트 수가 많다.

선지별 해설

① ASCII: 기본 ASCII는 문자 하나를 **7비트**로 표현하여 128개의 코드값을 구별한다. 영문 대문자·소문자, 숫자, 기호와 제어 문자를 포함한다.

② BCD: 여기서 문자 코드로 제시된 BCD는 **6비트** 체계로, 64개의 코드값을 구별할 수 있다. 십진 숫자 한 자리를 4비트로 나타내는 수치 표현 방식의 BCD와 구별한다.

③ EBCDIC: 문자 하나를 **8비트**로 표현하여 256개의 코드값을 구별한다. BCD와 기본 ASCII보다 비트 수가 많지만, 이 문항에서 비교하는 유니코드의 16비트보다는 적다.

④ 유니코드(Unicode): 여러 언어의 문자를 통합하여 표현하는 체계이며, 초기에는 **16비트 문자 코드**로 설계되었다. 이 문항은 이 초기 기준을 사용한다. 현재 유니코드의 실제 저장 길이는 UTF-8·UTF-16·UTF-32 등 인코딩 방식에 따라 달라지므로, 모든 문자가 항상 16비트라고 일반화하지는 않는다.

이 문항의 비교 기준에서는 BCD 6비트 < ASCII 7비트 < EBCDIC 8비트 < 유니코드 16비트이므로, ④ 유니코드의 비트 수가 가장 많다.

NoteBase

문 2. 다음은 어떤 시스템의 성능 개선에 대한 내용이다. 성능 개선 후 프로그램 P의 실행에 걸리는 소요시간은? (단, 시스템에서 프로그램 P만 실행된다고 가정한다)

◯ 성능 개선 전에 프로그램 P의 특정 부분 A의 실행에 30초가 소요되었고, A를 포함한 전체 프로그램 P의 실행에 50초가 소요되었다.
◯ 시스템의 성능을 개선하여 A의 실행 속도를 2배 향상시켰다.
◯ A의 실행 속도 향상 외에 성능 개선으로 인한 조건 변화는 없다.

① 25초
② 30초
③ 35초
④ 40초

정답 3번

간단한 해설

실행 속도가 2배가 된 **A의 실행 시간만 절반**으로 줄고, 나머지 부분의 실행 시간은 그대로이다. 개선 후 A는 15초, 나머지는 20초가 걸리므로 전체 실행 시간은 35초이다.

계산 과정

A 이외 부분: 개선 전 전체 실행 시간 50초에서 A의 실행 시간 30초를 빼면 50 − 30 = 20초이다. A의 실행 속도 향상 외에는 조건 변화가 없으므로 이 부분은 **개선 후에도 20초**가 걸린다.

A 부분: 같은 작업에서 실행 시간은 실행 속도에 반비례한다. 따라서 **실행 속도가 2배이면 소요시간은 절반**이므로, 개선 후 A의 실행 시간은 $\frac{30}{2} = 15$초이다.

전체 실행 시간: 개선 후 A의 실행 시간과 나머지 부분의 실행 시간을 합하면 15 + 20 = 35초이다. 이는 ③의 35초와 일치한다.

NoteBase

문 3. 부울 변수 X, Y, Z에 대한 등식으로 옳지 않은 것은? (단, ·은 AND, +는 OR, ′는 NOT 연산을 의미한다)

① $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$

② $X \cdot (X + Y) = X \cdot X + Y$

③ $(X + Y) + Z = X + (Y + Z)$

④ $(X + Y)' = X' \cdot Y'$

정답 2번

간단한 해설

②는 분배법칙을 적용할 때 두 번째 항에 곱해져야 할 변수를 빠뜨렸다. X는 괄호 안의 **두 항 모두에 AND로 결합**해야 하므로, 올바른 전개식은 $X \cdot X + X \cdot Y$이며 이는 X로 간단해진다.

선지별 해설

부울 변수는 0 또는 1의 값을 갖는다. AND는 두 입력이 모두 1일 때만 1이고, OR는 하나 이상이 1이면 1이다. NOT은 0과 1을 서로 바꾼다.

① O: 부울대수에서는 **OR도 AND에 대해 분배**된다. $X = 1$이면 양변이 모두 1이고, $X = 0$이면 양변이 모두 $Y \cdot Z$가 된다. 따라서 $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$가 성립한다.

② X: 분배법칙을 적용하면 $X \cdot (X + Y) = X \cdot X + X \cdot Y$이다. 우변의 두 번째 항은 Y가 아니라 $X \cdot Y$여야 한다. 또한 흡수법칙에 따라 $X \cdot (X + Y) = X$이다. 예를 들어 $X = 0, Y = 1$이면 제시된 등식의 좌변은 0이지만 우변은 1이므로 두 값이 다르다.

③ O: OR 연산의 **결합법칙**이다. 어느 두 변수를 먼저 OR로 묶어도 세 변수 중 하나 이상이 1이면 결과가 1이고, 모두 0일 때만 결과가 0이다. 따라서 괄호의 위치가 달라져도 연산 결과는 같다.

④ O: **드모르간의 법칙**이다. $(X + Y)'$는 X와 Y가 모두 0일 때만 1이다. $X' \cdot Y'$도 두 변수의 부정값이 모두 1일 때, 즉 원래 두 변수가 모두 0일 때만 1이므로 양변이 같다.

NoteBase

문 4. IP(Internet Protocol)에 대한 설명으로 옳지 않은 것은?

① 전송 계층에서 사용되는 프로토콜이다.

② 비연결형 프로토콜이다.

③ IPv4에서 IP 주소의 길이가 32비트이다.

④ IP 데이터그램이 목적지에 성공적으로 도달하는 것을 보장하지 않는다.

정답 1번

간단한 해설

IP는 **네트워크 계층**에서 목적지 IP 주소를 바탕으로 데이터그램을 전달하는 프로토콜이다. 전송 계층의 대표적인 프로토콜은 TCP와 UDP이다.

선지별 해설

① X: IP는 OSI 7계층의 **네트워크 계층**에 해당하며, 목적지 IP 주소를 바탕으로 데이터그램을 다른 네트워크까지 전달하는 역할을 한다. TCP/IP 모델에서는 인터넷 계층에 해당한다. 선지의 '전송 계층'은 '네트워크 계층'으로 고쳐야 하며, 전송 계층의 대표적인 프로토콜은 TCP와 UDP이다.

② O: IP는 데이터를 보내기 전에 송신 측과 수신 측 사이에 **연결을 설정하지 않는** 비연결형 프로토콜이다. 각 데이터그램은 독립적으로 처리되므로 같은 목적지로 보내더라도 서로 다른 경로를 거칠 수 있다.

③ O: IPv4 주소는 **32비트**로 구성된다. 이를 8비트씩 네 부분으로 나누고 각 부분을 십진수로 표시한 것이 192.168.0.1과 같은 일반적인 IPv4 주소 표기이다.

④ O: IP는 가능한 범위에서 데이터그램을 전달하는 **최선형 전달** 방식을 사용한다. 네트워크 혼잡이나 장비 장애 등으로 데이터그램이 유실될 수 있으며, IP 자체는 수신 확인과 재전송으로 성공적인 도착을 보장하지 않는다.

NoteBase

문 5. 다음에서 제시한 시스템에서 주기억장치 주소의 각 필드의 비트 수를 바르게 연결한 것은? (단, 주기억장치 주소는 바이트 단위로 할당되고, 1 KB는 1,024바이트이다)

○ 캐시기억장치는 4-way 집합 연관 사상(set-associative mapping) 방식을 사용한다.

○ 캐시기억장치는 크기가 8 KB이고 전체 라인 수가 256개이다.

○ 주기억장치 주소는 길이가 32비트이고, 캐시기억장치 접근(access)과 관련하여 아래의 세 필드로 구분된다.

태그(tag)	세트(set)	오프셋(offset)

① 태그(tag): 20, 세트(set): 6, 오프셋(offset): 6

② 태그(tag): 20, 세트(set): 7, 오프셋(offset): 5

③ 태그(tag): 21, 세트(set): 5, 오프셋(offset): 6

④ 태그(tag): 21, 세트(set): 6, 오프셋(offset): 5

정답 4번

간단한 해설

캐시 라인 하나의 크기는 32바이트이므로 **오프셋은 5비트**이다. 4-way에서는 256개 라인을 4개씩 묶어 64개 세트를 구성하므로 세트는 6비트이고, 32비트 주소에서 나머지 21비트가 태그이다.

NoteBase

계산 과정

주어진 캐시는 용량이 8 KB이고 라인이 256개인 4-way 집합 연관 캐시이다. 32비트 주기억장치 주소를 태그·세트·오프셋으로 나누며, 바이트 단위 주소이므로 오프셋은 라인 안의 바이트 위치를 나타낸다.

오프셋: 1 KB는 1,024바이트이므로 캐시 용량은 $8 \times 1{,}024 = 8{,}192$바이트이다. 이를 전체 라인 수로 나누면 한 라인의 크기는 $\frac{8{,}192}{256} = 32$바이트이다. **라인 안의 32개 바이트를 구별**해야 하므로 $32 = 2^5$에 따라 오프셋은 5비트이다.

세트: 4-way는 **한 세트에 라인 4개**가 있다는 뜻이다. 따라서 세트 수는 $\frac{256}{4} = 64$개이고, $64 = 2^6$이므로 세트 필드는 6비트이다. 세트 필드는 해당 세트를 선택하며, 그 안의 4개 라인은 태그를 비교하여 구별한다.

태그: 태그는 선택된 세트에 저장된 데이터가 어느 주기억장치 블록의 것인지 식별한다. 전체 주소 32비트에서 세트 6비트와 오프셋 5비트를 제외하므로 태그는 $32 - 6 - 5 = 21$비트이다.

따라서 태그 21비트, 세트 6비트, 오프셋 5비트이며, 이 구성은 ④와 일치한다.

NoteBase

문 6. 2의 보수로 표현된 부호 있는(signed) n비트 2진 정수에 대한 설명으로 옳지 않은 것은?

① 최저 음수의 값은 $-(2^{n-1}-1)$이다.

② 0에 대한 표현이 한 가지이다.

③ 0이 아닌 2진 정수 A의 2의 보수는 $(2^n - A)$이다.

④ 0이 아닌 2진 정수 A의 2의 보수는 A의 1의 보수에 1을 더해서 구할 수 있다.

정답 1번

간단한 해설

2의 보수로 표현한 n비트 정수의 최솟값은 -2^{n-1}이다. **최상위 비트만 1**인 비트열이 이 값을 나타내며, 양수의 최댓값에 부호만 붙인 값보다 1 작다.

선지별 해설

① X: 2의 보수 표현에서는 최상위 비트의 가중치가 -2^{n-1}이고 나머지 비트의 가중치는 양수이다. 따라서 최상위 비트만 1이고 나머지가 모두 0일 때 최솟값 -2^{n-1}이 된다. 표현 범위는 $-2^{n-1} \le x \le 2^{n-1}-1$이며, **음수와 양수의 범위가 대칭이 아니다**. 예를 들어 4비트에서는 1000이 -8, 0111이 7을 나타낸다. 선지의 $-(2^{n-1}-1)$은 1의 보수 표현에서의 최솟값이다.

② O: 2의 보수 표현에서 0은 **모든 비트가 0인 비트열 하나**로만 나타낸다. 최상위 비트가 1이면 나머지 비트를 모두 1로 해도 값이 -1이므로 별도의 음의 0이 생기지 않는다.

③ O: n비트 비트열을 부호 없는 수로 읽은 값을 A라고 하면, A가 0이 아닐 때 그 2의 보수는 $2^n - A$이다. 두 값을 더하면 2^n이 되어 **n비트 범위를 넘는 자리올림을 버린 결과가 0**이므로, 고정된 n비트에서 덧셈의 역원으로 작용한다.

④ O: 1의 보수는 n비트의 각 비트를 0과 1로 뒤집은 것으로, 그 값은 $(2^n - 1) - A$이다. 여기에 1을 더하면 $2^n - A$가 되므로 **비트 반전 후 1을 더하면** 2의 보수를 구할 수 있다. 예를 들어 4비트의 0101을 반전하면 1010이고, 여기에 1을 더하면 1011이 된다. 0101은 5, 1011은 2의 보수 표현에서 -5를 나타낸다.

NoteBase

문 7. 10진수 45.1875를 2진수로 변환한 것은?

① 101100.0011

② 101100.0101

③ 101101.0011

④ 101101.0101

정답 3번

간단한 해설

45는 32 + 8 + 4 + 1이므로 2진수 정수부는 101101이다. 소수부는 **2를 반복해서 곱해 얻는 정수 부분**을 차례로 쓰면 0.0011이 되어, 전체 값은 101101.0011이다.

계산 과정

정수부 45: 2진수의 정수 자릿값은 오른쪽부터 1, 2, 4, 8, 16, 32로 두 배씩 커진다. 45 = 32 + 8 + 4 + 1이므로 자릿값 32, 16, 8, 4, 2, 1에 대응하는 숫자는 각각 1, 0, 1, 1, 0, 1이다. 따라서 정수부는 101101이다.

소수부 0.1875: 소수에 2를 곱하면 2진 소수점 바로 아래의 숫자가 정수 부분으로 이동한다. 따라서 **매번 정수 부분을 기록하고 남은 소수 부분에 다시 2를 곱한다**.
0.1875 × 2 = 0.375 → 정수 부분 0
0.375 × 2 = 0.75 → 정수 부분 0
0.75 × 2 = 1.5 → 정수 부분 1, 남은 소수 부분 0.5
0.5 × 2 = 1.0 → 정수 부분 1, 남은 소수 부분 0
남은 소수 부분이 0이므로 변환을 끝낸다. 기록한 정수 부분을 나온 순서대로 읽으면 0011이므로 소수부는 0.0011이다.

두 부분을 합치면 $(45.1875)_{10} = (101101.0011)_2$이며, 이는 ③의 값과 일치한다.

NoteBase

문 8. 운영체제에서 다음 설명에 해당하는 페이지 교체 알고리즘은?

페이지 교체가 필요한 시점에서 최근 가장 오랫동안 사용되지 않은 페이지를 제거하여 교체한다.

① 최적(optimal) 교체 알고리즘
② FIFO(First In First Out) 교체 알고리즘
③ LRU(Least Recently Used) 교체 알고리즘
④ LFU(Least Frequently Used) 교체 알고리즘

정답 3번

간단한 해설

'가장 오랫동안 사용되지 않은'은 **마지막 사용 시점이 가장 오래된 페이지**를 뜻한다. 사용 횟수가 아니라 최근 사용 시점을 기준으로 교체하므로 LRU에 해당한다.

개념 구분

LRU: 메모리에 있는 페이지 중 **마지막으로 사용한 시점이 가장 오래된 페이지**를 교체한다. 예를 들어 A와 B가 메모리에 있고 A → B → A 순으로 참조했다면, 마지막 참조가 더 오래된 B를 교체한다.

FIFO: 마지막 사용 시점이 아니라 **메모리에 먼저 들어온 순서**로 교체한다. 먼저 들어온 페이지를 최근에 다시 사용했더라도 교체 순서가 뒤로 바뀌지 않는다는 점에서 LRU와 다르다.

LFU: **사용 횟수가 가장 적은 페이지**를 교체한다. '얼마나 오래 사용하지 않았는가'를 따지는 LRU와 달리 '얼마나 적게 사용했는가'를 기준으로 삼는다.

최적 교체 알고리즘: **앞으로 사용될 때까지의 시간이 가장 긴 페이지**를 교체한다. 과거의 사용 기록을 기준으로 하는 LRU와 달리 미래의 참조 정보를 기준으로 삼는다.

NoteBase

문 9. ICT 기술에 대한 설명으로 옳지 않은 것은?

① 기계학습(machine learning)의 학습 방법에는 지도학습(supervised learning), 비지도학습(unsupervised learning), 강화학습(reinforcement learning) 등이 있다.

② 가상현실(virtual reality)은 가상의 공간과 사물 등을 만들어, 일상적으로 경험하기 어려운 상황을 실제처럼 체험할 수 있도록 해준다.

③ RFID(Radio Frequency IDentification)에서 수동형 태그는 내장된 배터리를 사용하여 무선 신호를 발생시킨다.

④ 지그비(ZigBee)는 저비용, 저전력 무선 네트워크 기술로 센서 네트워크에서 사용할 수 있다.

정답 3번

간단한 해설

RFID의 **수동형 태그에는 내장 배터리가 없으며**, 리더가 보내는 전파에서 얻은 에너지로 작동한다. 내장 배터리를 이용하여 자체적으로 무선 신호를 송신하는 것은 **능동형 태그**의 특징이다.

선지별 해설

① O: 기계학습은 데이터나 경험으로부터 규칙을 학습하는 기술이다. 지도학습은 **입력과 정답이 짝지어진 데이터**로 예측 관계를 배우고, 비지도학습은 정답이 주어지지 않은 데이터에서 군집이나 패턴을 찾는다. 강화학습은 행동에 따른 보상을 바탕으로 누적 보상을 높이는 행동 방식을 학습한다.

② O: 가상현실은 컴퓨터로 만든 가상 공간과 사물을 이용해 사용자가 **그 환경 안에 있는 것처럼 체험**하도록 하는 기술이다. 예를 들어 가상의 비행 환경에서 조종 훈련을 하면 실제 비행의 위험을 줄이면서 여러 상황을 경험할 수 있다.

③ X: RFID는 전파를 이용하여 태그의 정보를 비접촉으로 식별하는 기술이다. 수동형 태그는 **내장 배터리 없이 리더의 전파에서 얻은 에너지**로 회로를 작동시키고 정보를 응답한다. 반면 능동형 태그는 내장 배터리를 사용하여 자체적으로 무선 신호를 송신한다. 선지의 '수동형 태그'를 '능동형 태그'로 고쳐야 한다.

④ O: 지그비는 저비용·저전력을 지향하는 무선 네트워크 기술로, 비교적 적은 양의 데이터를 주고받는 센서 네트워크에 적합하다. 온도·습도 측정값처럼 작은 데이터를 전송하는 데 활용할 수 있으며, **낮은 전력 소모**는 배터리로 작동하는 센서의 사용 시간을 늘리는 데 도움이 된다.

NoteBase

문 10. 다음 조건을 만족하는 가상기억장치에서 가상 페이지 번호(virtual page number)와 페이지 오프셋의 비트 수를 바르게 연결한 것은?

◯ 페이징 기법을 사용하며, 페이지 크기는 2,048바이트이다.
◯ 가상 주소는 길이가 32비트이고, 가상 페이지 번호와 페이지 오프셋으로 구분된다.

① 가상 페이지 번호: 11, 페이지 오프셋: 21
② 가상 페이지 번호: 13, 페이지 오프셋: 19
③ 가상 페이지 번호: 19, 페이지 오프셋: 13
④ 가상 페이지 번호: 21, 페이지 오프셋: 11

정답 4번

간단한 해설

페이지 크기 2,048바이트는 2^{11}바이트이므로, 페이지 안의 바이트 위치를 나타내는 **오프셋은 11비트**이다. 32비트 가상 주소에서 이를 제외한 **21비트가 가상 페이지 번호**가 된다.

계산 과정

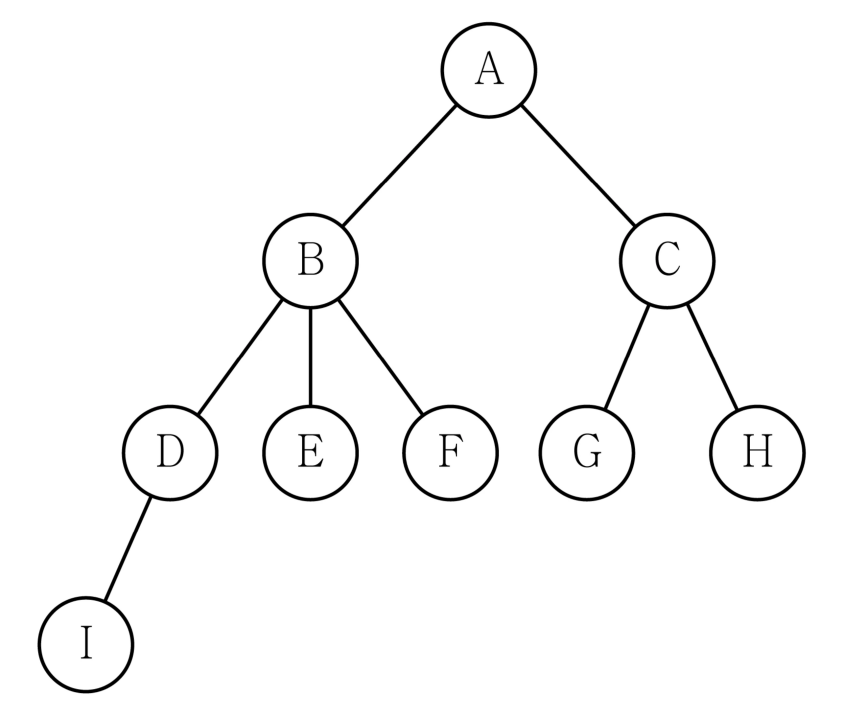
페이지 오프셋: 오프셋은 한 페이지 안에서의 바이트 위치를 나타낸다. n비트로 구분할 수 있는 위치는 2^n개이므로, 페이지 크기가 2^n바이트이면 오프셋에 n비트가 필요하다. 여기서는 $2{,}048 = 2^{11}$이므로 오프셋은 11비트이며, 페이지 안의 0~2,047번 바이트를 구분한다.

가상 페이지 번호: 어느 가상 페이지인지를 나타내는 부분이다. 가상 주소가 **가상 페이지 번호와 페이지 오프셋으로만 구분**되므로, 두 부분의 비트 수를 합하면 전체 주소 길이인 32비트가 된다. 따라서 가상 페이지 번호의 비트 수는 $32 - 11 = 21$비트이다.

가상 페이지 번호 21비트, 페이지 오프셋 11비트이므로 ④의 연결과 일치한다.

NoteBase

문 11. 다음 트리에 대한 설명으로 옳지 않은 것은?



① A 노드의 차수(degree)는 2이다.

② 트리의 차수는 4이다.

③ D 노드는 F 노드의 형제(sibling) 노드이다.

④ C 노드는 G 노드의 부모(parent) 노드이다.

정답 2번

간단한 해설

트리의 차수는 각 노드가 가진 **자식 수의 최댓값**이다. 그림에서 자식이 가장 많은 B의 자식은 D·E·F의 3개이므로, 트리의 차수는 4가 아니라 **3**이다.

NoteBase

선지별 해설

① O: 트리에서 노드의 차수는 그 노드의 **자식 노드 수**이다. A의 자식은 B와 C이므로 A의 차수는 2이다.

② X: 트리의 차수는 **노드 차수 중 최댓값**이다. A는 2, B는 3, C는 2, D는 1이며, 자식이 없는 E·F·G·H·I는 모두 0이다. 따라서 트리의 차수는 4가 아니라 3이다.

③ O: 형제 노드는 **부모가 같은 노드**이다. D와 F는 모두 B의 자식이므로 서로 형제 노드이다. E도 이들과 형제 관계이다.

④ O: 부모 노드는 어떤 노드의 **바로 위에 연결된 노드**이다. 그림에서 C의 바로 아래에 G와 H가 연결되어 있으므로 C는 G의 부모 노드이다.

NoteBase

문 12. 다음에서 설명하는 UML(Unified Modeling Language) 다이어그램(diagram)은?

객체들이 어떻게 상호 동작하는지를 메시지 순서에 초점을 맞춰 나타낸 것으로, 어떠한 작업이 객체 간에 발생하는지를 시간 순서에 따라 보여준다.

① 클래스(class) 다이어그램
② 순차(sequence) 다이어그램
③ 배치(deployment) 다이어그램
④ 컴포넌트(component) 다이어그램

정답 2번

간단한 해설

순차(sequence) 다이어그램은 객체 사이에 오가는 **메시지를 시간 순서대로** 표현한다. 제시문의 '메시지 순서'와 '시간 순서'가 이를 가리킨다.

핵심 풀이

순차(sequence) 다이어그램: 객체 간 상호작용에서 **어떤 메시지가 먼저 전달되는지**를 보여 준다. 참여 객체의 생명선을 세로로 표시하고, 객체 사이의 메시지를 화살표로 나타낸다. 시간은 보통 위에서 아래로 흐르므로, 메시지의 위아래 위치를 통해 전달 순서를 파악할 수 있다.

다른 후보와의 구분: 클래스 다이어그램은 클래스의 속성·연산과 클래스 사이의 관계를, 컴포넌트 다이어그램은 소프트웨어 구성 요소와 그 의존 관계를, 배치 다이어그램은 실행 환경인 노드와 소프트웨어의 배치 관계를 나타낸다. 이들은 **시스템의 구조**를 표현하므로, 객체 간 메시지의 시간적 순서를 강조한 제시문과 구별된다.

NoteBase

문 13. 리틀 엔디안(little endian) 방식을 사용하는 시스템에서 다음 C 프로그램의 출력 결과는? (단, int의 크기는 4바이트이다)

```
#include <stdio.h>

int main() {

    char i;

    union {

        int int_arr[2];

        char char_arr[8];

    } endian;

    for (i = 0; i < 8; i++)

        endian.char_arr[i] = i + 16;

    printf("%x", endian.int_arr[1]);

    return 0;

}
```

① 10111213

② 13121110

③ 14151617

④ 17161514

정답 4번

간단한 해설

int_arr[1]이 공유하는 네 바이트에는 낮은 주소부터 16진수 14, 15, 16, 17이 저장된다. **리틀 엔디안은 낮은 주소에 하위 바이트를 저장**하므로, 이를 정수로 읽어 %x로 출력하면 17161514가 된다.

NoteBase

계산 과정

1. 공용체의 메모리 대응: union의 멤버 int_arr와 char_arr는 **같은 메모리 공간을 공유**한다. int가 4바이트이므로 int_arr[0]은 char_arr[0]~char_arr[3]에, int_arr[1]은 char_arr[4]~char_arr[7]에 대응한다.

2. 반복문 실행: i는 0부터 7까지 증가하며 char_arr[i]에 i + 16을 저장한다. 따라서 char_arr[0]부터 차례로 저장되는 값은 10진수로 16, 17, 18, 19, 20, 21, 22, 23이다. 이를 16진수로 바꾸면 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17이다.

3. int_arr[1]의 값 해석: 두 번째 정수가 차지하는 char_arr[4]~char_arr[7]에는 낮은 주소부터 0x14, 0x15, 0x16, 0x17이 들어 있다. 리틀 엔디안에서는 **낮은 주소의 바이트가 정수의 하위 자릿수**를 맡으므로, 0x14가 최하위 바이트이고 0x17이 최상위 바이트이다. 따라서 정수의 값은 0x17161514이다.

4. 출력 결과: printf의 %x는 정수를 0x 접두사 없이 16진수로 출력한다. 따라서 출력되는 문자열은 17161514이며, ④의 값과 일치한다.

NoteBase

문 14. 2의 보수로 표현된 부호 있는 8비트 2진 정수 10110101을 2비트만큼 산술 우측 시프트(arithmetic right shift)한 결과는?

① 00101101

② 11010100

③ 11010111

④ 11101101

정답 4번

간단한 해설

산술 우측 시프트는 왼쪽의 빈자리를 **기존 부호 비트로 채운다**. 10110101의 부호 비트는 1이므로, 오른쪽으로 두 칸 이동하고 왼쪽 두 자리를 1로 채우면 11101101이 된다.

계산 과정

2의 보수로 표현된 부호 있는 8비트 정수에서 맨 왼쪽 비트는 부호 비트이다. 주어진 10110101은 부호 비트가 1이다. 산술 우측 시프트는 각 비트를 오른쪽으로 이동시키면서 **왼쪽 빈자리에 부호 비트를 복사**하고, 오른쪽 끝을 벗어난 비트는 버린다. 비트열의 길이는 8비트로 유지한다.

1비트 이동: 10110101 → 11011010. 맨 오른쪽의 1이 버려지고, 맨 왼쪽에 부호 비트 1이 들어간다. 이는 두 번의 이동 중 첫 번째 이동 결과이다.

다시 1비트 이동: 11011010 → 11101101. 이번에는 맨 오른쪽의 0이 버려지고, 맨 왼쪽에 다시 1이 들어간다.

따라서 2비트 산술 우측 시프트 결과는 11101101이며, ④의 비트열과 일치한다.

NoteBase

문 15. 다음 Java 프로그램의 출력 결과는?

```
public class Result {

    public static void main(String[] args) {

        int sum = 0;

        for (int i = 1; i <= 10; i++)

            if (i % 2 != 0 && i % 5 != 0)

                sum += i;

        System.out.println(sum);

    }

}
```

① 15
② 20
③ 25
④ 55

정답 2번

간단한 해설

조건식의 &&는 두 조건을 모두 만족해야 참이므로, 1부터 10까지 중 **2의 배수도 5의 배수도 아닌 수**만 더한다. 해당하는 수는 1, 3, 7, 9이며 합계는 20이다.

NoteBase

계산 과정

반복 범위: sum은 누적 합을 저장하는 변수로, 처음 값은 0이다. for문은 i를 1로 시작하여 i <= 10인 동안 실행하고 매번 i++로 1씩 증가시키므로, i는 1부터 10까지의 값을 차례로 갖는다.

조건 판정: %는 나머지를 구하는 연산자이고, !=는 두 값이 서로 다름을 뜻한다. i % 2 != 0은 2로 나누어떨어지지 않는다는 조건이고, i % 5 != 0은 5로 나누어떨어지지 않는다는 조건이다. &&로 연결되어 있으므로 **두 조건이 모두 참일 때만** sum += i를 실행한다. 따라서 짝수 2, 4, 6, 8, 10을 제외하고, 홀수 중에서도 5는 제외한다.

누적 계산: 조건을 만족하는 i는 1, 3, 7, 9이다. sum += i는 현재 sum에 i를 더하여 다시 sum에 저장하는 연산이다. sum은 0 → 1 → 4 → 11 → 20으로 변하며, 조건을 만족하지 않는 i에서는 그대로 유지된다. 최종 합은 1 + 3 + 7 + 9 = 20이다.

출력: for문의 반복 대상은 if문이고, 그 안의 sum += i는 조건이 참일 때만 실행된다. System.out.println(sum)은 **반복문이 끝난 뒤 한 번** 실행되어 최종 누적 합 20을 출력한다. 이는 ②의 값과 일치한다.

NoteBase

문 16. TCP/IP 프로토콜 계층 구조에서 다음 중 나머지 셋과 다른 계층에 속하는 프로토콜은?

① HTTP

② SMTP

③ DNS

④ ICMP

정답 4번

간단한 해설

ICMP는 IP 패킷 처리 중 발생한 오류를 알리거나 네트워크 상태를 진단하는 **인터넷 계층** 프로토콜이다.
웹·전자우편 등 응용 서비스를 위한 프로토콜과 계층이 다르다.

선지별 해설

TCP/IP 계층은 프로토콜이 수행하는 역할을 기준으로 구분한다. 응용 계층은 응용 서비스에 필요한 통신 규칙을 제공하고, 인터넷 계층은 IP 패킷 전달과 그에 필요한 제어 기능을 담당한다.

① HTTP: 웹 클라이언트와 서버 사이에서 요청과 응답을 주고받는 프로토콜로, **응용 계층**에 속한다.

② SMTP: 전자우편을 보내거나 메일 서버 사이에서 전달하는 프로토콜로, 응용 계층에 속한다.

③ DNS: 도메인 이름에 대응하는 IP 주소 등의 정보를 조회하는 프로토콜로, 응용 계층에 속한다. IP 주소를 조회한다는 이유로 인터넷 계층에 속하는 것은 아니다.

④ ICMP: 목적지에 도달할 수 없다는 오류를 알리거나 에코 요청·응답으로 통신 상태를 진단하는 프로토콜이다. IP의 동작을 보조하므로 **인터넷 계층**에 속한다.

HTTP·SMTP·DNS는 응용 계층에 속하고, ④ ICMP만 인터넷 계층에 속하므로 나머지 셋과 계층이 다르다.

NoteBase

문 17. 데이터베이스 언어에 대한 설명으로 옳지 않은 것은?

① 데이터 제어어(data control language)는 사용자가 데이터에 대한 검색, 삽입, 삭제, 수정 등의 처리를 DBMS에 요구하기 위해 사용되는 언어이다.

② 데이터 제어어는 데이터베이스의 보안, 무결성, 회복(recovery) 등을 지원하기 위해 사용된다.

③ 절차적 데이터 조작어(procedural data manipulation language)는 사용자가 원하는 데이터와 그 데이터로의 접근 방법을 명시해야 하는 언어이다.

④ 데이터 정의어(data definition language)는 데이터베이스 스키마의 생성, 변경, 삭제 등에 사용되는 언어이다.

정답 1번

간단한 해설

검색·삽입·삭제·수정처럼 데이터를 처리하도록 요구하는 언어는 **데이터 조작어(DML)**이다. 데이터 제어어(DCL)는 데이터베이스의 보안·무결성·회복 등을 지원하는 데 사용된다.

선지별 해설

① X: 사용자가 DBMS에 데이터의 검색·삽입·삭제·수정을 요구할 때 사용하는 언어는 **데이터 조작어(DML)**이다. 따라서 선지의 '데이터 제어어'를 '데이터 조작어'로 고쳐야 한다. 예를 들어 학생의 성적을 조회하거나 저장된 성적 값을 바꾸는 작업은 데이터 조작에 해당한다.

② O: 데이터 제어어(DCL)는 데이터베이스를 안전하고 일관되게 사용할 수 있도록 **보안·무결성·회복** 등의 제어 기능을 지원한다. 보안은 허가되지 않은 접근을 막는 것이고, 무결성은 데이터의 정확성과 일관성을 유지하는 것이며, 회복은 장애가 발생했을 때 데이터베이스를 정상적인 상태로 되돌리는 것이다.

③ O: 절차적 데이터 조작어는 **원하는 데이터와 그 데이터에 접근하는 방법**을 함께 명시한다. 무엇을 얻을지뿐 아니라 어떤 절차로 얻을지도 사용자가 지정하는 방식이다. 반면 비절차적 데이터 조작어는 원하는 데이터를 명시하고, 구체적인 접근 방법의 결정은 DBMS에 맡긴다.

④ O: 데이터 정의어(DDL)는 데이터베이스의 구조를 기술하는 **스키마를 생성·변경·삭제**하는 데 사용된다. 예를 들어 테이블을 만들거나 열을 추가하는 작업이 해당한다. 테이블 자체를 삭제하는 것은 구조를 없애는 DDL 작업이고, 테이블을 유지하면서 그 안의 특정 행을 삭제하는 것은 DML 작업이다.

NoteBase

문 18. TCP(Transmission Control Protocol)에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 네트워크 계층에서 사용되는 프로토콜이다.
ㄴ. 흐름 제어와 혼잡 제어를 수행한다.
ㄷ. 연결지향형 프로토콜이다.
ㄹ. IP 주소를 이용하여 데이터그램을 목적지 호스트까지 전송하는 역할을 한다.

① ㄱ, ㄴ
② ㄱ, ㄹ
③ ㄴ, ㄷ
④ ㄷ, ㄹ

정답 3번

간단한 해설

TCP는 수신 측의 처리 능력에 맞추는 **흐름 제어**와 네트워크의 혼잡을 줄이는 **혼잡 제어**를 수행한다. 또한 데이터를 보내기 전에 상대와 논리적 연결을 설정하는 **연결지향형** 전송 계층 프로토콜이다.

보기별 해설

ㄱ. X: TCP는 네트워크 계층이 아니라 **전송 계층**의 프로토콜이다. 응용 프로그램 사이에서 데이터가 신뢰성 있게 전달되도록 하며, 네트워크 계층의 대표적인 프로토콜은 IP이다.

ㄴ. O: 흐름 제어는 **수신 측이 처리할 수 있는 양**에 맞추어 송신량을 조절하는 기능이다. 혼잡 제어는 **네트워크의 혼잡 상태**에 따라 송신량을 조절하는 기능이다. TCP는 두 기능을 모두 수행한다.

ㄷ. O: TCP는 데이터 전송 전에 양쪽이 연결 설정 메시지를 주고받는 **3방향 핸드셰이크**로 논리적 연결을 설정한다. 이 연결을 바탕으로 데이터를 주고받으므로 연결지향형 프로토콜이다.

ㄹ. X: IP 주소를 이용해 데이터그램을 목적지 호스트까지 전달하는 것은 **IP의 역할**이다. TCP는 전송 계층에서 포트 번호로 통신할 응용 프로그램을 구별하고, 순서 제어와 재전송 등을 통해 신뢰성 있는 데이터 전송을 제공한다.

따라서 TCP에 대한 설명으로 옳은 보기는 ㄴ과 ㄷ이며, 이 조합은 ③에 해당한다.

NoteBase

문 19. 다음은 프로세스가 준비 상태 큐에 도착한 시간과 프로세스를 처리하는 데 필요한 실행 시간을 보여준다. 선점형 SJF(Shortest Job First) 스케줄링 알고리즘인 SRT(Shortest Remaining Time) 알고리즘을 사용할 경우, 프로세스들의 대기 시간 총합은? (단, 프로세스 간 문맥 교환에 따른 오버헤드는 무시하며, 주어진 4개 프로세스 외에 처리할 다른 프로세스는 없다고 가정한다)

프로세스	도착 시간	실행 시간
P_1	0	30
P_2	5	10
P_3	10	15
P_4	15	10

① 40

② 45

③ 50

④ 55

정답 3번

간단한 해설

SRT는 도착한 프로세스 중 **남은 실행 시간이 가장 짧은 프로세스**를 실행한다. 각 프로세스의 대기 시간은 35, 0, 15, 0이므로 총합은 50이다.

NoteBase

계산 과정

SRT는 실행 중인 프로세스까지 포함하여 **남은 실행 시간이 가장 짧은 프로세스**에 CPU를 배정한다. 표의 도착 시각과 실행 시간은 각각 P_1이 (0, 30), P_2가 (5, 10), P_3이 (10, 15), P_4가 (15, 10)이다. 문맥 교환 오버헤드는 무시하며, 시간은 표와 같은 단위로 계산한다.

0~5: 도착한 프로세스가 P_1뿐이므로 5만큼 실행한다. 시각 5에 P_1의 남은 시간은 30 − 5 = 25이다. 이때 실행 시간이 10인 P_2가 도착하므로 P_1을 중단하고 P_2를 실행한다.

5~15: P_2를 실행한다. 시각 10에 P_3이 도착하지만, P_2의 남은 시간은 5로 P_3의 15보다 짧으므로 **실행을 계속한다**. P_2는 시각 15에 완료된다.

15~25: 새로 도착한 P_4의 실행 시간 10이 P_1의 남은 시간 25, P_3의 실행 시간 15보다 짧다. 따라서 P_4를 실행하여 시각 25에 완료한다.

25~40, 40~65: 남아 있는 P_3의 실행 시간 15가 P_1의 남은 시간 25보다 짧으므로 P_3을 먼저 실행하여 시각 40에 완료한다. 이후 P_1을 25만큼 더 실행하여 시각 65에 완료한다.

대기 시간은 준비 상태에서 CPU를 기다린 시간이다. 이 문제에서는 도착부터 완료까지의 시간이 실행 시간과 대기 시간으로 나뉘므로, **대기 시간 = 완료 시각 − 도착 시각 − 실행 시간**으로 구한다. 중간에 실행이 중단된 뒤 다시 기다린 시간도 포함한다.

P_1: 65 − 0 − 30 = 35이다. 처음 0~5에 실행되었더라도, 중단된 뒤 5~40에 기다린 35를 대기 시간에 포함한다.

P_2: 15 − 5 − 10 = 0이다. 도착하자마자 실행되어 중단 없이 완료된다.

P_3: 40 − 10 − 15 = 15이다. 도착 시각 10부터 실행을 시작한 시각 25까지 기다린다.

P_4: 25 − 15 − 10 = 0이다. 도착하자마자 실행되어 완료된다.

대기 시간 총합은 35 + 0 + 15 + 0 = 50이며, 선택지 ③의 50에 해당한다.

NoteBase

문 20. 공백 상태인 이진 탐색 트리(binary search tree)에 1부터 5까지의 정수를 삽입하고자 한다. 삽입 결과, 이진 탐색 트리의 높이가 가장 높은 삽입 순서는?

① 1, 2, 3, 4, 5

② 1, 4, 2, 5, 3

③ 3, 1, 4, 2, 5

④ 5, 3, 4, 1, 2

정답 1번

간단한 해설

작은 값부터 차례로 삽입하면 새 값이 매번 기존 노드의 오른쪽에 붙어 **한쪽으로 치우친 트리**가 된다. 5개 노드가 하나의 경로로 이어지므로 높이가 가장 커진다.

선지별 해설

이진 탐색 트리는 각 노드보다 작은 값을 왼쪽 서브트리에, 큰 값을 오른쪽 서브트리에 둔다. 새 값은 루트부터 비교하며 내려가 처음 만난 빈 자리에 삽입한다. 여기서는 높이를 루트에서 가장 먼 잎 노드까지의 간선 수로 센다.

① 1, 2, 3, 4, 5: 1이 루트가 되고, 2부터 5까지는 앞서 삽입한 모든 값보다 크므로 오른쪽으로 계속 내려가 삽입된다. **1 → 2 → 3 → 4 → 5가 하나의 경로**를 이루며, 간선이 4개이므로 높이는 4이다.

② 1, 4, 2, 5, 3: 루트 1의 오른쪽 자식은 4이다. 2는 4의 왼쪽 자식, 5는 4의 오른쪽 자식이 된다. 마지막 3은 1보다 크고 4보다 작으며 2보다 크므로 2의 오른쪽 자식이 된다. 가장 긴 경로는 **1 → 4 → 2 → 3**이며 높이는 3이다.

③ 3, 1, 4, 2, 5: 루트 3의 왼쪽 자식은 1, 오른쪽 자식은 4이다. 2는 1의 오른쪽 자식, 5는 4의 오른쪽 자식이 된다. 가장 긴 경로는 3 → 1 → 2와 3 → 4 → 5로, **두 경로 모두 간선이 2개**이므로 높이는 2이다.

④ 5, 3, 4, 1, 2: 루트 5의 왼쪽 자식은 3이다. 4는 3의 오른쪽 자식, 1은 3의 왼쪽 자식이 된다. 마지막 2는 5와 3보다 작고 1보다 크므로 1의 오른쪽 자식이 된다. 가장 긴 경로는 **5 → 3 → 1 → 2**이며 높이는 3이다.

높이를 비교하면 ①은 4, ②와 ④는 3, ③은 2이다. 따라서 가장 높은 트리를 만드는 삽입 순서는 ①의 1, 2, 3, 4, 5이다.

NoteBase