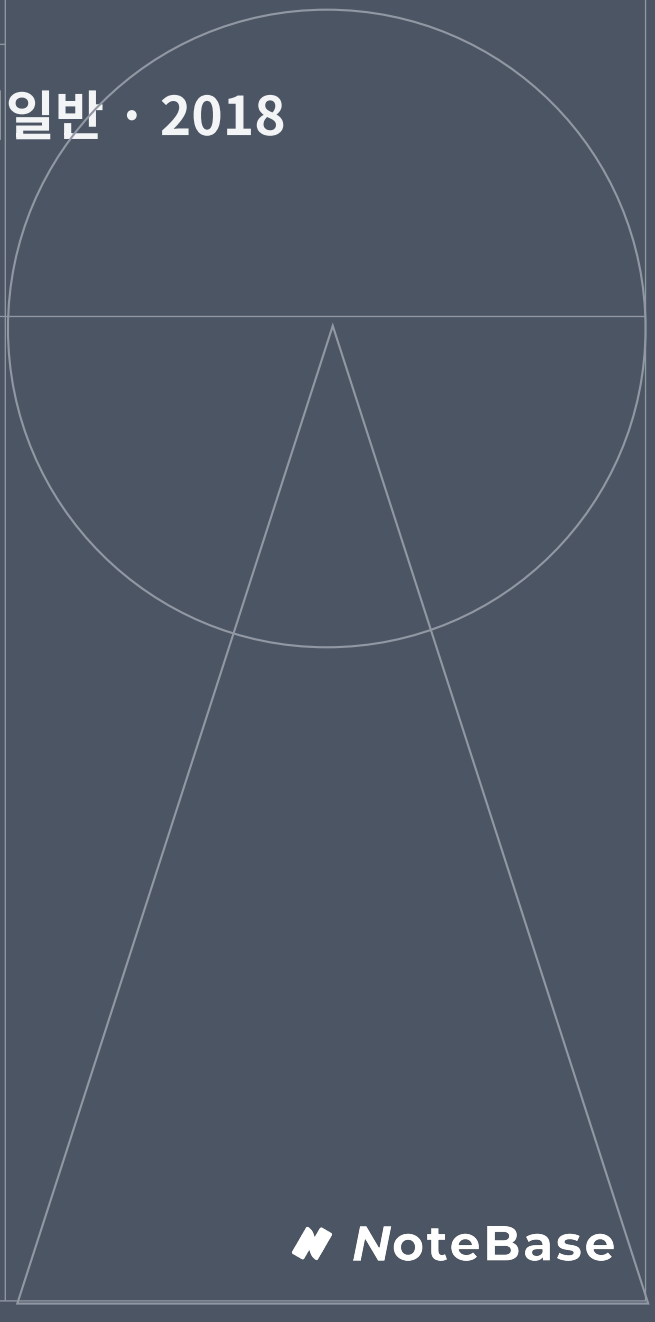
기출OX 해설집

공무원/국가직/9급 · 컴퓨터일반 · 2018



NoteBase

기출OX 해설집

공무원/국가직/9급 · 컴퓨터일반 · 2018

공무원/국가직/9급 · 컴퓨터일반 · 2018

문 1. 유닉스 운영체제에 대한 설명으로 옳지 않은 것은?

① 계층적 파일시스템과 다중 사용자를 지원하는 운영체제이다.

② BSD 유닉스의 모든 코드는 어셈블리 언어로 작성되었다.

③ CPU 이용률을 높일 수 있는 다중 프로그래밍 기법을 사용한다.

④ 사용자 프로그램은 시스템 호출을 통해 커널 기능을 사용할 수 있다.

정답 2번

간단한 해설

BSD 유닉스는 **주로 C 언어**로 작성되었으며, 하드웨어에 밀접한 일부 부분에 어셈블리 언어를 사용한다. 따라서 '모든 코드가 어셈블리 언어로 작성되었다'는 설명은 잘못되었다.

선지별 해설

① O: 유닉스는 루트 디렉터리에서 하위 디렉터리와 파일로 이어지는 **트리 형태의 계층적 파일시스템**을 사용한다. 또한 여러 사용자가 각각의 계정으로 시스템 자원을 이용할 수 있는 다중 사용자 운영체제이다.

② X: BSD 유닉스는 **주로 C 언어로 작성**되었으며, 하드웨어에 밀접한 일부 부분에 어셈블리 언어를 사용한다. 모든 코드가 어셈블리 언어로 작성된 것은 아니며, C 언어의 사용은 서로 다른 하드웨어로 이식하기 쉽게 하는 데 기여한다.

③ O: 다중 프로그래밍은 여러 프로그램을 메모리에 두고 CPU를 번갈아 할당하는 기법이다. 한 프로그램이 입출력 완료를 기다리는 동안 **다른 프로그램을 실행**하면 CPU의 유휴 시간을 줄여 이용률을 높일 수 있다.

④ O: 커널은 프로세스·메모리·파일 등 시스템 자원을 관리하는 운영체제의 핵심 부분이다. 사용자 프로그램은 **시스템 호출**을 통해 파일 읽기·쓰기와 같은 커널의 서비스를 요청한다.

NoteBase

문 2. 다음에서 설명하는 해킹 공격 방법은?

공격자는 사용자의 합법적 도메인을 탈취하거나 도메인 네임 시스템(DNS) 또는 프락시 서버의 주소를 변조하여, 사용자가 진짜 사이트로 오인하여 접속하도록 유도한 후 개인정보를 훔친다.

① 스니핑(Sniffing)
② 파밍(Pharming)
③ 트로이 목마(Trojan Horse)
④ 하이재킹(Hijacking)

정답 2번

간단한 해설

DNS나 프락시 서버의 주소 정보를 변조하여 정상 사이트에 접속하려는 사용자를 **가짜 사이트로 유도**하고 개인정보를 훔치는 수법은 **파밍(Pharming)**이다.

공통 해설

파밍(Pharming)은 도메인이나 주소 연결 정보를 조작하여 사용자를 위조 사이트로 유도하는 공격이다. DNS는 사람이 사용하는 도메인 이름을 서버의 IP 주소로 연결해 주는 시스템이다. 이 연결 정보가 변조되면 사용자가 **정상적인 도메인 이름을 입력해도 공격자가 지정한 서버로 접속**할 수 있다. 사용자는 진짜 사이트에 접속했다고 믿고 로그인 정보 등 개인정보를 입력하고, 공격자는 이를 탈취한다.

지문에서는 합법적 도메인을 탈취하거나 DNS 또는 프락시 서버의 주소를 변조하여 접속을 유도한다. 따라서 공격을 구별하는 핵심은 단순히 개인정보를 훔친다는 결과가 아니라 **접속 대상을 가짜 사이트로 바꾸는 방식**이며, 이는 파밍의 특징이다.

NoteBase

문 3. 다음 SQL 명령어에서 DDL(Data Definition Language) 명령어만을 모두 고른 것은?

ㄱ. ALTER
ㄴ. DROP
ㄷ. INSERT
ㄹ. UPDATE

① ㄱ, ㄴ
② ㄴ, ㄷ
③ ㄴ, ㄹ
④ ㄷ, ㄹ

정답 1번

간단한 해설

DDL은 테이블 같은 데이터베이스 객체의 **구조를 정의·변경·삭제**하는 명령어이다. DROP은 저장된 행만 지우는 것이 아니라 **객체 자체를 삭제**하므로 DDL에 속한다.

보기별 해설

DDL은 데이터베이스 객체의 구조를 다루고, DML은 그 구조 안에 저장되는 데이터를 다룬다.

ㄱ. ALTER: DDL에 속한다. 이미 존재하는 테이블 등 데이터베이스 객체의 **구조를 변경**한다. 예를 들어 테이블에 새로운 열을 추가하는 작업에 사용한다.

ㄴ. DROP: DDL에 속한다. 테이블 등 **데이터베이스 객체 자체를 삭제**한다. 테이블의 구조는 남겨 두고 행만 삭제하는 DELETE와 구별된다.

ㄷ. INSERT: DML(Data Manipulation Language, 데이터 조작어)에 속한다. 테이블의 구조를 바꾸는 것이 아니라 **새로운 행을 삽입**한다.

ㄹ. UPDATE: DML에 속한다. 기존 행에 저장된 **데이터 값을 수정**한다. 열을 추가하거나 열의 자료형을 바꾸는 구조 변경은 ALTER의 역할이다.

DDL에 해당하는 보기는 ㄱ, ㄴ이며, 이를 모두 고른 조합은 ①이다.

NoteBase

문 4. 다음 수식에서 이진수 Y의 값은? (단, 수식의 모든 수는 8 비트 이진수이고 1의 보수로 표현된다)

$11110100_{(2)} + Y = 11011111_{(2)}$

① $11101001_{(2)}$
② $11101010_{(2)}$
③ $11101011_{(2)}$
④ $11101100_{(2)}$

정답 2번

간단한 해설

1의 보수로 해석하면 주어진 두 수는 각각 -11과 -32이므로 Y는 -21이다. 21의 8비트 표현에서 **모든 비트를 반전**하면 $11101010_{(2)}$을 얻는다.

계산 과정

주어진 두 수는 최상위 비트가 1이므로 음수이다. 1의 보수로 표현된 음수는 **모든 비트를 반전하면 절댓값**을 얻는다.

주어진 두 수의 해석: $11110100_{(2)}$을 반전하면 $00001011_{(2)}$이며, 이는 8 + 2 + 1 = 11이므로 원래 수는 -11이다. $11011111_{(2)}$을 반전하면 $00100000_{(2)}$이며, 이는 32이므로 원래 수는 -32이다.

Y의 값 계산: 주어진 식을 십진수로 옮기면 $-11 + Y = -32$이다. 양변에 11을 더하면 $Y = -32 + 11 = -21$이다.

Y의 8비트 표현: 21 = 16 + 4 + 1이므로 양수 21은 $00010101_{(2)}$이다. -21은 이를 **비트별로 반전**한 $11101010_{(2)}$으로 표현한다. 이 값은 ②의 이진수와 일치한다.

NoteBase

문 5. 다음 진리표를 만족하는 부울 함수로 옳은 것은? (단, ·은 AND, ⊕는 XOR, ⊙는 XNOR 연산을 의미한다)

입력			출력
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

① $Y = A \cdot B \oplus C$

② $Y = A \oplus B \odot C$

③ $Y = A \oplus B \oplus C$

④ $Y = A \odot B \odot C$

정답 2번

간단한 해설

진리표는 입력 세 값 중 **1의 개수가 짝수일 때 1**을 출력한다. 이는 두 입력의 XOR 결과와 나머지 입력이 **같을 때 1이 되는 XNOR**로 나타낼 수 있다.

NoteBase

계산 과정

연산 기준: 입력 A, B, C와 출력 Y는 각각 0 또는 1이다. AND는 두 값이 모두 1일 때만 1이고, XOR는 두 값이 다를 때 1, 같을 때 0이다. XNOR는 XOR의 출력을 반전하므로 **두 값이 같을 때 1**, 다를 때 0이다.

진리표의 규칙: 출력 Y가 1인 입력 (A, B, C)는 (0, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 0)이다. 이 네 경우에는 **1의 개수가 0개 또는 2개**이고, 나머지 네 경우에는 1개 또는 3개이므로 Y가 0이다.

함수 도출: $T = A \oplus B$로 두면 T는 A와 B에 포함된 1의 개수가 짝수일 때 0, 홀수일 때 1이다. C가 0이면 T도 0이어야 전체 1의 개수가 짝수이고, C가 1이면 T도 1이어야 전체 1의 개수가 짝수이다. 따라서 **T와 C가 같을 때만 Y가 1**이므로 $Y = T \odot C$이다.

다른 식 확인: 첫 행의 입력 (0, 0, 0)을 대입하면 ①은 AND 결과 0과 C의 XOR이므로 0이고, ③도 XOR를 연속 적용하므로 0이다. ④는 먼저 $0 \odot 0 = 1$, 이어 $1 \odot 0 = 0$이 된다. 세 식 모두 첫 행의 출력 1과 다르다.

선지 연결: T를 원래 식으로 바꾸면 $Y = (A \oplus B) \odot C$이다. 이 식은 진리표의 여덟 입력 조합을 모두 만족하며 ②에 해당한다.

NoteBase

문 6. 스레싱(Thrashing)에 대한 설명으로 옳지 않은 것은?

① 프로세스의 작업 집합(Working Set)이 새로운 작업 집합으로 전이 시 페이지 부재율이 높아질 수 있다.

② 작업 집합 기법과 페이지 부재 빈도(Page Fault Frequency) 기법은 한 프로세스를 중단(Suspend)시킴으로써 다른 프로세스들의 스레싱을 감소시킬 수 있다.

③ 각 프로세스에 설정된 작업 집합 크기와 페이지 프레임 수가 매우 큰 경우 다중 프로그래밍 정도(Degree of Multiprogramming)를 증가시킨다.

④ 페이지 부재 빈도 기법은 프로세스의 할당받은 현재 페이지 프레임 수가 설정한 페이지 부재율의 하한보다 낮아지면 보유한 프레임 수를 감소시킨다.

정답 3번

간단한 해설

③의 '다중 프로그래밍 정도를 증가시킨다'는 스레싱을 악화시킬 수 있다. 프로세스들의 작업 집합이 요구하는 총 프레임 수가 가용량을 초과하면, 일부 프로세스를 중단하여 **다중 프로그래밍 정도를 줄여야 한다**.

NoteBase

선지별 해설

스레싱은 필요한 페이지를 메모리에 유지하지 못해 실행보다 페이지 교체에 시간을 더 많이 쓰는 상태이다. 작업 집합은 프로세스가 최근 일정 구간에서 참조한 페이지들의 집합이며, 프레임은 페이지를 담는 물리 메모리 공간이다.

① O: 작업 집합이 바뀌면 새로 사용하는 페이지가 메모리에 없을 수 있다. 이 페이지들을 적재하는 동안 **페이지 부재율이 일시적으로 높아질 수 있다**.

② O: 메모리가 부족할 때 한 프로세스를 중단하고 그 프로세스의 프레임을 회수하면, 다른 프로세스에 필요한 프레임을 더 할당할 수 있다. 작업 집합 기법과 페이지 부재 빈도 기법 모두 이러한 **메모리 부족 시의 중단과 재할당**으로 스레싱을 줄일 수 있다.

③ X: 다중 프로그래밍 정도는 메모리에서 함께 실행 대상으로 유지하는 프로세스의 수를 뜻한다. 각 프로세스의 작업 집합이 커져 총 프레임 요구량이 가용량을 초과하면, 프로세스를 더 늘릴수록 프레임 부족이 심해진다. 이때는 일부 프로세스를 중단해 **다중 프로그래밍 정도를 감소**시켜야 한다. 작업 집합이 크다는 사실만으로 증가 여부를 정하지 않고, 총 요구량과 가용 프레임 수를 비교한다.

④ O: (페이지 부재율을 비교한다는 취지로 읽을 때) 페이지 부재 빈도 기법은 **측정한 페이지 부재율이 하한보다 낮으면** 해당 프로세스의 프레임 일부를 회수하고, 상한보다 높으면 프레임을 추가로 할당한다. 다만 원문의 '현재 페이지 프레임 수가 설정한 페이지 부재율의 하한보다 낮아지면'은 프레임 수와 비율을 비교한 부정확한 표현이다. 문언 그대로의 비교는 성립하지 않으며, 하한과 비교할 대상은 페이지 부재율이다.

NoteBase

문 7. 인공신경망에 대한 설명으로 옳은 것만을 모두 고른 것은?

ㄱ. 단층 퍼셉트론은 배타적 합(Exclusive-OR) 연산자를 학습할 수 있다.
ㄴ. 다층 신경망은 입력 층, 출력 층, 하나 이상의 은닉층들로 구성된다.
ㄷ. 뉴런 간 연결 가중치(Connection Weight)를 조정하여 학습한다.
ㄹ. 생물학적 뉴런 망을 모델링한 방식이다.

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

정답 4번

간단한 해설

단층 퍼셉트론은 입력을 하나의 직선이나 초평면으로 나눌 수 있는 **선형 분리 가능한 문제**만 학습할 수 있다.
XOR은 선형 분리가 불가능하므로, 이를 구현하려면 비선형 활성화 함수를 사용하는 은닉층을 둔 다층 신경망이 필요하다.

NoteBase

보기별 해설

ㄱ. X: 단층 퍼셉트론은 입력의 가중합을 임계값과 비교하여 출력을 결정하므로, 두 부류를 하나의 직선이나 초평면으로 나눌 수 있는 문제만 학습할 수 있다. XOR은 두 입력이 같으면 0, 다르면 1을 출력한다. 입력 평면에서 (0, 0)과 (1, 1)은 출력 0이고, (0, 1)과 (1, 0)은 출력 1이어서 하나의 직선으로 두 부류를 나눌 수 없다. 따라서 XOR은 **단층 퍼셉트론으로 학습할 수 없으며**, 비선형 활성화 함수를 사용하는 은닉층을 둔 다층 신경망으로 구현할 수 있다.

ㄴ. O: 다층 신경망은 데이터를 받아들이는 입력층, 중간 처리를 수행하는 **하나 이상의 은닉층**, 처리 결과를 내보내는 출력층으로 구성된다. 은닉층은 입력과 출력 사이에서 입력 특징을 변환한다.

ㄷ. O: 연결 가중치는 앞선 뉴런의 출력이 다음 뉴런에 미치는 영향의 크기와 방향을 정한다. 신경망은 학습 규칙에 따라 **연결 가중치를 조정**하여 입력에 대한 출력이 달라지도록 한다. 예를 들어 지도학습에서는 예측 출력과 목표 출력의 오차를 줄이는 방향으로 가중치를 갱신한다.

ㄹ. O: 인공신경망은 생물학적 뉴런들이 연결되어 신호를 주고받는 구조와 작동 방식에서 착안한 모델이다. 인공 뉴런은 입력 신호를 가중합하고 활성화 함수를 적용하여 다음 뉴런에 전달한다. 이는 생물학적 신경망의 정보 처리를 **수학적으로 단순화하여 모델링**한 것이다.

옳은 보기는 ㄴ, ㄷ, ㄹ이며, 이 조합은 ④에 해당한다.

NoteBase

문 8. 네트워크 기술에 대한 설명으로 옳지 않은 것은?

① IPv6는 인터넷 주소 크기가 128비트이고 호스트 자동 설정기능을 제공한다.

② 광대역통합망은 응용 서비스별로 약속된 서비스 레벨 보증(Service Level Agreement) 품질 수준을 보장해줄 수 있다.

③ 모바일 와이맥스(WiMAX)는 휴대형 단말기를 이용해 고속 인터넷 접속 서비스를 제공하는 무선망 기술이다.

④ SMTP(Simple Mail Transfer Protocol)는 사용자 인터페이스 구성방법을 지정하는 전송 계층 프로토콜이다.

정답 4번

간단한 해설

SMTP는 사용자 인터페이스 구성 방법을 정하는 규약이 아니라 **전자우편을 전송하는 응용 계층 프로토콜**이다. 전송 계층의 TCP를 이용하지만, SMTP 자체가 전송 계층에 속하는 것은 아니다.

선지별 해설

① O: IPv6의 주소 길이는 **128비트**이다. 또한 호스트가 주소를 자동으로 설정하는 기능을 지원한다. 예를 들어 상태 비저장 주소 자동 설정(SLAAC)에서는 호스트가 라우터가 알린 네트워크 접두사와 인터페이스 식별자를 결합하여 주소를 구성할 수 있다.

② O: 광대역통합망(BcN)은 음성·영상·데이터 등 여러 서비스를 통합하여 제공하며, **서비스별로 요구되는 품질을 구분하여 보장**할 수 있다. SLA는 서비스 제공자와 이용자 사이에 약속한 서비스 수준에 관한 협약이다. 대역폭, 지연, 패킷 손실 등 품질 지표를 관리하여 약속된 수준을 충족하도록 한다.

③ O: 모바일 와이맥스는 **이동성을 지원하는 광대역 무선 접속 기술**이다. 휴대형 단말기가 무선으로 기지국에 접속하여 고속 인터넷 서비스를 이용할 수 있게 한다.

④ X: SMTP는 **전자우편의 송신·중계에 사용하는 응용 계층 프로토콜**이다. 메일 클라이언트가 서버로 메일을 보내거나 메일 서버 사이에서 메일을 전달할 때 사용한다. 사용자 인터페이스의 구성 방법을 지정하지 않으며, 전송 계층 프로토콜도 아니다. SMTP가 메일 전송 규칙을 정하고, 그 아래의 TCP가 신뢰성 있는 데이터 전송을 담당한다.

NoteBase

문 9. 다음 Java 프로그램의 출력 값은?

```
class Super {
	Super() {
		System.out.print('A');
	}

	Super(char x) {
		System.out.print(x);
	}
}

class Sub extends Super {
	Sub() {
		super();
		System.out.print('B');
	}

	Sub(char x) {
		this();
		System.out.print(x);
	}
}

public class Test {
	public static void main(String[] args) {
		Super s1 = new Super('C');
		Super s2 = new Sub('D');
	}
}
```

① ABCD

② ACBD

③ CABD

④ CBAD

정답 3번

간단한 해설

new Sub('D')는 **this()로 Sub()를 먼저 호출**하고, Sub()는 super()로 부모 생성자를 호출한다. 따라서 부모 생성자의 A, Sub()의 B가 출력된 뒤 Sub(char x)의 D가 출력된다.

NoteBase

올바른 순서

C → A → B → D

C 출력: main의 첫 문장 new Super('C')는 문자 인수를 받는 Super(char x)를 호출한다. 이 생성자는 전달받은 x를 출력하므로 C가 나온다. **Super()가 함께 실행되는 것은 아니므로** 이때 A는 출력되지 않는다.

A 출력: 다음 문장 new Sub('D')는 Sub(char x)를 호출한다. 이 생성자의 this()는 같은 클래스의 인수 없는 생성자 Sub()를 호출하고, Sub()의 super()는 부모의 인수 없는 생성자 Super()를 호출한다. 따라서 **부모 생성자의 출력문이 먼저 실행**되어 A가 나온다. 변수 s2의 선언형이 Super여도 실제로 생성하는 객체는 Sub이므로 이 호출 흐름을 따른다.

B 출력: Super()가 끝나면 호출한 Sub()로 돌아온다. super() 다음의 System.out.print('B')가 실행되어 B가 나온다. Sub(char x)의 this() 호출은 **Sub()의 실행이 끝나야 완료**된다.

D 출력: Sub()가 끝나면 Sub(char x)의 this() 다음으로 돌아온다. 이때 x는 처음 전달받은 'D'이므로 System.out.print(x)가 D를 출력한다. 각 print는 줄을 바꾸거나 공백을 넣지 않으므로 문자들이 이어서 출력된다.

NoteBase

문 10. 개발자가 사용해야 하는 서브시스템의 가장 앞쪽에 위치하면서 서브시스템에 있는 객체들을 사용할 수 있도록 인터페이스 역할을 하는 디자인 패턴은?

① Facade 패턴
② Strategy 패턴
③ Adapter 패턴
④ Singleton 패턴

정답 1번

간단한 해설

Facade 패턴은 복잡한 서브시스템 앞에 **사용하기 쉬운 통합 인터페이스**를 제공한다. 개발자는 이 인터페이스를 통해 내부 객체들의 기능을 이용할 수 있다.

공통 해설

Facade 패턴은 여러 객체로 이루어진 서브시스템의 기능을 쉽게 이용하도록 통합된 인터페이스를 제공하는 디자인 패턴이다. 개발자가 내부 객체를 하나하나 다루는 대신 Facade에 작업을 요청하면, Facade가 필요한 내부 객체에 작업을 전달한다. 문제의 '서브시스템의 가장 앞쪽'은 이처럼 **개발자와 서브시스템 사이에서 사용 창구 역할을 한다**는 뜻이다.

예를 들어 영상 재생에 영상 처리 객체와 음성 처리 객체의 여러 작업이 필요하다면, Facade에 '재생 시작'을 요청했을 때 필요한 내부 작업들을 수행하도록 구성할 수 있다. 핵심은 서브시스템 내부의 복잡한 사용 절차를 감추고 간단한 이용 방법을 제공한다는 점이다.

NoteBase

문 11. 소프트웨어 모듈 평가 기준으로 판단할 때, 다음 4명 중 가장 좋게 설계한 사람과 가장 좋지 않게 설계한 사람을 순서대로 바르게 나열한 것은?

◯ 철수: 절차적 응집도 + 공통 결합도
◯ 영희: 우연적 응집도 + 내용 결합도
◯ 동수: 기능적 응집도 + 자료 결합도
◯ 민희: 논리적 응집도 + 스탬프 결합도

① 철수, 영희
② 철수, 민희
③ 동수, 영희
④ 동수, 민희

정답 3번

간단한 해설

모듈은 내부 기능의 관련성을 나타내는 **응집도가 높고**, 다른 모듈에 의존하는 정도인 **결합도가 낮을수록** 좋다. 동수의 기능적 응집도·자료 결합도가 가장 바람직하며, 영희의 우연적 응집도·내용 결합도가 가장 바람직하지 않다.

NoteBase

선지별 해설

응집도는 한 모듈 안의 기능들이 서로 관련된 정도이고, 결합도는 모듈 사이의 의존 정도이다. 좋은 설계는 내부 기능이 하나의 목적에 집중하면서 다른 모듈의 내부 사정에 적게 의존하므로, 응집도는 높이고 결합도는 낮추는 것이 바람직하다.

철수: 절차적 응집도는 정해진 실행 순서에 따라 수행되는 기능들을 묶은 것이다. 순서상 관련은 있지만 모두 하나의 기능에 집중하는 기능적 응집도보다는 낮다. 공통 결합도는 여러 모듈이 **전역 데이터 등을 공유**하는 것으로, 공유 데이터의 변경이 다른 모듈에 영향을 줄 수 있어 자료 결합도보다 의존성이 크다.

영희: 우연적 응집도는 **서로 관련 없는 기능들을 한 모듈에 모은 것**으로, 제시된 응집도 중 가장 낮다. 내용 결합도는 한 모듈이 다른 모듈의 내부 코드나 데이터에 직접 접근하거나 이를 변경하는 것으로, 제시된 결합도 중 가장 높다. 내부 기능의 관련성은 가장 약하고 모듈 간 의존성은 가장 강하므로 가장 좋지 않은 설계이다.

동수: 기능적 응집도는 모듈의 모든 구성 요소가 **하나의 명확한 기능을 수행**하도록 모인 것으로, 제시된 응집도 중 가장 높다. 자료 결합도는 모듈 사이에서 필요한 자료를 매개변수로 전달하는 방식으로, 제시된 결합도 중 가장 낮다. 내부 기능은 하나의 목적에 집중하고 외부와의 의존은 작으므로 가장 좋은 설계이다.

민희: 논리적 응집도는 입력 처리처럼 논리적으로 비슷한 종류의 기능들을 묶고 그중 수행할 기능을 선택하는 것이다. 하나의 구체적인 기능에 집중한 기능적 응집도보다는 낮다. 스탬프 결합도는 레코드 같은 **자료구조 전체를 전달**하는 방식으로, 일부 자료만 필요해도 자료구조에 의존하게 되므로 필요한 자료만 전달하는 자료 결합도보다 결합도가 높다.

가장 좋게 설계한 사람은 동수이고 가장 좋지 않게 설계한 사람은 영희이므로, ③의 '동수, 영희'와 일치한다.

NoteBase

문 12. 자료구조에 대한 설명으로 옳지 않은 것은?

① 데크는 삽입과 삭제를 한쪽 끝에서만 수행한다.

② 연결리스트로 구현된 스택은 그 크기가 가변적이다.

③ 배열로 구현된 스택은 구현이 간단하지만 그 크기가 고정적이다.

④ 원형연결리스트는 한 노드에서 다른 모든 노드로 접근이 가능하다.

정답 1번

간단한 해설

데크는 **양쪽 끝에서 삽입과 삭제**를 할 수 있다. 삽입과 삭제를 같은 한쪽 끝에서만 수행하는 자료구조는 스택이다.

선지별 해설

① X: 데크(deque)는 양쪽 끝을 모두 사용할 수 있는 자료구조로, **앞과 뒤에서 각각 삽입과 삭제**가 가능하다. 삽입과 삭제를 같은 한쪽 끝에서만 수행하는 것은 스택의 특성이다. 스택에서는 A 다음 B를 넣으면 나중에 넣은 B가 먼저 나오는 후입선출 방식으로 자료를 처리한다.

② O: 연결리스트로 구현한 스택은 자료를 넣을 때 노드를 동적으로 할당하고, 꺼낼 때 해당 노드를 해제할 수 있다. 따라서 **사용 가능한 메모리 범위에서** 노드 수를 늘리거나 줄일 수 있어 크기가 가변적이다.

③ O: 미리 크기를 정한 배열로 스택을 구현하면 배열과 맨 위 자료의 위치를 나타내는 인덱스로 삽입·삭제를 처리할 수 있어 구현이 간단하다. 다만 **확보한 배열의 용량은 고정**되어 있으므로, 그 용량을 넘는 자료를 넣으려면 더 큰 배열을 마련하는 등의 별도 처리가 필요하다.

④ O: 원형연결리스트는 마지막 노드의 링크가 첫 노드를 가리켜 **연결이 고리처럼 이어지는** 구조이다. 따라서 어느 노드에서 시작하더라도 다음 노드의 링크를 계속 따라가면 다른 모든 노드에 접근하고 다시 시작 노드로 돌아올 수 있다.

NoteBase

문 13. IPv4가 제공하는 기능만을 모두 고른 것은?

ㄱ. 혼잡제어
ㄴ. 인터넷 주소지정과 라우팅
ㄷ. 신뢰성 있는 전달 서비스
ㄹ. 패킷 단편화와 재조립

① ㄱ, ㄴ
② ㄴ, ㄷ
③ ㄴ, ㄹ
④ ㄷ, ㄹ

정답 3번

간단한 해설

IPv4는 **주소지정과 라우팅**으로 패킷을 목적지까지 전달하고, 필요한 경우 **단편화와 재조립**을 수행한다. 다만 최선형 전달 방식이므로 패킷의 도착을 보장하거나 혼잡을 직접 제어하지 않는다.

보기별 해설

ㄱ. X: 혼잡제어는 네트워크에 과도한 트래픽이 유입되지 않도록 전송량을 조절하는 기능이다. **IPv4 자체는 혼잡제어를 제공하지 않는다.** TCP는 네트워크의 혼잡 상황에 따라 송신량을 조절하는 혼잡제어 기능을 제공한다.

ㄴ. O: IPv4는 출발지와 목적지를 식별하는 논리적 주소를 사용한다. 라우터는 패킷의 **목적지 IPv4 주소와 라우팅 테이블**을 바탕으로 다음 전달 경로를 결정하여 다른 네트워크로 패킷을 보낸다.

ㄷ. X: IPv4는 패킷의 도착, 순서, 중복 없는 전달을 보장하지 않는 **최선형 전달 서비스**를 제공한다. 패킷이 손실되어도 IPv4 자체가 이를 재전송하지 않는다. 신뢰성 있는 전달은 TCP와 같은 상위 계층 프로토콜이 확인응답과 재전송 등을 통해 제공한다.

ㄹ. O: 패킷이 통과할 링크의 최대 전송 단위(MTU)보다 크면, **단편화가 허용된 경우** 송신 호스트나 중간 라우터가 패킷을 작은 조각으로 나눌 수 있다. IPv4에서는 헤더의 DF 비트가 0일 때 단편화가 허용되며, 나뉜 조각은 최종 목적지 호스트에서 재조립된다.

IPv4가 제공하는 기능은 ㄴ과 ㄹ이므로, 이 둘을 묶은 ③에 해당한다.

NoteBase

문 14. 결정 명령문 내의 각 조건식이 참, 거짓을 한 번 이상 갖도록 조합하여 테스트 케이스를 설계하는 방법은?

① 문장 검증 기준(Statement Coverage)

② 조건 검증 기준(Condition Coverage)

③ 분기 검증 기준(Branch Coverage)

④ 다중 조건 검증 기준(Multiple Condition Coverage)

정답 2번

간단한 해설

조건 검증 기준은 결정 명령문 안의 **각 개별 조건식이 참과 거짓을 각각 한 번 이상** 갖도록 테스트한다. 모든 조건값의 조합을 빠짐없이 시험해야 하는 것은 아니다.

공통 해설

조건 검증 기준(Condition Coverage)은 결정 명령문을 구성하는 **개별 조건식마다 참과 거짓이 모두 나타났는지** 확인하는 기준이다. 문제의 '각 조건식이 참, 거짓을 한 번 이상 갖도록'이라는 요구가 이에 해당한다. 확인 대상은 실행된 문장이나 결정 명령문 전체의 분기 결과가 아니라, 그 안의 각 조건식이다.

예를 들어 개별 조건식 A와 B가 각각 평가될 때, 첫 테스트에서 A는 참·B는 거짓, 둘째 테스트에서 A는 거짓·B는 참이라면 두 조건식 모두 참과 거짓을 한 번씩 갖는다. 따라서 조건 검증 기준을 충족한다. 이때 (참, 참)과 (거짓, 거짓)까지 시험할 필요는 없다. **각 조건의 두 결과를 확보하는 것**과 모든 조건값의 조합을 시험하는 다중 조건 검증 기준은 구별된다.

NoteBase

문 15. 가상 머신(Virtual Machine)에 대한 설명으로 옳지 않은 것은?

① 단일 컴퓨터에서 가상화를 사용하여 다수의 게스트 운영체제를 실행할 수 있다.

② 가상 머신은 사용자에게 다른 가상 머신의 동작에 간섭을 주지 않는 격리된 실행환경을 제공한다.

③ 가상 머신 모니터(Virtual Machine Monitor)를 사용하여 가상화하는 경우 반드시 호스트 운영체제가 필요하다.

④ 자바 가상 머신은 자바 바이트 코드가 다양한 운영체제 상에서 수행될 수 있도록 한다.

정답 3번

간단한 해설

③의 '반드시 호스트 운영체제가 필요하다'는 설명이 잘못되었다. 가상 머신 모니터는 **하드웨어 위에서 직접 실행되는 방식**도 있으므로, 별도의 호스트 운영체제가 필수는 아니다.

선지별 해설

① O: 가상화는 하나의 물리 컴퓨터 자원을 나누어 여러 가상 머신에 제공한다. 각 가상 머신에 게스트 운영체제를 설치할 수 있으므로, **한 컴퓨터에서 여러 운영체제**를 실행할 수 있다. 예를 들어 서로 다른 가상 머신에서 윈도와 리눅스를 함께 실행할 수 있다.

② O: 가상 머신 모니터는 각 가상 머신에 별도의 가상 하드웨어와 메모리 영역을 제공하고 자원 접근을 관리한다. 이에 따라 한 가상 머신의 프로그램이 다른 가상 머신의 메모리를 임의로 변경하지 못하도록 하는 **격리된 실행환경**을 제공한다. 물리 자원을 공유하더라도 실행환경은 논리적으로 분리된다.

③ X: 가상 머신 모니터(VMM)는 하이퍼바이저라고도 하며, 가상 머신을 생성하고 물리 자원을 배분·관리한다. **제1형 하이퍼바이저**는 하드웨어 위에서 직접 실행되므로 별도의 호스트 운영체제가 필요하지 않다. 반면 **제2형 하이퍼바이저**는 호스트 운영체제 위에서 실행된다. 따라서 호스트 운영체제의 필요 여부는 가상화 구조에 따라 달라지며, 모든 VMM에 반드시 필요한 것은 아니다.

④ O: 자바 소스 코드를 컴파일하면 자바 바이트 코드가 생성되고, 자바 가상 머신(JVM)이 이를 실행한다. 운영체제마다 그 환경에 맞는 JVM이 마련되어 있으면 **동일한 바이트 코드를 서로 다른 운영체제에서 실행**할 수 있다. JVM이 바이트 코드와 실제 실행환경 사이의 차이를 처리하기 때문이다.

NoteBase

문 16. IEEE 802.11 무선 랜에 대한 설명으로 옳은 것은?

① IEEE 802.11a는 5 GHz 대역에서 5.5Mbps의 전송률을 제공한다.

② IEEE 802.11b는 직교 주파수 분할 다중화(OFDM) 방식을 사용하여 최대 22Mbps의 전송률을 제공한다.

③ IEEE 802.11g는 5 GHz 대역에서 직접 순서 확산 대역(DSSS) 방식을 사용한다.

④ IEEE 802.11n은 다중입력 다중출력(MIMO) 안테나 기술을 사용한다.

정답 4번

간단한 해설

IEEE 802.11n은 여러 송신·수신 안테나를 활용하는 **다중입력 다중출력(MIMO)** 기술을 사용한다. 여러 데이터 흐름을 동시에 전송하여 전송률을 높일 수 있다.

선지별 해설

① X: IEEE 802.11a는 5 GHz 대역에서 OFDM 방식을 사용하며, 최대 전송률은 **54 Mbps**이다. 5.5 Mbps는 802.11a의 전송률이 아니라 802.11b가 지원하는 전송률 중 하나이다.

② X: IEEE 802.11b는 2.4 GHz 대역에서 **DSSS 계열 방식**을 사용하며, 최대 전송률은 **11 Mbps**이다. 따라서 OFDM을 사용하고 최대 22 Mbps를 제공한다는 두 부분을 모두 고쳐야 한다.

③ X: IEEE 802.11g는 **2.4 GHz 대역**에서 동작하며, 고속 전송에는 **OFDM**을 사용하여 최대 54 Mbps를 제공한다. 802.11b와의 호환을 위해 DSSS 계열 전송도 지원하지만, 선지처럼 5 GHz 대역에서 동작하는 표준은 아니다.

④ O: IEEE 802.11n은 **MIMO 안테나 기술**을 사용한다. MIMO는 송신 측과 수신 측에서 여러 안테나를 활용하는 기술이다. 공간 다중화를 이용하면 같은 주파수 채널에서 여러 데이터 흐름을 동시에 전송하여 전송률을 높일 수 있다.

NoteBase

문 17. 데이터베이스의 동시성 제어에 대한 설명으로 옳지 않은 것은? (단, T1, T2, T3는 트랜잭션이고, A는 데이터 항목이다)

① 다중버전 동시성 제어 기법은 한 데이터 항목이 변경될 때 그 항목의 이전 값을 보존한다.

② T1이 A에 배타 로크를 요청할 때, 현재 T2가 A에 대한 공유 로크를 보유하고 있고 T3가 A에 공유 로크를 동시에 요청한다면, 트랜잭션 기아 회피기법이 없는 경우 A에 대한 로크를 T3가 T1보다 먼저 보유한다.

③ 로크 전환이 가능한 상태에서 T1이 A에 대한 배타 로크를 요청할 때, 현재 T1이 A에 대한 공유 로크를 보유하고 있는 유일한 트랜잭션인 경우 T1은 A에 대한 로크를 배타 로크로 상승할 수 있다.

④ 2단계 로킹 프로토콜에서 각 트랜잭션이 정상적으로 커밋될 때까지 자신이 가진 모든 배타적 로크들을 해제하지 않는다면 모든 교착상태를 방지할 수 있다.

정답 4번

간단한 해설

배타 로크를 커밋까지 유지하는 엄격한 2단계 로킹은 **연쇄 복귀를 방지**하지만, 트랜잭션들이 서로의 로크 해제를 기다리는 **교착상태까지 방지하지는 못한다**.

NoteBase

선지별 해설

① O: 다중버전 동시성 제어는 데이터가 변경될 때 **이전 버전을 보존**하여 여러 버전을 관리한다. 읽기 작업은 자신에게 허용된 시점의 버전을 참조할 수 있으므로, 다른 트랜잭션이 데이터를 변경하더라도 필요한 과거 값을 읽을 수 있다.

② O: **공유 로크끼리는 양립**하지만, 배타 로크는 다른 트랜잭션의 공유 로크와 양립하지 않는다. 따라서 T2가 A의 공유 로크를 보유하는 동안 T1의 배타 로크 요청은 대기한다. 기아 회피를 위한 대기 요청 우선 처리 등의 제한이 없으면 T3의 공유 로크 요청은 T2의 공유 로크와 양립하므로 허용되어, T3가 T1보다 먼저 로크를 보유한다. 이런 공유 로크 요청이 계속 허용되면 T1이 장기간 기다리는 기아가 발생할 수 있다.

③ O: 공유 로크를 배타 로크로 바꾸는 것을 로크 상승이라고 한다. 다른 트랜잭션도 A의 공유 로크를 보유하고 있다면 배타 로크로 바꿀 수 없지만, **T1만 공유 로크를 보유**하고 있다면 충돌하는 다른 트랜잭션의 로크가 없다. 따라서 로크 전환이 허용된 상태에서 T1은 A의 공유 로크를 배타 로크로 상승시킬 수 있다.

④ X: 2단계 로킹에서 배타 로크를 커밋하거나 중단할 때까지 유지하는 방식은 엄격한 2단계 로킹이다. 다른 트랜잭션이 아직 확정되지 않은 변경 값을 읽지 못하게 하므로 연쇄 복귀를 방지하지만, **교착상태는 여전히 발생할 수 있다**. 예를 들어 두 데이터 항목 A와 B에 대해 T1이 A의 배타 로크를, T2가 B의 배타 로크를 보유한 채 서로 상대방이 보유한 항목의 배타 로크를 요청하면 둘 다 대기한다. 각자 가진 로크를 유지하는 것만으로는 이 순환 대기를 끊을 수 없으므로, '모든 교착상태를 방지할 수 있다'는 설명은 '연쇄 복귀를 방지할 수 있다'로 고쳐야 한다.

NoteBase

문 18. 파일구조에 대한 설명으로 옳지 않은 것은?

① VSAM은 B+ 트리 인덱스 구조를 사용한다.

② 히프 파일은 레코드들을 키 순서와 관계없이 저장할 수 있다.

③ ISAM은 레코드 삽입을 위한 별도의 오버플로우 영역을 필요로 하지 않는다.

④ 순차 파일에서 일부 레코드들이 키 순서와 다르게 저장된 경우, 파일 재구성 과정을 통해 키 순서대로 저장될 수 있다.

정답 3번

간단한 해설

ISAM은 기존 데이터 블록에 삽입 공간이 부족하면 새 레코드를 **별도의 오버플로우 영역**에 저장한다. 따라서 오버플로우 영역이 필요하지 않다는 설명은 잘못되었다.

선지별 해설

① O: VSAM의 키 순차 파일은 **B+ 트리 인덱스 구조**를 이용한다. 인덱스를 따라 특정 키의 레코드를 찾을 수 있고, 키 순서에 따른 순차 접근도 지원한다. 삽입으로 저장 공간이 차면 블록을 분할하고 인덱스를 갱신하여 구조를 유지한다.

② O: 히프 파일은 레코드의 **키 순서를 유지하지 않는** 파일 구조이다. 새 레코드를 빈 공간이 있는 위치에 저장할 수 있으므로, 저장된 레코드들이 키 순서대로 배열될 필요가 없다.

③ X: ISAM은 인덱스 구조가 정적으로 유지되므로, 새 레코드가 들어갈 데이터 블록에 여유가 없으면 **오버플로우 영역에 저장하고 연결**한다. 예를 들어 키 10과 20이 있는 블록이 가득 찬 상태에서 키 15를 삽입하면, 키 15의 레코드를 오버플로우 영역에 두고 연결하여 찾을 수 있게 한다. 따라서 '별도의 오버플로우 영역을 필요로 하지 않는다'는 표현을 '삽입 공간이 부족하면 별도의 오버플로우 영역을 사용한다'로 고쳐야 한다.

④ O: 순차 파일은 레코드를 키 순서에 따라 저장하는 구조이다. 삽입된 일부 레코드가 키 순서와 다르게 저장되어 있다면, **파일 재구성** 과정에서 기존 레코드와 추가 레코드를 함께 정렬하고 다시 기록하여 키 순서대로 배치할 수 있다.

NoteBase

문 19. 다음 C 프로그램의 출력 값은?

```c
#include <stdio.h>

int a = 10;
int b = 20;
int c = 30;

void func(void)
{
    static int a = 100;
    int b = 200;

    a++;
    b++;
    c = a;
}

int main(void)
{
    func();
    func();

    printf("a = %d, b = %d, c = %d\n", a, b, c);

    return 0;
}
```

① a = 10, b = 20, c = 30

② a = 10, b = 20, c = 102

③ a = 101, b = 201, c = 101

④ a = 102, b = 202, c = 102

정답 2번

간단한 해설

func()의 **정적 지역변수 a는 호출 사이에 값을 유지**하므로 두 번 증가하여 102가 되고, 이 값이 전역변수 c에 저장된다. main()에서 출력하는 a와 b는 func()의 지역변수가 아니라 **값이 바뀌지 않은 전역변수**이므로 각각 10과 20이다.

NoteBase

계산 과정

변수의 범위와 수명: 전역변수의 초기값은 a = 10, b = 20, c = 30이다. func() 안에서는 같은 이름의 지역변수 a와 b가 전역변수를 가리므로 a++와 b++는 지역변수만 증가시킨다. static으로 선언한 a는 100으로 한 번 초기화되고 **함수가 끝나도 값을 유지**한다. 반면 일반 지역변수 b는 호출할 때마다 200으로 초기화된다. func() 안에 지역변수 c는 없으므로 c = a는 전역변수 c에 지역변수 a의 값을 대입한다.

첫 번째 func() 호출: 지역변수 a는 100에서 101로, 지역변수 b는 200에서 201로 증가한다. 이어서 c = a에 의해 전역변수 c는 101이 된다. 함수가 끝나면 일반 지역변수 b의 수명은 끝나지만, 정적 지역변수 a의 값 101은 남는다.

두 번째 func() 호출: 정적 지역변수 a는 **이전 값 101에서 시작**하여 a++ 후 102가 된다. 일반 지역변수 b는 다시 200으로 초기화된 뒤 b++로 201이 된다. 이어서 c = a에 의해 전역변수 c는 102가 된다.

출력: main()의 printf에서 a와 b는 func() 안의 지역변수가 아닌 전역변수를 가리킨다. 전역변수 a와 b는 수정된 적이 없으므로 각각 10과 20이며, 전역변수 c는 두 번째 호출에서 저장된 102이다. 따라서 출력은 a = 10, b = 20, c = 102로, ②의 값과 일치한다.

NoteBase

문 20. 해싱(Hashing)에 대한 설명으로 옳지 않은 것은?

① 서로 다른 탐색키가 해시 함수를 통해 동일한 해시 주소로 사상될 수 있다.

② 충돌(Collision)이 발생하지 않는 해시 함수를 사용한다면 해싱의 탐색 시간 복잡도는 O(1)이다.

③ 선형 조사법(Linear Probing)은 연결리스트(Linked List)를 사용하여 오버플로우 문제를 해결한다.

④ 폴딩함수(Folding Function)는 탐색키를 여러 부분으로 나누어 이들을 더하거나 배타적 논리합을 하여 해시 주소를 얻는다.

정답 3번

간단한 해설

선형 조사법은 충돌이 나면 **해시 테이블의 다음 칸을 차례로 조사**하여 빈자리를 찾는다. 같은 해시 주소를 갖는 자료를 연결리스트로 연결하는 방식은 **체이닝**이다.

선지별 해설

① O: 해시 함수는 탐색키를 해시 테이블의 주소로 변환한다. 서로 다른 키라도 **같은 해시 주소**가 나올 수 있으며, 이를 충돌이라고 한다.

② O: 충돌이 없으면 해시 함수가 산출한 주소에서 해당 자료를 바로 찾을 수 있다. 해시 함수 계산과 테이블 한 칸 접근을 상수 시간으로 보는 일반적인 분석에서, 탐색 시간은 **자료 수와 무관한 상수 시간**인 $O(1)$이다.

③ X: 선형 조사법은 충돌이 발생한 칸의 **다음 칸부터 순서대로 조사**하여 빈 칸에 자료를 저장하는 개방 주소법이다. 예를 들어 3번 칸에서 충돌하고 4번 칸도 차 있다면, 5번 칸을 확인하여 비어 있을 때 저장한다. 연결리스트에 충돌한 자료들을 연결하는 방식은 **체이닝**이다.

④ O: 폴딩함수는 탐색키를 여러 부분으로 나누고, 각 부분을 **더하거나 배타적 논리합(XOR)**하여 합친 값으로 해시 주소를 구한다. 이렇게 나눈 부분들을 결합하는 것이 폴딩의 핵심이며, 필요한 경우 결합한 값을 해시 테이블의 주소 범위에 맞춘다.

NoteBase