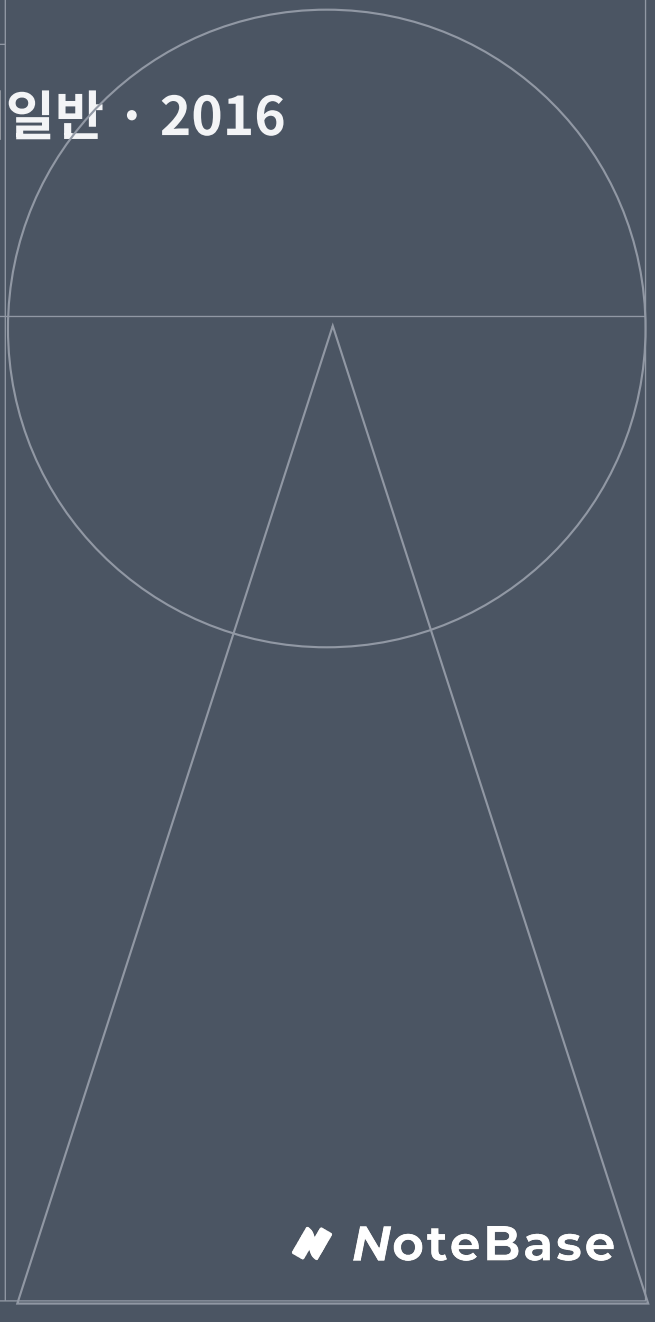
기출OX 해설집

공무원/계리직/9급 · 컴퓨터일반 · 2016



NoteBase

기출OX 해설집

공무원/계리직/9급 · 컴퓨터일반 · 2016

공무원/계리직/9급 · 컴퓨터일반 · 2016

문 1. 2진수 11110000과 10101010에 대해 XOR 논리 연산을 수행한 결과값을 16진수로 바르게 표현한 것은?

① 5A ② 6B ③ A5 ④ B6

정답 1번

출제 포인트

이 문제는 두 2진수의 각 비트를 XOR 연산한 뒤, 결과를 4비트씩 묶어 16진수로 변환하는 문제이다.

조건 정리

주어진 값: 2진수 11110000과 10101010
구해야 할 값: 두 수를 XOR 연산한 결과의 16진수 표현
주의할 조건: XOR은 같은 자리의 두 비트가 서로 다르면 1, 같으면 0이 된다.

사용할 공식

0 XOR 0=0, 0 XOR 1=1, 1 XOR 0=1, 1 XOR 1=0

XOR은 각 자릿수를 독립적으로 비교하는 비트 논리 연산이므로 이 규칙을 같은 위치의 비트마다 적용한다.

계산 과정

1. 같은 자리끼리 XOR을 계산한다.
11110000 XOR 10101010=01011010
2. 결과를 오른쪽부터 4비트씩 묶으면 0101과 1010이다.
3. 0101은 16진수 5이고 1010은 16진수 A이므로 결과는 5A이다.

NoteBase

선지 판단

① O — 계산 결과 01011010을 4비트씩 변환하면 0101은 5, 1010은 A이므로 5A가 된다.

② X — 6B는 계산 결과와 일치하지 않는다. 틀린 표현 '6B'는 '5A'로 고쳐야 한다.

③ X — A5는 두 16진수 자릿수의 순서를 뒤바꾼 값이다. 틀린 표현 'A5'는 상위 4비트 0101을 먼저 적은 '5A'로 고쳐야 한다.

④ X — B6는 계산 결과와 일치하지 않는다. 틀린 표현 'B6'는 '5A'로 고쳐야 한다.

오답 포인트

XOR을 OR처럼 계산하거나, 01011010을 4비트씩 나눈 뒤 하위 묶음 1010부터 써서 A5로 적는 실수가 함정이다. 16진수는 상위 4비트 묶음부터 차례대로 적는다.

다시 나오면 이렇게 판단

각 자리의 두 비트가 다르면 1, 같으면 0으로 계산한 뒤, 결과를 오른쪽부터 4비트씩 묶고 각 묶음을 상위 자리부터 16진수 한 자리로 바꾼다.

암기 포인트

XOR은 다르면 1, 같으면 0이다. 2진수를 16진수로 바꿀 때는 4비트가 16진수 한 자리이다.

NoteBase

문 2. 무선 네트워크 방식에 대한 설명으로 옳은 것은?

① 블루투스(Bluetooth)는 동일한 유형의 기기 간에만 통신이 가능하다.

② NFC 방식이 블루투스 방식보다 최대 전송 속도가 빠르다.

③ NFC 방식은 액세스 포인트(access point) 없이 두 장치 간의 통신이 가능하다.

④ 최대 통신 가능 거리를 가까운 것에서 먼 순서로 나열하면 Bluetooth < Wi-Fi < NFC < LTE 순이다.

정답 3번

출제 포인트

이 문제는 Bluetooth, NFC, Wi-Fi, LTE의 통신 대상, 전송 속도, 접속 방식, 통신 가능 거리를 구분하는 문제이다.

핵심 개념

Bluetooth: 가까운 거리에서 다양한 호환 기기를 연결하는 근거리 무선 통신 기술이다.
NFC: 수 cm 정도의 매우 가까운 거리에서 기기 간 접촉·근접 통신을 지원하는 기술이며, 장치 간 직접 통신이 가능하다.
Wi-Fi: 주로 액세스 포인트를 통해 비교적 넓은 근거리에서 네트워크에 접속하는 기술이다.
LTE: 이동통신 기지국을 이용하여 넓은 지역에서 통신하는 기술이다.
거리의 대표적 비교는 NFC < Bluetooth < Wi-Fi < LTE이다.

선지별 해설

① X — 틀린 표현은 '동일한 유형의 기기 간에만'이다. Bluetooth는 같은 종류의 기기로 제한되지 않으며, Bluetooth 규격과 프로파일이 호환되는 서로 다른 유형의 기기 간에도 통신할 수 있다.
② X — 틀린 표현은 'NFC 방식이 Bluetooth 방식보다 최대 전송 속도가 빠르다'이다. 일반적으로 NFC의 최대 전송 속도는 Bluetooth보다 느리다.
③ O — NFC는 장치 간 통신 모드를 지원하므로 액세스 포인트 없이 가까이 있는 두 장치가 직접 통신할 수 있다.
④ X — NFC의 위치가 잘못되었다. 최대 통신 가능 거리를 가까운 것부터 대표적으로 나열하면 NFC < Bluetooth < Wi-Fi < LTE 순이다.

NoteBase

오답 포인트

③의 '액세스 포인트 없이'라는 표현을 Wi-Fi의 일반적인 접속 방식과 혼동하기 쉽다. NFC는 극히 가까운 거리에서 장치 간 직접 통신이 가능하며, ④에서는 NFC가 Bluetooth와 Wi-Fi보다 앞에 와야 한다.

다시 나오면 이렇게 판단

통신 기술을 비교할 때는 통신 거리, 액세스 포인트·기지국의 필요 여부, 장치 간 직접 통신 가능 여부, 전송 속도를 각각 나누어 판단한다. NFC는 '수 cm의 초근거리·직접 통신', Bluetooth는 '근거리 기기 연결', Wi-Fi는 '액세스 포인트 기반 네트워크', LTE는 '기지국 기반 광역 이동통신'으로 구분한다.

암기 포인트

거리 순서는 NFC < Bluetooth < Wi-Fi < LTE, NFC는 AP 없이 장치 간 직접 통신이 가능하다.

NoteBase

문 3. <보기>의 프로세스 P1, P2, P3을 시간 할당량(time quantum)이 2인 RR(Round-Robin) 알고리즘으로 스케줄링할 때, 평균 응답시간으로 옳은 것은?(단, 응답시간이란 프로세스의 도착 시간부터 처리가 종료될 때까지의 시간을 말한다. 계산 결과값을 소수점 둘째자리에서 반올림한다)

<보기>

프로세스	도착시간	실행시간
P1	0	3
P2	1	4
P3	3	2

① 5.7 ② 6.0 ③ 7.0 ④ 7.3

정답 1번

출제 포인트

시간 할당량이 2인 RR(Round-Robin) 스케줄링의 실행 순서를 구한 뒤, 각 프로세스의 도착 시점부터 종료 시점까지인 응답시간의 평균을 계산하는 문제이다.

조건 정리

주어진 값: P1은 도착시간 0·실행시간 3, P2는 도착시간 1·실행시간 4, P3은 도착시간 3·실행시간 2이며 시간 할당량은 2이다.
구해야 할 값: 각 프로세스의 응답시간 평균이다.
주의할 조건: 이 문제에서 응답시간은 최초 실행까지의 대기시간이 아니라 도착부터 처리가 끝날 때까지의 시간이다. 시간 할당량을 모두 쓴 미완료 프로세스는 준비 큐의 뒤로 이동한다.

사용할 공식

프로세스 i의 응답시간은 다음과 같다.
$T_i = C_i - A_i$
평균 응답시간은 다음과 같다.
$\bar{T} = \frac{\sum T_i}{n}$
여기서 C_i는 종료시간, A_i는 도착시간이다. 문제에서 응답시간을 도착부터 처리 종료까지의 시간으로 정의했으므로 이 공식을 사용한다.

NoteBase

계산 과정

1. RR 실행 순서는 P1(0~2) → P2(2~4) → P1(4~5) → P3(5~7) → P2(7~9)이다.
2. 종료시간은 P1이 5, P2가 9, P3이 7이다.
3. 응답시간은 P1이 $5 - 0 = 5$, P2가 $9 - 1 = 8$, P3이 $7 - 3 = 4$이다.
4. 평균은 다음과 같다.
$\frac{5+8+4}{3} = \frac{17}{3} = 5.666\ldots$
5. 소수점 둘째 자리에서 반올림하면 5.7이다.

선지 판단

① O — 계산 결과를 소수점 둘째 자리에서 반올림하면 5.7이므로 맞다.
② X — 6.0은 계산된 평균과 일치하지 않는다. 틀린 값 6.0 → 올바르게는 5.7이다.
③ X — 7.0은 계산된 평균과 일치하지 않는다. 틀린 값 7.0 → 올바르게는 5.7이다.
④ X — 7.3은 계산된 평균과 일치하지 않는다. 틀린 값 7.3 → 올바르게는 5.7이다.

오답 포인트

이 문항의 함정은 '응답시간'을 최초로 CPU를 배정받을 때까지의 시간으로 해석하거나, 시간 할당량이 끝난 프로세스를 준비 큐의 맨 앞에 두는 데 있다. 또한 17/3을 구한 뒤 문제의 지시대로 소수점 둘째 자리에서 반올림해야 한다.

다시 나오면 이렇게 판단

각 시점의 준비 큐를 갱신하여 간트 차트를 먼저 완성하고, 각 프로세스에 대해 종료시간에서 도착시간을 뺀 뒤 평균과 반올림 자릿수를 확인한다.

암기 포인트

RR은 시간 할당량만큼 실행한 미완료 프로세스를 큐의 뒤로 보내며, 이 문항의 응답시간은 종료시간 − 도착시간이다.

NoteBase

문 4. 다음 C 프로그램의 실행 결과로 옳은 것은?

```
#include <stdio.h>
int main() {
  int a=120, b=45;
  while ( a != b ) {
    if ( a > b ) a = a - b;
    else b = b - a;
  }
  printf("%d", a);
}
```

① 5 ② 15 ③ 20 ④ 25

정답 2번

출제 포인트

이 문제는 두 수가 같아질 때까지 큰 수에서 작은 수를 빼는 반복문의 실행 과정을 추적하여 최종 출력값을 구하는 문제이다. 이 방식은 뺄셈을 이용한 유클리드 알고리즘으로, 마지막에 같아진 값은 두 수의 최대공약수이다.

조건 정리

초깃값은 $a = 120$, $b = 45$이다. $a \neq b$인 동안 반복하며, $a > b$이면 a에서 b를 빼고 그렇지 않으면 b에서 a를 뺀다. 두 값이 같아지면 반복을 끝내고 a를 출력한다.

사용할 공식

핵심 원리는 $\gcd(a, b) = \gcd(a - b, b)$ (단, $a > b$)와 $\gcd(a, b) = \gcd(a, b - a)$ (단, $b > a$)이다. 큰 수에서 작은 수를 빼도 최대공약수는 변하지 않으므로 이 원리를 반복 적용할 수 있다.

계산 과정

1. (120, 45)에서 120 > 45이므로 (75, 45)가 된다.
2. 75 > 45이므로 (30, 45)가 된다.
3. 30 < 45이므로 (30, 15)가 된다.
4. 30 > 15이므로 (15, 15)가 된다.
5. 두 값이 같아져 반복문이 종료되고 printf("%d", a)가 15를 출력한다.

NoteBase

선지 판단

① X — 출력값은 5가 아니라 15이다. 올바르게는 '실행 결과는 15이다'이다.
② O — 반복 종료 시 $a = b = 15$이므로 출력값은 15이다.
③ X — 출력값은 20이 아니라 15이다. 올바르게는 '실행 결과는 15이다'이다.
④ X — 출력값은 25가 아니라 15이다. 올바르게는 '실행 결과는 15이다'이다.

오답 포인트

반복문을 한두 번만 실행하고 중간값을 답으로 고르면 안 된다. 조건은 $a \neq b$이므로 a와 b가 같아질 때까지 모든 대입 결과를 순서대로 추적해야 한다. 또한 else는 $a < b$일 때 실행되어 $b = b - a$가 된다는 점을 놓치기 쉽다.

다시 나오면 이렇게 판단

반복문 문제는 각 반복이 끝난 뒤의 변수 값을 순서쌍으로 기록한다. 이 코드처럼 큰 수에서 작은 수를 계속 빼는 구조에서는 두 수가 같아지는 시점을 찾고, 반복문 밖의 출력문이 어느 변수를 출력하는지 확인한다.

암기 포인트

큰 수에서 작은 수를 반복해서 빼다가 두 수가 같아지면, 그 공통값은 처음 두 수의 최대공약수이다.
gcd(120, 45) = 15이다.

NoteBase

문 5. <보기>와 같이 수행되는 정렬 알고리즘으로 옳은 것은?

<보기>

단 계 0:	6	5	8	9	4	2
단 계 1:	6	5	8	2	4	9
단 계 2:	6	5	4	2	8	9
단 계 3:	2	5	4	6	8	9
단 계 4:	2	4	5	6	8	9
단 계 5:	2	4	5	6	8	9

① 쉘 정렬(shell sort)

② 히프 정렬(heap sort)

③ 버블 정렬(bubble sort)

④ 선택 정렬(selection sort)

정답 4번

출제 포인트

각 단계에서 미정렬 구간의 최댓값을 골라 그 구간의 맨 뒤 원소와 교환하는 선택 정렬의 수행 과정을 판별하는 문제이다.

올바른 순서

1. 단계 0: 6 5 8 9 4 2 — 전체 배열이 미정렬 구간이다.
2. 단계 1: 6 5 8 2 4 9 — 최댓값 9를 마지막 원소 2와 교환해 맨 뒤에 확정한다.
3. 단계 2: 6 5 4 2 8 9 — 앞의 다섯 원소 중 최댓값 8을 4와 교환해 뒤에서 두 번째에 확정한다.
4. 단계 3: 2 5 4 6 8 9 — 앞의 네 원소 중 최댓값 6을 2와 교환한다.
5. 단계 4: 2 4 5 6 8 9 — 앞의 세 원소 중 최댓값 5를 4와 교환한다.
6. 단계 5: 2 4 5 6 8 9 — 남은 두 원소가 이미 순서대로 있어 배열이 그대로 유지된다.

왜 이 순서인가: 한 단계에서 미정렬 구간의 최댓값을 맨 뒤에 놓아 정렬 구간을 오른쪽에서부터 한 칸씩 넓히기 때문이다.

NoteBase

선지별 해설

① X — 셸 정렬은 간격을 정해 떨어진 원소들을 부분 배열로 묶어 삽입 정렬하고, 간격을 줄여 가는 방식이다. 보기에는 간격별 부분 정렬이 아니라 미정렬 구간의 최댓값을 뒤쪽에 확정하는 과정이 나타난다. 틀린 표현: '셸 정렬' → 올바르게는 '선택 정렬'이다.
② X — 히프 정렬은 먼저 히프를 구성한 뒤 루트의 최댓값 또는 최솟값을 꺼내고 히프를 다시 조정한다. 보기에는 히프 구성이나 재조정 과정 없이 최댓값을 찾아 끝 원소와 직접 교환한다. 틀린 표현: '히프 정렬' → 올바르게는 '선택 정렬'이다.
③ X — 버블 정렬은 인접한 두 원소를 차례로 비교·교환한다. 오름차순 첫 회전이라면 인접 교환이 연속되어야 하지만, 보기의 단계 0에서 단계 1은 9와 떨어져 있는 2를 직접 교환한 형태이다. 틀린 표현: '버블 정렬' → 올바르게는 '선택 정렬'이다.
④ O — 매 단계에서 미정렬 구간의 최댓값을 선택해 그 구간의 마지막 원소와 교환하고, 오른쪽의 정렬 완료 구간을 한 칸씩 늘리므로 선택 정렬이다.

오답 포인트

정렬 결과가 오른쪽부터 완성된다는 사실만으로 버블 정렬과 혼동하기 쉽다. 버블 정렬은 인접 원소를 연속해서 교환하지만, 이 보기에서는 9와 2, 8과 4처럼 떨어진 원소가 한 단계에서 교환된다. 이는 최댓값을 먼저 선택한 뒤 끝자리와 바꾸는 선택 정렬의 특징이다.

다시 나오면 이렇게 판단

각 단계에서 확정되는 위치와 교환 범위를 확인한다. 미정렬 구간의 최댓값 또는 최솟값 하나를 선택해 경계 원소와 직접 교환하면 선택 정렬이고, 인접 원소만 연속 교환하면 버블 정렬이다.

암기 포인트

선택 정렬은 '최댓값·최솟값을 선택하고 경계 자리와 교환'한다.

NoteBase

문 6. 직원(사번, 이름, 입사년도, 부서) 테이블에 대한 SQL문 중 문법적으로 옳은 것은?

① SELECT COUNT(부서) FROM 직원 GROUP 부서;

② SELECT * FROM 직원 WHERE 입사년도 IS NULL;

③ SELECT 이름, 입사년도 FROM 직원 WHERE 이름 = '최%';

④ SELECT 이름, 부서 FROM 직원 WHERE 입사년도 = (2014, 2015);

정답 2번

출제 포인트

이 문제는 SQL의 GROUP BY, NULL 비교, 문자열 패턴 검색, 여러 값 비교에 맞는 연산자와 구문을 구별하는 문제이다.

핵심 개념

GROUP BY는 같은 값을 가진 행을 그룹으로 묶는 구문이며 GROUP과 BY를 함께 쓴다. NULL 여부는 =가 아니라 IS NULL 또는 IS NOT NULL로 판단한다. 문자열 패턴은 LIKE와 %, 여러 후보 값은 IN을 사용해 비교한다.

선지별 해설

① X

틀린 표현은 GROUP 부서이다. 그룹화 구문에는 BY가 필요하므로 SELECT COUNT(부서) FROM 직원 GROUP BY 부서;로 고쳐야 한다.

② O

입사년도 값이 NULL인 행은 WHERE 입사년도 IS NULL로 조회한다. NULL 여부를 검사하는 IS NULL을 올바르게 사용한 문장이다.

③ X

틀린 표현은 이름 = '최%'이다. =는 문자열을 그대로 비교하므로 %를 와일드카드로 해석하지 않는다. 성이 '최'인 이름을 찾으려면 SELECT 이름, 입사년도 FROM 직원 WHERE 이름 LIKE '최%';로 고쳐야 한다.

④ X

틀린 표현은 입사년도 = (2014, 2015)이다. =는 하나의 값과 비교하는 연산자이므로 여러 후보 값을 괄호로 나열할 수 없다. SELECT 이름, 부서 FROM 직원 WHERE 입사년도 IN (2014, 2015);로 고쳐야 한다.

NoteBase

오답 포인트

①의 GROUP에서 BY가 빠진 점, ③에서 %를 썼더라도 =가 아니라 LIKE가 필요하다는 점, ④에서 여러 값에는 =가 아니라 IN이 필요하다는 점이 핵심 함정이다.

다시 나오면 이렇게 판단

NULL 확인은 IS NULL, 패턴 검색은 LIKE, 여러 값 중 하나와의 일치는 IN, 그룹화는 GROUP BY라는 짝을 먼저 확인한다.

암기 포인트

NULL은 IS, 패턴은 LIKE, 목록은 IN, 그룹은 GROUP BY이다.

NoteBase

문 7. <보기>와 같은 특성을 갖는 하드 디스크의 최대 저장 용량은?

<보기>

- 실린더(cylinder) 개수: 32,768개
- 면(surface) 개수: 4개
- 트랙(track) 당 섹터(sector) 개수: 256개
- 섹터 크기(sector size): 512 bytes

① 4 GB ② 16 GB ③ 64 GB ④ 1 TB

정답 2번

출제 포인트

하드 디스크의 최대 저장 용량을 실린더 수, 면 수, 트랙당 섹터 수, 섹터 크기의 곱으로 계산하는 문제이다.

조건 정리

주어진 값: 실린더 32,768개, 면 4개, 트랙당 섹터 256개, 섹터 크기 512*bytes*이다.
구해야 할 값: 하드 디스크의 최대 저장 용량이다.
주의할 조건: 실린더 하나에는 각 면마다 트랙이 하나씩 있으므로 전체 트랙 수는 실린더 수와 면 수를 곱한다.

사용할 공식

저장 용량=실린더 수×면 수×트랙당 섹터 수×섹터 크기
각 면의 트랙 수가 실린더 수와 같고, 각 트랙은 같은 수의 섹터로 구성되므로 이 공식을 사용한다.

계산 과정

1. 전체 트랙 수는 $32,768 \times 4 = 131,072$개이다.
2. 전체 섹터 수는 $131,072 \times 256 = 33,554,432$개이다.
3. 최대 저장 용량은 $33,554,432 \times 512 = 17,179,869,184$*bytes*이다.
4. $1GB = 2^{30}bytes$로 환산하면 $17,179,869,184 \div 2^{30} = 16GB$이다.

NoteBase

선지 판단

① X — 계산 결과는 16*GB*이므로 4*GB*는 틀리다. 올바르게는 16*GB*이다.
② O — 계산 결과인 16*GB*와 일치한다.
③ X — 계산 결과는 16*GB*이므로 64*GB*는 틀리다. 올바르게는 16*GB*이다.
④ X — 1*TB*가 아니라 16*GB*이다. 16*GB*는 1*TB*보다 훨씬 작다.
계산 결과가 16*GB*이므로 정답은 2번이다.

오답 포인트

면 수 4개를 트랙 수 계산에서 누락하거나, 반대로 중복하여 곱하면 ① 또는 ③을 고르기 쉽다. 또한 byte를 GB로 환산하지 않고 큰 수만 보고 ④를 선택하면 안 된다. 이 문항에서는 $1GB = 2^{30} bytes$를 적용한다.

다시 나오면 이렇게 판단

실린더 수에 면 수를 곱해 전체 트랙 수를 구하고, 트랙당 섹터 수와 섹터 크기를 차례로 곱한 뒤 마지막에 byte를 GB 또는 TB로 환산한다.

암기 포인트

디스크 용량은 '실린더 × 면 × 트랙당 섹터 × 섹터 크기'이다.

NoteBase

문 8. <보기>의 직원 테이블에서 키(key)와 관련된 설명으로 옳지 않은 것은?(단, 사번과 주민등록번호는 각각 유일한 값을 갖고, 부서번호는 부서 테이블을 참조하는 속성이며, 나이가 같은 동명이인이 존재할 수 있다)

<보기>

직원(사번, 이름, 주민등록번호, 주소, 나이, 성별, 부서번호)

① 부서번호는 외래키이다.
② 사번은 기본키가 될 수 있다.
③ (이름, 나이)는 후보키가 될 수 있다.
④ 주민등록번호는 대체키가 될 수 있다.

정답 3번

출제 포인트

이 문제는 기본키·후보키·대체키·외래키의 정의와 주어진 속성의 유일성 조건을 구분하는 문제이다.

핵심 개념

후보키는 튜플을 유일하게 식별하는 최소 속성 집합이고, 기본키는 후보키 중 대표로 선택한 키이다. 대체키는 기본키로 선택되지 않은 나머지 후보키이며, 외래키는 다른 테이블의 키를 참조하는 속성이다. 이 문제에서는 사번과 주민등록번호가 각각 유일하므로 후보키가 될 수 있지만, 같은 이름과 나이를 가진 동명이인이 존재할 수 있으므로 (이름, 나이)는 유일성을 보장하지 못한다.

선지별 해설

① O — 부서번호는 부서 테이블을 참조하는 속성이므로 외래키이다.
② O — 사번은 각 직원마다 유일하므로 후보키가 될 수 있고, 후보키 중 대표로 선택하면 기본키가 될 수 있다.
③ X — 틀린 표현은 '(이름, 나이)는 후보키가 될 수 있다'이다. 같은 이름과 같은 나이를 가진 동명이인이 존재할 수 있어 (이름, 나이)는 직원을 유일하게 식별하지 못한다. 올바르게는 '(이름, 나이)는 후보키가 될 수 없다'이다.
④ O — 주민등록번호는 각 직원마다 유일하므로 후보키가 될 수 있다. 사번을 기본키로 선택하면 주민등록번호는 선택되지 않은 후보키인 대체키가 될 수 있다.

오답 포인트

③에서 두 속성을 결합했다는 사실만으로 후보키라고 판단하기 쉽다. 복합 속성도 모든 튜플을 유일하게 식별해야 하는데, 문제에서 같은 이름과 나이를 가진 동명이인을 허용하므로 유일성 조건을 충족하지 못한다.

NoteBase

다시 나오면 이렇게 판단

먼저 각 속성 또는 속성 집합이 모든 행을 유일하게 식별하는지 확인한다. 유일성과 최소성을 모두 만족하면 후보키, 후보키 중 선택된 것은 기본키, 선택되지 않은 후보키는 대체키, 다른 테이블을 참조하는 속성은 외래키로 판단한다.

암기 포인트

후보키는 유일성+최소성, 기본키는 선택된 후보키, 대체키는 남은 후보키, 외래키는 다른 테이블 참조이다.

NoteBase

문 9. 소프트웨어 테스트에 대한 설명으로 옳지 않은 것은?

① 베타(beta) 테스트는 고객 사이트에서 사용자에 의해서 수행된다.

② 회귀(regression) 테스트는 한 모듈의 수정이 다른 부분에 미치는 영향을 검사한다.

③ 화이트 박스(white box) 테스트는 모듈의 내부 구현보다는 입력과 출력에 의해 기능을 검사한다.

④ 스트레스(stress) 테스트는 비정상적으로 과도한 분량 또는 빈도로 자원을 요청할 때의 영향을 검사한다.

정답 3번

출제 포인트

소프트웨어 테스트의 종류별 수행 환경과 검사 대상을 구분하는 문제이다. 특히 화이트박스 테스트와 블랙박스 테스트의 차이가 핵심이다.

핵심 개념

베타 테스트: 정식 출시 전에 실제 사용자에게 사용하게 하여 결함과 사용성을 확인하는 인수 테스트이다.
회귀 테스트: 소프트웨어를 변경한 뒤 기존 기능이나 다른 부분에 부작용이 생기지 않았는지 다시 검사하는 테스트이다.
화이트박스 테스트: 프로그램의 내부 구조와 구현을 바탕으로 논리 경로, 조건, 반복문 등을 검사하는 구조적 테스트이다.
블랙박스 테스트: 내부 구현을 보지 않고 입력에 따른 출력과 기능이 요구사항에 맞는지 검사하는 기능적 테스트이다.
스트레스 테스트: 정상 범위를 넘는 과도한 부하나 자원 부족 상황에서 시스템의 한계와 안정성을 검사하는 테스트이다.

NoteBase

선지별 해설

① O — 베타 테스트는 개발 조직 밖의 실제 사용자 또는 고객이 실제 사용 환경에서 수행하여 출시 전 문제를 찾는 테스트이므로 옳다.
② O — 회귀 테스트는 수정이나 기능 추가가 이미 정상적으로 동작하던 다른 부분에 부작용을 일으켰는지 확인하므로 옳다.
③ X — 틀린 표현은 '모듈 내부 구현보다는 입력과 출력에 의해 기능을 검사한다'이다. 이는 블랙박스 테스트의 설명이다. 올바르게는 '화이트박스 테스트는 모듈의 내부 구현과 구조를 바탕으로 논리 경로와 제어 구조 등을 검사한다'이다.
④ O — 스트레스 테스트는 비정상적으로 과도한 작업량, 높은 요청 빈도, 제한된 자원 등 극한 조건을 가해 시스템의 한계와 장애 대응을 확인하므로 옳다.

출제자가 노린 함정

③에서 '화이트박스'라는 용어에 블랙박스의 검사 방식인 '내부 구현을 보지 않고 입력과 출력으로 기능을 검사한다'를 붙였다. 테스트 명칭보다 무엇을 보고 검사하는지를 기준으로 판단해야 한다.

다시 나오면 이렇게 판단

내부 코드·논리 경로·제어 구조를 보면 화이트박스, 외부에서 입력과 출력·요구 기능을 보면 블랙박스로 판단한다. 변경의 부작용은 회귀, 실제 사용자의 출시 전 시험은 베타, 한계를 넘는 부하는 스트레스로 구분한다.

암기 포인트

화이트박스는 내부 구조, 블랙박스는 입력·출력, 회귀는 변경 후 재검사, 베타는 실제 사용자, 스트레스는 과부하이다.

NoteBase

문 10. <보기>에 선언된 배열 A의 원소 A[8][7]의 주소를 행 우선(row-major) 순서와 열 우선(column-major) 순서로 각각 바르게 계산한 것은?(단, 첫 번째 원소 A[0][0]의 주소는 1,000이고 하나의 원소는 1 byte를 차지한다)

<보기>

```
char A[20][30];
```

	행 우선 주소	열 우선 주소
①	1,167	1,148
②	1,167	1,218
③	1,247	1,148
④	1,247	1,218

정답 3번

출제 포인트

이 문제는 2차원 배열에서 행 우선 순서와 열 우선 순서의 원소 위치 공식을 이용해 A[8][7]의 주소를 각각 계산하는 문제이다.

조건 정리

배열은 char A[20][30]이므로 행의 수는 20, 열의 수는 30이다.
첫 원소 A[0][0]의 주소는 1,000이다.
한 원소의 크기는 1*byte*이므로 원소 오프셋과 주소 오프셋의 수치가 같다.
구해야 할 값은 A[8][7]의 행 우선 주소와 열 우선 주소이다.

사용할 공식

행 우선 주소는 다음과 같다.
주소=시작 주소+(행 인덱스×열의 수+열 인덱스)×원소 크기
열 우선 주소는 다음과 같다.
주소=시작 주소+(열 인덱스×행의 수+행 인덱스)×원소 크기
행 우선은 한 행의 30개 원소를 먼저 저장하므로 행 인덱스에 열의 수 30을 곱한다. 열 우선은 한 열의 20개 원소를 먼저 저장하므로 열 인덱스에 행의 수 20을 곱한다.

NoteBase

계산 과정

1. 행 우선 오프셋은 8 × 30 + 7 = 247이다. 따라서 주소는 1000 + 247 × 1 = 1247이다.
2. 열 우선 오프셋은 7 × 20 + 8 = 148이다. 따라서 주소는 1000 + 148 × 1 = 1148이다.
3. 두 결과를 순서대로 쓰면 행 우선 주소 1,247, 열 우선 주소 1,148이다.

선지 판단

① X — 열 우선 주소 1,148은 맞지만 행 우선 주소 1,167은 틀리다. 1,167은 행 우선 계산에서 열의 수 30 대신 행의 수 20을 넣어 1000 + (8 × 20 + 7)로 계산한 값이다. 올바른 조합은 행 우선 1,247, 열 우선 1,148이다.
② X — 행 우선 주소 1,167과 열 우선 주소 1,218이 모두 틀리다. 행 우선에는 열의 수 30을, 열 우선에는 행의 수 20을 사용해야 한다. 올바른 조합은 행 우선 1,247, 열 우선 1,148이다.
③ O — 행 우선 주소는 1000 + (8 × 30 + 7) = 1247, 열 우선 주소는 1000 + (7 × 20 + 8) = 1148 이므로 두 값이 모두 맞다.
④ X — 행 우선 주소 1,247은 맞지만 열 우선 주소 1,218은 틀리다. 1,218은 열 우선 계산에서 행의 수 20 대신 열의 수 30을 넣어 1000 + (7 × 30 + 8)로 계산한 값이다. 올바른 열 우선 주소는 1,148이다.
계산 결과가 행 우선 1,247, 열 우선 1,148이므로 정답은 3번이다.

오답 포인트

핵심 함정은 행 우선과 열 우선에서 곱하는 차원의 크기를 뒤바꾸는 것이다. 행 우선에서는 행 인덱스 8에 열의 수 30을 곱하고, 열 우선에서는 열 인덱스 7에 행의 수 20을 곱한다. 인덱스가 0부터 시작하므로 8이나 7에 1을 더하지 않는다.

다시 나오면 이렇게 판단

행 우선은 i×열의 수+j, 열 우선은 j×행의 수+i로 원소 오프셋을 구한 뒤 원소 크기를 곱하고 시작 주소를 더한다. 마지막에는 배열 인덱스가 0부터 시작하는지와 원소 크기를 함께 확인한다.

암기 포인트

행 우선은 행 인덱스에 열의 수를 곱하고, 열 우선은 열 인덱스에 행의 수를 곱한다.

NoteBase

문 11. 인터넷 주소 체계인 IPv4와 IPv6의 주소 길이와 주소 표시 방법을 각각 바르게 나열한 것은?

	IPv4	IPv6
①	(32비트, 8비트씩 4부분)	(128비트, 16비트씩 8부분)
②	(32비트, 8비트씩 4부분)	(128비트, 8비트씩 16부분)
③	(64비트, 16비트씩 4부분)	(256비트, 32비트씩 8부분)
④	(64비트, 16비트씩 4부분)	(256비트, 16비트씩 16부분)

정답 1번

출제 포인트

IPv4와 IPv6의 전체 주소 길이와 주소를 나누어 표시하는 단위를 구별하는 문제이다.

핵심 개념

IPv4: 전체 길이는 32비트이며, 8비트씩 4부분으로 나누어 점(.)으로 구분한 10진수로 표시한다.
IPv6: 전체 길이는 128비트이며, 16비트씩 8부분으로 나누어 콜론(:)으로 구분한 16진수로 표시한다.
두 체계는 '전체 비트 수 ÷ 한 부분의 비트 수 = 부분 수'의 관계로 구별할 수 있다.

선지별 해설

① O — IPv4는 32비트를 8비트씩 4부분으로, IPv6는 128비트를 16비트씩 8부분으로 나누므로 모두 옳다.
② X — IPv6의 '8비트씩 16부분'이 틀렸다. 올바르게는 '128비트, 16비트씩 8부분'이다.
③ X — IPv4의 '64비트, 16비트씩 4부분'과 IPv6의 '256비트, 32비트씩 8부분'이 모두 틀렸다. 올바르게는 IPv4가 '32비트, 8비트씩 4부분', IPv6가 '128비트, 16비트씩 8부분'이다.
④ X — IPv4의 '64비트, 16비트씩 4부분'과 IPv6의 '256비트, 16비트씩 16부분'이 모두 틀렸다. 올바르게는 IPv4가 '32비트, 8비트씩 4부분', IPv6가 '128비트, 16비트씩 8부분'이다.

오답 포인트

②는 IPv6의 전체 길이 128비트를 유지하면서 한 부분의 크기와 부분 수를 서로 바꾼 함정이다. 또한 ③·④는 IPv4와 IPv6의 전체 주소 길이부터 각각 64비트와 256비트로 부풀려 제시한다.

다시 나오면 이렇게 판단

먼저 전체 길이를 IPv4 32비트, IPv6 128비트로 확인한 뒤, 각 부분의 크기로 나누어 부분 수를 검산한다. IPv4는 32÷8=4, IPv6는 128÷16=8이 된다.

NoteBase

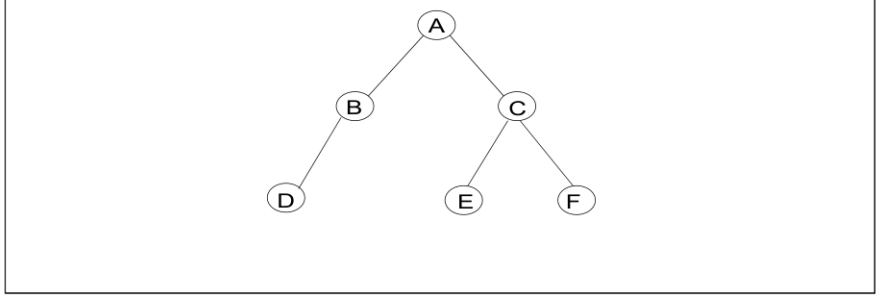
암기 포인트

IPv4는 32-8-4, IPv6는 128-16-8이다.

NoteBase

문 12. <보기>의 이진 트리에 대해 지정된 방법으로 순회한 결과가 옳지 않은 것은?

<보기>



① 중위순회: D → B → A → E → C → F
② 레벨순회: A → B → C → D → E → F
③ 전위순회: A → B → D → C → E → F
④ 후위순회: D → B → A → E → F → C

정답 4번

출제 포인트

이 문제는 이진 트리에서 중위순회, 레벨순회, 전위순회, 후위순회의 방문 순서를 구분하는 문제이다.

올바른 순서

중위순회(왼쪽-루트-오른쪽): D → B → A → E → C → F
레벨순회(위에서 아래로, 같은 레벨은 왼쪽에서 오른쪽으로): A → B → C → D → E → F
전위순회(루트-왼쪽-오른쪽): A → B → D → C → E → F
후위순회(왼쪽-오른쪽-루트): D → B → E → F → C → A
후위순회에서는 각 부분 트리의 자식 노드를 부모보다 먼저 방문하므로 루트 A가 전체 순회의 마지막이 된다.

NoteBase

선지별 해설

① O — 중위순회는 왼쪽 부분 트리, 루트, 오른쪽 부분 트리 순서이다. 따라서 D → B 다음에 A를 방문하고, 오른쪽 부분 트리에서 E → C → F를 방문하므로 제시된 순서가 맞다.
② O — 레벨순회는 깊이가 얕은 노드부터 같은 레벨의 왼쪽 노드와 오른쪽 노드 순으로 방문한다. 따라서 A → B → C → D → E → F가 맞다.
③ O — 전위순회는 루트를 먼저 방문한 뒤 왼쪽 부분 트리와 오른쪽 부분 트리를 차례로 방문한다. 따라서 A → B → D → C → E → F가 맞다.
④ X — 후위순회인데 루트 A를 세 번째에 둔 부분이 틀렸다. 후위순회는 왼쪽 부분 트리 D → B, 오른쪽 부분 트리 E → F → C를 모두 방문한 뒤 루트 A를 마지막에 방문한다. 올바르게는 D → B → E → F → C → A이다.

오답 포인트

④는 왼쪽 부분 트리의 후위순회 D → B 뒤에 곧바로 루트 A를 넣어 중위순회와 혼동하게 만든다.
후위순회에서는 오른쪽 부분 트리까지 모두 처리한 뒤 전체 루트를 마지막에 방문한다.

다시 나오면 이렇게 판단

순회 이름을 보자마자 루트의 위치를 먼저 확인한다. 전위는 루트가 앞, 중위는 루트가 왼쪽과 오른쪽 부분 트리 사이, 후위는 루트가 뒤이며, 레벨순회는 깊이 순으로 판단한다. 각 부분 트리에도 같은 규칙을 재귀적으로 적용한다.

암기 포인트

전위는 루트 먼저, 중위는 루트 가운데, 후위는 루트 마지막, 레벨은 위에서 아래이다.

NoteBase

문 13. 컴퓨터 시스템의 인터럽트(interrupt)에 대한 설명으로 옳지 않은 것은?

① 인터럽트는 입출력 연산, 하드웨어 실패, 프로그램 오류 등에 의해서 발생한다.

② 인터럽트 처리 우선순위 결정 방식에는 폴링(polling) 방식과 데이지 체인(daisy-chain) 방식이 있다.

③ 인터럽트가 추가된 명령어 사이클은 인출 사이클, 인터럽트 사이클, 실행 사이클의 순서로 수행된다.

④ 인터럽트가 발생할 경우, 진행 중인 프로그램의 재개(resume)에 필요한 레지스터 문맥(register context)을 저장한다.

정답 3번

출제 포인트

인터럽트의 발생 원인과 우선순위 결정 방식, 인터럽트가 포함된 명령어 사이클의 순서, 프로그램 재개를 위한 문맥 저장을 구별하는 문제이다.

핵심 개념

인터럽트는 입출력 완료, 하드웨어 이상, 프로그램 오류 등과 같이 즉시 처리가 필요한 사건이 발생했음을 CPU에 알리는 신호이다. CPU는 현재 명령어의 실행을 마친 뒤 프로그램의 상태를 나타내는 레지스터 문맥을 저장하고 인터럽트 서비스 루틴을 수행한다. 인터럽트가 요청된 명령어 사이클의 기본 흐름은 인출 사이클 → 실행 사이클 → 인터럽트 사이클이며, 인터럽트 처리가 끝나면 저장한 문맥을 복원하여 중단된 프로그램을 재개한다.

선지별 해설

① O — 입출력 동작의 완료나 이상, 하드웨어 실패, 프로그램 실행 중 오류 등은 대표적인 인터럽트 발생 원인이다.

② O — 여러 인터럽트 요청의 우선순위를 판별하는 방식에는 소프트웨어적으로 장치를 차례로 확인하는 폴링 방식과 하드웨어적으로 우선순위를 연결하는 데이지 체인 방식이 있다.

③ X — 틀린 부분은 '인출 사이클, 인터럽트 사이클, 실행 사이클의 순서'이다. 올바르게는 인출 사이클 → 실행 사이클 → 인터럽트 사이클의 순서로 수행된다. 인터럽트는 현재 명령어의 실행을 완료한 뒤 검사·처리한다.

④ O — 인터럽트 처리 후 중단된 프로그램을 정확히 재개하려면 프로그램 카운터와 상태 정보 등 당시의 레지스터 문맥을 저장해야 한다.

NoteBase

오답 포인트

③은 인터럽트 요청이 발생하면 실행 중인 명령어를 곧바로 중간에서 끊는다고 생각하게 만드는 함정이다. 일반적인 명령어 사이클에서는 현재 명령어의 실행을 끝낸 뒤 인터럽트 사이클로 진입하므로 실행 사이클과 인터럽트 사이클의 순서를 바꾸면 안 된다.

다시 나오면 이렇게 판단

인터럽트 사이클의 위치를 물으면 '현재 명령어 실행 완료 → 인터럽트 여부 확인 및 처리 → 다음 명령어 인출'의 흐름으로 판단한다. 우선순위 방식은 폴링과 데이지 체인, 재개 조건은 레지스터 문맥의 저장과 복원으로 연결한다.

암기 포인트

명령어는 인출 후 실행하고, 인터럽트는 실행을 마친 뒤 처리하며, 문맥을 복원해 프로그램을 재개한다.

NoteBase

문 14. 다음은 3년간 연이율 4%로 매월 적립하는 월 복리 정기적금의 만기지급금을 계산한 결과이다. 셀 C2에 들어갈 수식으로 옳은 것은?(단, 만기지급금의 10원 단위 미만은 절사한다)

	A	B	C
1	성명	월적립액	만기지급금
2	김**	₩30,000	₩1,145,440
3	이**	₩50,000	₩1,909,070

① =ROUNDDOWN(FV(4%, 3*12, -B2), -1)

② =ROUNDDOWN(FV(4%, 3*12, -B2), -2)

③ =ROUNDDOWN(FV(4%/12, 3*12, -B2), -1)

④ =ROUNDDOWN(FV(4%/12, 3*12, -B2), -2)

정답 3번

출제 포인트

연이율을 월이율로 환산하고, 월 적립 횟수와 10원 단위 미만 절사 조건을 FV 함수와 ROUNDDOWN 함수에 정확히 반영하는 문제이다.

조건 정리

주어진 값: 연이율 4%, 적립 기간 3년, 월 적립액 B2
구해야 할 값: 매월 말 적립하는 월 복리 정기적금의 만기지급금
주의할 조건: 월이율은 4%/12, 납입 횟수는 3 × 12 = 36회이다. 만기지급금의 10원 단위 미만, 즉 1원 자리를 버리므로 ROUNDDOWN의 자릿수는 -1이다.

사용할 공식

$C2 = \text{ROUNDDOWN}(\text{FV}(4\%/12, 3 \times 12, -B2), -1)$
FV 함수는 일정한 이율로 같은 금액을 매 기간 적립할 때의 미래가치를 구한다. 이 문제는 매월 적립하므로 기간당 이율과 기간 수를 각각 월 단위로 맞춘다. FV 함수의 현금 흐름 부호 규칙에 따라 납입액 B2 앞에 음수 부호를 붙이면 만기 수령액이 양수로 계산된다.

계산 과정

1. 월이율을 4%/12로 환산한다.
2. 3년간 매월 적립하므로 납입 횟수를 3 × 12 = 36회로 둔다.
3. B2가 30,000원일 때 FV(4%/12, 36, −30000)의 결과는 약 1,145,446.87원이다.
4. ROUNDDOWN의 자릿수 -1을 적용하여 10원 미만을 절사하면 1,145,440원이 되어 표의 C2와 일치한다.

NoteBase

선지 판단

① X - FV의 이율 인수에 연이율 4%를 그대로 넣어 이를 월이율로 잘못 적용한다. B2가 30,000원일 때 2,327,940원이 되어 C2와 일치하지 않는다. 틀린 표현 'FV(4%, 3*12, -B2)'는 'FV(4%/12, 3*12, -B2)'로 고쳐야 한다.
② X - 연이율 4%를 월이율로 환산하지 않았고, 자릿수 -2는 100원 미만을 절사한다. B2가 30,000원일 때 2,327,900원이 되어 C2와 일치하지 않는다. 틀린 표현 'FV(4%, 3*12, -B2), -2'는 'FV(4%/12, 3*12, -B2), -1'로 고쳐야 한다.
③ O - 월이율 4%/12, 36회의 적립, 납입액 -B2, 10원 미만 절사를 뜻하는 자릿수 -1을 모두 정확히 반영한다. 계산 결과는 C2의 1,145,440원과 일치한다.
④ X - FV의 이율과 기간은 맞지만 ROUNDDOWN의 자릿수 -2는 100원 미만을 절사하여 1,145,400원이 된다. 틀린 표현 '-2'는 10원 미만을 절사하는 '-1'로 고쳐야 한다.

오답 포인트

연이율 4%를 월 적립의 기간당 이율로 그대로 넣는 것과 '10원 단위 미만 절사'를 -2로 해석하는 것이 함정이다. 월 단위 계산에서는 연이율을 12로 나누며, ROUNDDOWN에서 -1은 1원 자리 절사, -2는 10원 자리까지 절사한다.

다시 나오면 이렇게 판단

적립 주기에 맞춰 이율과 기간 수의 단위를 먼저 통일한 뒤 FV의 현금 흐름 부호를 확인하고, 절사할 자리 바로 왼쪽 자릿수를 남기도록 ROUNDDOWN의 자릿수를 정한다.

암기 포인트

월 적립은 연이율÷12, 기간은 연수×12, 10원 미만 절사는 ROUNDDOWN의 -1이다.

NoteBase

문 15. 정점의 개수가 n인 연결그래프로부터 생성 가능한 신장트리(spanning tree)의 간선의 개수는?

① $n-1$ ② n ③ $\frac{n(n-1)}{2}$ ④ n^2

정답 1번

출제 포인트

정점이 *n*개인 신장트리가 갖는 간선의 개수를 묻는 문제이다.

조건 정리

주어진 조건: 정점이 *n*개인 연결그래프이다.
구해야 할 값: 이 그래프의 신장트리가 갖는 간선의 개수이다.
주의할 조건: 신장트리는 원래 그래프의 모든 정점을 포함하면서 연결되어 있고 사이클이 없는 부분 그래프이다.

사용할 공식

정점이 *n*개인 트리의 간선 수는 다음과 같다.
$|E| = n - 1$
신장트리도 트리이므로 이 공식을 그대로 적용한다.

계산 과정

1. 정점 하나에서 시작하면 간선은 0개이다.
2. 연결 상태를 유지하면서 사이클을 만들지 않으려면 정점을 하나 추가할 때마다 간선을 정확히 하나씩 추가해야 한다.
3. 나머지 $n - 1$개의 정점을 연결하므로 간선의 개수는 $n - 1$개이다.

선지 판단

① O — $n - 1$은 정점이 *n*개인 모든 트리의 간선 수이므로 옳다.
② X — *n*은 간선이 하나 더 많은 값이다. 정점이 *n*개인 연결그래프에 간선이 *n*개이면 사이클이 생기므로, 올바른 표현은 $n - 1$이다.
③ X — $\frac{n(n-1)}{2}$는 정점이 *n*개인 단순 무방향 완전그래프가 가질 수 있는 간선 수이다. 신장트리의 간선 수로는 틀리며, 올바른 표현은 $n - 1$이다.
④ X — n^2은 $n \times n$ 인접행렬의 전체 원소 수에 해당할 수 있을 뿐 신장트리의 간선 수가 아니다. 올바른 표현은 $n - 1$이다.
따라서 정답은 1번이다.

NoteBase

오답 포인트

‘생성 가능한 신장트리’라는 표현을 ‘신장트리의 가능한 개수’로 잘못 읽기 쉽다. 이 문항은 신장트리가 몇 종류인지가 아니라 하나의 신장트리를 구성하는 간선이 몇 개인지를 묻는다. 또한 ③은 완전그래프의 간선 수 공식과 혼동하도록 만든 선지이다.

다시 나오면 이렇게 판단

신장트리의 간선 수를 물으면 원래 연결그래프의 간선 수나 모양과 관계없이 정점 수에서 1을 뺀 $n-1$을 적용한다.

암기 포인트

트리와 신장트리: 정점 n개이면 간선은 항상 $n-1$개이다.

NoteBase

문 16. <보기>는 관계형 데이터베이스의 정규화 작업을 설명한 것이다. 제1정규형, 제2정규형, 제3정규형, BCNF를 생성하는 정규화 작업을 순서대로 나열한 것은?

<보기>

ㄱ. 결정자가 후보키가 아닌 함수 종속성을 제거한다.
ㄴ. 부분 함수 종속성을 제거한다.
ㄷ. 속성을 원자값만 갖도록 분해한다.
ㄹ. 이행적 함수 종속성을 제거한다.

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

정답 4번

출제 포인트

이 문제는 관계형 데이터베이스의 정규화 단계별 작업을 제1정규형, 제2정규형, 제3정규형, BCNF의 순서에 맞게 배열하는 문제이다.

올바른 순서

1. ㄷ — 속성의 값을 원자값만 갖도록 분해하여 제1정규형을 만든다.
2. ㄴ — 기본키의 일부에만 종속되는 부분 함수 종속성을 제거하여 제2정규형을 만든다.
3. ㄹ — 기본키가 아닌 속성을 거쳐 종속되는 이행적 함수 종속성을 제거하여 제3정규형을 만든다.
4. ㄱ — 모든 결정자가 후보키가 되도록, 결정자가 후보키가 아닌 함수 종속성을 제거하여 BCNF를 만든다.

왜 이 순서: 각 단계는 바로 앞 정규형의 조건을 충족한 관계를 대상으로 더 강한 함수 종속성 제약을 적용하므로 ㄷ → ㄴ → ㄹ → ㄱ의 순서가 된다.

NoteBase

선지별 해설

① X — ㄱ을 첫 단계에 둔 것이 틀렸다. ㄱ은 BCNF를 만드는 마지막 작업이다. 올바르게는 ㄷ → ㄴ → ㄹ → ㄱ이다.
② X — ㄱ을 첫 단계에 두고 ㄴ을 마지막에 둔 것이 틀렸다. 원자값 분해인 ㄷ부터 시작하고, 부분 함수 종속성 제거인 ㄴ은 그다음에 수행한다. 올바르게는 ㄷ → ㄴ → ㄹ → ㄱ이다.
③ X — ㄷ 다음에 ㄱ을 둔 것이 틀렸다. 제1정규형을 만든 뒤에는 부분 함수 종속성, 이행적 함수 종속성을 차례로 제거한 후 BCNF 조건을 적용한다. 올바르게는 ㄷ → ㄴ → ㄹ → ㄱ이다.
④ O — ㄷ은 제1정규형, ㄴ은 제2정규형, ㄹ은 제3정규형, ㄱ은 BCNF를 만드는 작업이므로 ㄷ → ㄴ → ㄹ → ㄱ이 정확하다.

오답 포인트

ㄱ의 '결정자가 후보키가 아닌 함수 종속성 제거'를 기본적인 함수 종속성 정리로 보고 앞에 놓기 쉽다. 그러나 이는 제3정규형보다 더 엄격한 BCNF 조건이므로 마지막 단계이다. 또한 부분 함수 종속성은 제2정규형, 이행적 함수 종속성은 제3정규형에서 제거한다.

다시 나오면 이렇게 판단

정규화 순서 문제는 원자값 확인 → 부분 함수 종속성 제거 → 이행적 함수 종속성 제거 → 모든 결정자가 후보키인지 확인의 흐름으로 판단한다.

암기 포인트

원자값(1NF) → 부분(2NF) → 이행(3NF) → 결정자는 후보키(BCNF)이다.

NoteBase

문 17. 다음 Java 프로그램의 실행 결과로 옳은 것은?

```
class Division {
  public static void main(String[] args) {
    int a, b, result;
    a = 3;
    b = 0;
    try {
      result  = a / b;
      System.out.print("A");
    }
    catch (ArithmeticException e) {
      System.out.print("B");
    }
    finally {
      System.out.print("C");
    }
    System.out.print("D");
  }
}
```

① ACD ② BCD ③ ABCD ④ BACD

정답 2번

출제 포인트

이 문제는 정수의 0 나눗셈에서 발생하는 ArithmeticException과 try-catch-finally문의 실행 흐름을 묻는 문제이다.

NoteBase

올바른 순서

1. result = a / b 실행 중 3을 0으로 나누어 ArithmeticException이 발생한다.
2. 예외가 발생한 지점부터 try 블록의 나머지 문장은 건너뛰므로 A는 출력되지 않는다.
3. ArithmeticException을 처리하는 catch 블록이 실행되어 B가 출력된다.
4. finally 블록이 실행되어 C가 출력된다.
5. 예외 처리가 끝난 뒤 다음 문장이 실행되어 D가 출력된다.

따라서 최종 출력은 BCD이다. catch가 해당 예외를 처리한 뒤에도 finally가 실행되고, 그 다음 문장으로 흐름이 이어지기 때문이다.

선지별 해설

① X — ACD는 A가 출력된다고 보았고 B를 빠뜨린 점이 틀리다. 나눗셈 문장에서 예외가 발생하므로 그 다음의 A 출력문은 실행되지 않으며, catch 블록의 B가 출력된다. 올바르게는 BCD이다.

② O — 예외 발생으로 A는 건너뛰고 catch에서 B, finally에서 C, try-catch-finally 다음 문장에서 D가 차례로 출력되므로 BCD가 맞다.

③ X — ABCD는 A까지 출력된다고 본 점이 틀리다. A 출력문에 도달하기 전에 나눗셈 예외가 발생한다. 올바르게는 BCD이다.

④ X — BACD는 B 다음에 A가 출력된다고 본 점이 틀리다. 예외가 발생하면 try 블록의 남은 부분으로 되돌아가지 않으므로 A는 출력되지 않는다. 올바르게는 BCD이다.

오답 포인트

핵심 함정은 finally가 실행된다는 사실만 보고 A도 함께 출력된다고 착각하는 것이다. 예외가 발생한 순간 try 블록의 이후 문장인 print("A")는 건너뛰지만, 일치하는 catch와 finally는 실행되고 예외가 처리되었으므로 이후의 print("D")도 실행된다.

다시 나오면 이렇게 판단

예외 처리 흐름은 예외 발생 지점 확인 → try 블록의 남은 문장 생략 → 일치하는 catch 실행 → finally 실행 → 예외가 처리되었으면 다음 문장 실행의 순서로 판단한다.

암기 포인트

0으로 정수 나눗셈: A 생략, catch의 B → finally의 C → 다음 문장의 D이다.

NoteBase

문 18. <보기>는 공개키 암호 방식을 전자 서명(digital signature)에 적용하여 A가 B에게 메시지를 전송하는 과정에 대한 설명이다. ㉠, ㉡에 들어갈 내용으로 옳은 것은?

<보기>

(1) A와 B는 개인키와 공개키 쌍을 각각 생성한다.
(2) A는 (㉠)를 사용하여 암호화한 메시지를 B에게 전송한다.
(3) B는 (㉡)를 사용하여 수신된 메시지를 해독한다.

	㉠	㉡
①	A의 개인키	A의 공개키
②	A의 개인키	B의 공개키
③	A의 공개키	B의 개인키
④	B의 공개키	B의 개인키

정답 1번

출제 포인트

이 문제는 공개키 암호 방식을 전자 서명에 적용할 때 서명 생성과 서명 확인에 누구의 어떤 키를 사용하는지 묻는 문제이다.

핵심 개념

전자 서명은 송신자 A가 A의 개인키로 서명하고, 수신자 B가 A의 공개키로 그 서명을 확인하는 방식이다. A의 개인키는 A만 알고 있으므로 서명자 확인과 위조 방지에 쓰이며, A의 공개키는 누구나 이용할 수 있으므로 A가 만든 서명을 검증하는 데 쓰인다.
기밀성을 위한 암호화는 수신자 B의 공개키로 암호화하고 B의 개인키로 복호화하지만, 전자 서명은 송신자 A의 개인키와 A의 공개키를 사용한다.

NoteBase

선지별 해설

① O — A가 A의 개인키로 서명에 해당하는 암호화를 하고, B가 A의 공개키로 이를 확인하므로 옳다. 따라서 ㉠은 A의 개인키, ㉡은 A의 공개키이다.
② X — ㉡의 'B의 공개키'가 틀렸다. A의 개인키로 만든 전자 서명은 A의 공개키로 확인하므로, 올바르게는 '㉠ A의 개인키, ㉡ A의 공개키'이다.
③ X — '㉠ A의 공개키, ㉡ B의 개인키'가 모두 틀렸다. 전자 서명에서는 A가 A의 개인키를 사용하고 B는 A의 공개키를 사용하므로, 올바르게는 '㉠ A의 개인키, ㉡ A의 공개키'이다.
④ X — '㉠ B의 공개키, ㉡ B의 개인키'는 B에게만 내용을 공개하려는 기밀성 암호화의 키 조합이다. 전자 서명에 맞게 고치면 '㉠ A의 개인키, ㉡ A의 공개키'이다.

오답 포인트

전자 서명과 기밀성 암호화를 혼동하기 쉽다. ④의 'B의 공개키 → B의 개인키'는 수신자 B만 메시지를 읽게 하는 조합이고, 이 문제의 전자 서명은 서명자 A를 증명해야 하므로 'A의 개인키 → A의 공개키' 조합을 사용한다.

다시 나오면 이렇게 판단

목적이 전자 서명이면 송신자 기준으로 '송신자의 개인키로 서명, 송신자의 공개키로 확인'이라고 판단한다.
목적이 기밀성이면 수신자 기준으로 '수신자의 공개키로 암호화, 수신자의 개인키로 복호화'라고 판단한다.

암기 포인트

전자 서명은 보내는 사람의 개인키로 만들고, 보내는 사람의 공개키로 확인한다.

NoteBase

문 19. 프로그래밍 언어에 대한 설명으로 옳지 않은 것은?

① Objective-C, Java, C#은 객체지향 언어이다.

② Python은 정적 타이핑을 지원하는 컴파일러 방식의 언어이다.

③ ASP, JSP, PHP는 서버 측에서 실행되는 스크립트 언어이다.

④ XML은 전자문서를 표현하는 확장가능한 표준 마크업 언어이다.

정답 2번

출제 포인트

프로그래밍 언어와 문서 표현 언어의 대표적 특징을 구분하는 문제이다. 특히 정적 타이핑과 동적 타이핑의 차이를 판단해야 한다.

핵심 개념

객체지향 언어는 객체를 중심으로 프로그램을 구성하며 Objective-C, Java, C#이 대표적이다.
정적 타이핑은 변수의 자료형이 컴파일 시점에 결정되는 방식이고, 동적 타이핑은 실행 중 값에 따라 자료형이 결정되는 방식이다. Python은 일반적으로 동적 타이핑을 사용하는 인터프리터 방식의 언어로 분류된다.
서버 측 스크립트 언어는 서버에서 실행되어 처리 결과를 클라이언트에 전달하며 ASP, JSP, PHP가 이에 해당한다.
XML은 데이터를 구조화하여 전자문서를 표현하고 교환하는 데 사용하는 확장 가능한 표준 마크업 언어이다.

선지별 해설

① O — Objective-C, Java, C#은 객체와 클래스를 중심으로 프로그램을 구성하는 객체지향 언어이다.
② X — 틀린 표현은 '정적 타이핑을 지원하는 컴파일러 방식'이다. Python은 일반적으로 동적 타이핑을 사용하는 인터프리터 방식의 언어이다.
③ O — ASP, JSP, PHP는 서버에서 실행되어 동적인 웹 콘텐츠를 생성하는 서버 측 스크립트 기술 또는 언어이다.
④ O — XML은 사용자가 목적에 맞는 태그를 정의할 수 있는 확장 가능한 표준 마크업 언어로서 전자문서와 데이터를 구조적으로 표현한다.

오답 포인트

②는 Python도 자료형을 다룬다는 사실을 '정적 타이핑'으로 잘못 연결하게 만든다. 핵심은 자료형의 존재 여부가 아니라 자료형이 결정되는 시점이며, Python은 실행 중에 자료형이 결정되는 동적 타이핑 언어이다.

NoteBase

다시 나오면 이렇게 판단

언어의 특징을 묻는 선지는 '프로그래밍 패러다임', '자료형 결정 시점', '실행 위치', '문서 표현 목적'을 각각 분리해 판단한다. Python과 정적 타이핑·컴파일러 방식이 직접 연결되면 우선 틀린 설명인지 확인한다.

암기 포인트

Python은 동적 타이핑·인터프리터, ASP·JSP·PHP는 서버 측, XML은 확장 가능한 마크업 언어이다.

NoteBase

문 20. <보기>의 설명에 해당하는 기술로 가장 적절한 것은?

<보기>

- 서비스 모델은 IaaS, PaaS, SaaS로 구분한다.
- 필요한 만큼 자원을 임대하여 사용할 수 있다.
- 가상화 기술, 서비스 프로비저닝(provisioning) 기술, 과금 체계 등을 필요로 한다.

① 빅데이터(bigdata)

② 딥 러닝(deep learning)

③ 사물 인터넷(internet of things)

④ 클라우드 컴퓨팅(cloud computing)

- 끝 -

정답 4번

출제 포인트

이 문제는 서비스 모델, 자원 임대, 기반 기술이라는 특징을 통해 클라우드 컴퓨팅을 식별하는 문제이다.

핵심 개념

클라우드 컴퓨팅은 인터넷을 통해 서버, 저장 공간, 플랫폼, 소프트웨어 등의 컴퓨팅 자원을 필요한 만큼 빌려 사용하는 기술이다. 서비스 제공 범위에 따라 IaaS, PaaS, SaaS로 구분하며, 가상화와 자원 프로비저닝 등을 이용해 자원을 탄력적으로 제공한다.

빅데이터는 대규모·다양한 데이터를 수집하고 분석하여 가치를 얻는 기술이고, 딥 러닝은 다층 인공 신경망을 이용하는 기계 학습 기법이며, 사물 인터넷은 사물에 센서와 통신 기능을 부여해 서로 연결하는 기술이다.

NoteBase

선지별 해설

① X — 빅데이터는 대규모·다양한 데이터를 수집·저장·처리·분석하는 기술이므로, 보기의 'IaaS·PaaS·SaaS 서비스 모델로 구분하고 자원을 임대한다'는 설명에 해당하지 않는다. 올바르게는 보기의 설명이 클라우드 컴퓨팅에 해당한다.
② X — 딥 러닝은 여러 층의 인공 신경망이 데이터의 특징을 학습하는 기계 학습 기법이므로, 필요한 만큼 컴퓨팅 자원을 임대하는 기술이 아니다. 올바르게는 필요한 만큼 자원을 임대해 쓰는 기술을 클라우드 컴퓨팅이라고 한다.
③ X — 사물 인터넷은 센서와 통신 기능을 갖춘 사물을 인터넷으로 연결하여 데이터를 주고받는 기술이다. 보기의 'IaaS·PaaS·SaaS, 자원 임대, 가상화와 프로비저닝'은 사물 인터넷이 아니라 클라우드 컴퓨팅의 특징이다.
④ O — 클라우드 컴퓨팅은 IaaS·PaaS·SaaS로 서비스를 구분하고, 이용자가 필요한 만큼 자원을 빌려 쓰게 하며, 가상화·프로비저닝·과금 체계 등을 기반으로 서비스를 제공한다.

오답 포인트

빅데이터, 딥 러닝, 사물 인터넷도 클라우드 환경을 활용할 수 있어 혼동하기 쉽다. 그러나 보기의 핵심은 활용 대상이 아니라 '컴퓨팅 자원의 서비스 제공 방식'이며, 특히 IaaS·PaaS·SaaS와 자원 임대가 결정적 단서이다.

다시 나오면 이렇게 판단

IaaS·PaaS·SaaS, 사용량 기반 과금, 필요한 만큼의 자원 임대, 가상화, 프로비저닝이 제시되면 클라우드 컴퓨팅으로 판단한다. 데이터의 규모·분석은 빅데이터, 다층 신경망 학습은 딥 러닝, 사물 간 연결은 사물 인터넷으로 구분한다.

암기 포인트

클라우드 컴퓨팅은 IaaS·PaaS·SaaS, 필요한 만큼 빌려 쓰는 컴퓨팅 자원이다.

NoteBase