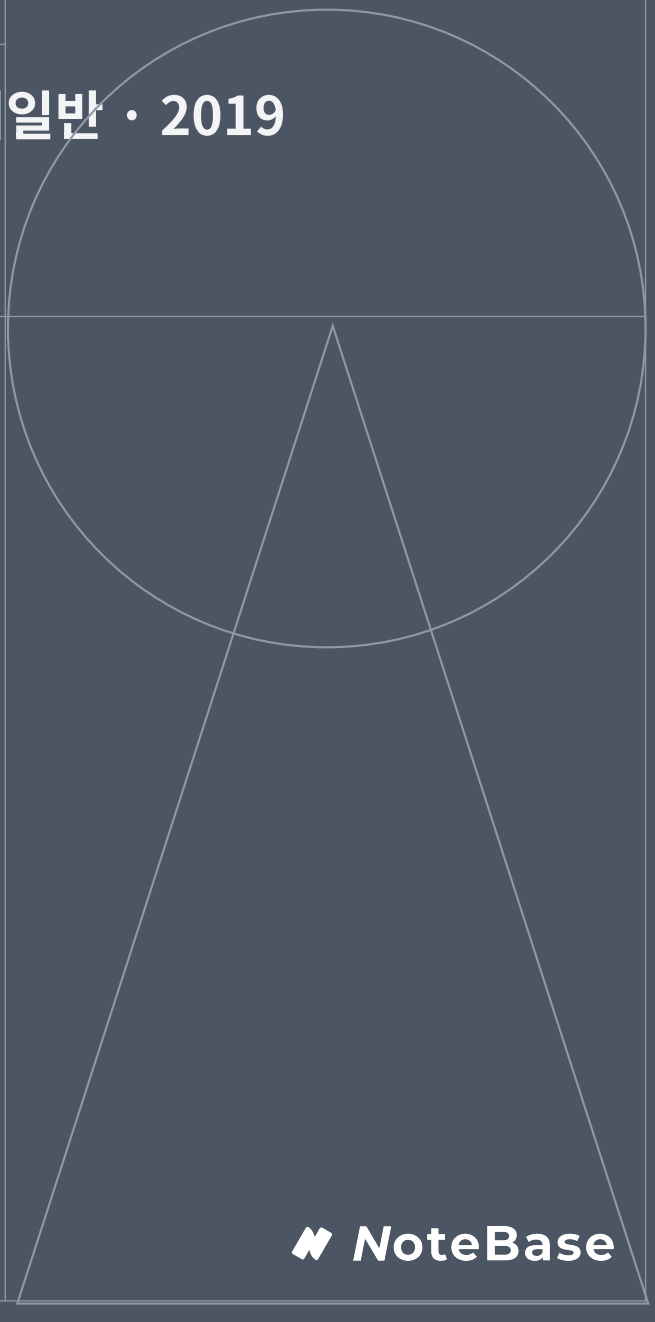
기출OX 해설집

공무원/지방직/9급 · 컴퓨터일반 · 2019



NoteBase

기출OX 해설집

공무원/지방직/9급 · 컴퓨터일반 · 2019

공무원/지방직/9급 · 컴퓨터일반 · 2019

문 1. 저급언어에 해당하는 프로그래밍 언어는?

① 어셈블리어(Assembly Language)

② 자바(Java)

③ 코볼(COBOL)

④ 포트란(Fortran)

정답 1번

간단한 해설

어셈블리어는 **기계어 명령을 기호로 표현**하여 하드웨어의 명령 구조에 가깝게 작성하는 저급언어이다.

공통 해설

저급언어와 고급언어를 구분하는 기준은 **하드웨어로부터의 추상화 수준**이다. 저급언어는 특정 CPU의 명령 체계와 밀접하게 연결되며, 고급언어는 구체적인 기계 명령보다 연산·자료·처리 절차를 중심으로 프로그램을 표현한다.

어셈블리어는 기계어 명령을 사람이 읽기 쉬운 기호로 나타내는 저급언어이며, 어셈블러를 통해 기계어로 변환된다. 반면 자바·코볼·포트란은 하드웨어의 세부 명령을 추상화하여 표현하는 고급언어이므로 저급언어에 해당하지 않는다.

NoteBase

문 2. 중앙처리장치(CPU)의 구성 요소로만 묶은 것은?

ㄱ. ALU
ㄴ. DRAM
ㄷ. PCI
ㄹ. 레지스터
ㅁ. 메인보드
ㅂ. 제어장치

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

정답 2번

간단한 해설

CPU가 연산 도중 사용하는 값을 내부에 임시 저장하는 장치는 **레지스터**이다. DRAM은 주기억장치에 쓰이는 메모리이므로 CPU 자체의 기본 구성 요소와 구분한다.

NoteBase

보기별 해설

ㄱ. ALU: CPU 구성 요소이다. 산술논리연산장치로, 덧셈·뺄셈 등의 산술연산과 AND·OR 등의 **논리연산을 수행**한다.

ㄴ. DRAM: CPU의 기본 구성 요소가 아니라 **주기억장치에 사용되는 메모리**이다. 프로그램과 데이터를 저장하며, CPU 내부에서 처리 중인 값을 잠시 보관하는 레지스터와 구분한다.

ㄷ. PCI: CPU 구성 요소가 아니라 **주변장치 연결을 위한 버스 규격**이다. 확장 카드 등의 장치가 컴퓨터 시스템과 데이터를 주고받도록 한다.

ㄹ. 레지스터: CPU 구성 요소이다. CPU 내부에서 현재 실행할 명령어, 연산에 사용할 데이터, 연산 결과 등을 **일시적으로 저장**하는 고속 저장장치이다.

ㅁ. 메인보드: CPU 구성 요소가 아니라 CPU·메모리·확장 장치 등을 장착하고 연결하는 **주회로기판**이다. CPU가 메인보드에 장착되는 것이며, 메인보드가 CPU 내부에 포함되는 것은 아니다.

ㅂ. 제어장치: CPU 구성 요소이다. 명령어를 해석하고 필요한 **제어신호를 생성**하여 연산장치·기억장치 등이 명령에 맞게 동작하도록 조정한다.

CPU의 기본 구성 요소에 해당하는 것은 ㄱ·ㄹ·ㅂ이며, ②의 묶음과 일치한다.

NoteBase

문 3. 다음에서 설명하는 네트워크 구조는?

◯ 구축 비용이 저렴하고 새로운 노드를 추가하기 쉽다.
◯ 네트워크의 시작과 끝에는 터미네이터(Terminator)가 붙는다.
◯ 연결된 노드가 많거나 트래픽이 증가하면 네트워크 성능이 크게 저하된다.

① 링(Ring)형
② 망(Mesh)형
③ 버스(Bus)형
④ 성(Star)형

정답 3번

간단한 해설

하나의 공통 통신 케이블에 여러 노드를 연결하고 그 양 끝에 터미네이터를 설치하는 구조는 버스형이다. 여러 노드가 같은 전송 매체를 공유하므로 노드나 트래픽이 늘면 성능이 저하된다.

공통 해설

버스(Bus)형은 하나의 공통 통신 케이블을 따라 여러 노드를 연결하는 구조이다. 각 노드 사이에 별도 회선을 일일이 설치할 필요가 없어 구축 비용이 저렴하며, 공통 케이블에 새 노드를 연결하여 확장하기 쉽다.

케이블의 열린 양 끝에는 **터미네이터를 설치하여 신호의 반사를 막는다**. 신호가 케이블 끝에서 반사되어 되돌아오면 통신을 방해할 수 있으므로, 터미네이터가 도달한 신호를 흡수한다. 지문의 '시작과 끝에 터미네이터가 붙는다'는 버스형을 식별하는 결정적인 단서이다.

또한 여러 노드가 **같은 전송 매체를 공유**하므로, 노드가 많아지거나 트래픽이 증가하면 전송 기회를 두고 경쟁하고 충돌에 따른 재전송이 늘어 성능이 크게 저하될 수 있다. 저렴한 구축 비용, 노드 추가의 용이성, 양 끝의 터미네이터, 부하 증가에 따른 성능 저하는 모두 이러한 버스형 구조와 연결된다.

NoteBase

문 4. 다음에서 설명하는 객체지향 프로그래밍의 특징은?

◯ 객체를 구성하는 속성과 메서드가 하나로 묶여 있다.
◯ 객체의 외부와 내부를 분리하여 외부 모습은 추상적인 내용으로 보여준다.
◯ 객체 내의 정보를 외부로부터 숨길 수도 있고, 외부에 보이게 할 수도 있다.
◯ 객체 내부의 세부 동작을 모르더라도 객체의 메서드를 통해 객체의 기능을 활용할 수 있다.

① 구조성
② 다형성
③ 상속성
④ 캡슐화

정답 4번

간단한 해설

속성과 메서드를 하나로 묶고, 외부에서 접근할 수 있는 부분을 정해 내부 구현을 감추는 특징은 **캡슐화**이다. 사용자는 공개된 메서드만으로 객체의 기능을 이용할 수 있다.

공통 해설

캡슐화는 객체의 상태를 나타내는 속성과 그 상태를 다루는 메서드를 하나로 묶고, **외부에 공개할 부분과 감출 부분을 구분**하는 것이다. 지문의 '속성과 메서드가 하나로 묶여 있다'는 하나의 객체로 구성한다는 뜻이고, '외부와 내부를 분리'한다는 것은 외부에는 기능을 사용하는 방법을 제공하면서 내부의 구체적인 구현은 감춘다는 뜻이다. 정보의 공개 범위를 정하므로 외부에서는 내부의 세부 동작을 몰라도 공개된 메서드를 호출해 기능을 이용할 수 있다. 이 설명들을 함께 나타내는 개념이 캡슐화이다.

예를 들어 계좌 객체가 잔액 속성과 입금·출금 메서드를 함께 가지고, 잔액을 외부에서 직접 바꾸지 못하게 한다고 하자. 사용자는 잔액이 내부에 어떻게 저장되는지 몰라도 입금 메서드로 금액을 넣을 수 있다. 이처럼 **내부 상태에 대한 접근을 메서드로 통제**하는 것이 캡슐화의 적용 모습이다.

NoteBase

문 5. 하나의 프로세스가 CPU를 할당받은 후에는, 스스로 CPU를 반납할 때까지 다른 프로세스가 CPU를 차지할 수 없도록 하는 스케줄링 기법에 해당하는 것만을 모두 고르면?

ㄱ. FCFS(First Come First Served)
ㄴ. RR(Round Robin)
ㄷ. SRT(Shortest Remaining Time)

① ㄱ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

정답 1번

간단한 해설

FCFS는 먼저 도착한 프로세스에 CPU를 할당하고, 실행 중인 프로세스가 스스로 반납할 때까지 CPU를 빼앗지 않는 **비선점형 스케줄링**이다.

보기별 해설

문제의 조건은 운영체제가 실행 중인 프로세스의 CPU를 강제로 회수하지 않는 비선점형 스케줄링을 뜻한다. 선점형은 프로세스가 스스로 반납하기 전에도 일정한 조건에 따라 CPU를 회수할 수 있다.

ㄱ. FCFS — 비선점형: 준비 큐에 도착한 순서대로 CPU를 할당한다. 실행 중인 프로세스는 작업을 마치거나 입출력을 기다리기 위해 CPU를 반납할 때까지 실행을 계속하며, **다른 프로세스가 도착해도 CPU를 빼앗기지 않는다**.

ㄴ. RR — 선점형: 각 프로세스에 일정한 시간 할당량을 주고 순환하며 실행한다. 실행이 끝나지 않았더라도 **시간 할당량이 만료되면 CPU를 회수**하고 준비 큐의 뒤로 보내므로, 스스로 반납할 때까지 실행을 보장하는 방식이 아니다.

ㄷ. SRT — 선점형: CPU 실행의 남은 시간이 가장 짧은 프로세스를 선택한다. 실행 중인 프로세스보다 **남은 실행 시간이 더 짧은 프로세스가 준비 상태가 되면** 현재 실행을 중단하고 그 프로세스에 CPU를 넘길 수 있다. 이는 비선점형 SJF와 구별되는 점이다.

비선점형에 해당하는 것은 ㄱ의 FCFS뿐이며, 이 조합은 ①에 해당한다.

NoteBase

문 6. 프로그램 내장 방식에 대한 설명으로 옳지 않은 것은?

① 프로그램 내장 방식을 사용한 최초의 컴퓨터는 에니악(ENIAC)이다.

② 현재 사용되는 대부분의 컴퓨터는 프로그램 내장 방식을 사용하고 있다.

③ 컴퓨터가 작업을 할 때마다 설치된 스위치를 다시 세팅해야 하는 번거로움을 해결하기 위해 폰 노이만이 제안하였다.

④ 프로그램과 자료를 내부의 기억장치에 저장한 후 프로그램 내의 명령문을 순서대로 꺼내 해독하고 실행하는 개념이다.

정답 1번

간단한 해설

에니악(ENIAC)은 처음에 **스위치와 배선을 바꾸어 프로그램을 설정**하는 방식을 사용했다. 최초의 실용적인 프로그램 내장 컴퓨터는 **에드삭(EDSAC)**이다.

선지별 해설

① X: 초기 에니악은 명령어를 기억장치에 저장해 실행하는 방식이 아니라, 스위치 설정과 배선 연결로 수행할 작업을 지정했다. 프로그램을 전자식 기억장치에 저장해 처음 실행한 실험용 컴퓨터는 맨체스터 베이비(1948년)이며, **최초의 실용적인 프로그램 내장 컴퓨터는 에드삭(EDSAC, 1949년)**이다.

② O: 대부분의 현대 컴퓨터는 실행할 프로그램을 기억장치에 저장하고, 중앙처리장치가 그 명령어를 읽어 실행한다. 같은 컴퓨터에서도 **저장된 프로그램을 바꾸어 서로 다른 작업**을 수행할 수 있다.

③ O: 초기 컴퓨터는 새로운 작업을 수행하려면 스위치를 다시 설정하거나 배선을 바꾸어야 했다. 폰 노이만은 EDVAC 설계 보고서에서 **프로그램도 자료처럼 기억장치에 저장**하는 구상을 제시했다. 이 방식에서는 작업을 바꿀 때마다 하드웨어의 연결을 바꾸는 대신 실행할 프로그램을 기억장치에 넣는다.

④ O: 프로그램 내장 방식은 프로그램의 명령어와 처리할 자료를 내부 기억장치에 저장하는 방식이다. 중앙처리장치는 명령어를 **인출 → 해독 → 실행**하는 과정을 반복한다. 기본적으로 다음 명령어로 진행하되, 분기 명령을 실행하면 지정된 위치의 명령어로 이동한다.

NoteBase

문 7. CISC(Complex Instruction Set Computer)와 RISC(Reduced Instruction Set Computer)에 대한 설명으로 옳지 않은 것은?

① CISC 구조에서 명령어의 길이는 가변적이다.

② 전형적인 RISC 구조의 명령어는 메모리의 피연산자를 직접 처리한다.

③ RISC 구조는 명령어 처리구조를 단순화시켜 기계어 명령의 수를 줄인 것을 말한다.

④ CISC 구조는 RISC 구조에 비해서 상대적으로 명령어 실행 단계가 많고 회로 설계가 복잡하다.

정답 2번

간단한 해설

전형적인 RISC는 **레지스터에 있는 피연산자**로 산술·논리 연산을 수행한다. 메모리의 데이터는 load 명령으로 레지스터에 가져오고, 처리 결과는 store 명령으로 메모리에 저장한다.

선지별 해설

① O: CISC는 다양한 연산과 주소 지정 방식을 지원하므로, 명령어에 담는 피연산자나 주소 정보에 따라 **명령어의 길이가 달라진다**. 전형적인 RISC는 명령어 길이를 고정하여 해독 구조를 단순하게 한다.

② X: 전형적인 RISC는 메모리 접근과 연산을 분리하는 **load/store 구조**이다. 산술·논리 명령은 메모리의 피연산자를 직접 처리하지 않고 레지스터의 값을 처리한다. 예를 들어 메모리에 있는 두 값을 더하려면 먼저 load로 각 값을 레지스터에 가져온 뒤 덧셈을 수행하고, 필요하면 store로 결과를 메모리에 저장한다.

③ O: RISC는 복잡하고 다양한 명령 대신 단순하고 자주 쓰는 명령을 중심으로 **명령어의 종류와 처리 구조를 단순화**한다. 여기서 기계어 명령의 수를 줄인다는 것은 지원하는 명령어 종류를 줄인다는 뜻이다. 같은 작업을 수행하는 프로그램의 명령어 개수가 반드시 줄어든다는 뜻은 아니며, 복잡한 작업에는 여러 단순 명령이 필요할 수 있다.

④ O: 전통적인 CISC는 복잡한 명령을 여러 내부 동작으로 나누어 처리하므로, 단순한 명령 중심의 RISC보다 명령 하나를 실행하는 과정이 복잡하고 여러 단계가 필요하다. 다양한 명령 형식과 주소 지정 방식을 해독·제어해야 하므로 **명령어 해독 및 제어 회로도 복잡**하다.

NoteBase

문 8. 릴레이션 $R = \{A, B, C, D, E\}$이 함수적 종속성들의 집합 $FD = \{A \rightarrow C, \{A, B\} \rightarrow D, D \rightarrow E, \{A, B\} \rightarrow E\}$를 만족할 때, R이 속할 수 있는 가장 높은 차수의 정규형으로 옳은 것은? (단, 기본키는 복합속성 $\{A, B\}$이고, 릴레이션 R의 속성 값은 더 이상 분해될 수 없는 원자 값으로만 구성된다)

① 제1정규형

② 제2정규형

③ 제3정규형

④ 보이스/ 코드 정규형

정답 1번

간단한 해설

모든 속성 값이 원자 값이므로 제1정규형은 만족한다. 그러나 $A \rightarrow C$는 복합키 $\{A, B\}$의 일부인 A만으로 C가 결정되는 **부분 함수 종속**이므로, **제2정규형을 만족하지 못한다**.

공통 해설

제1정규형은 각 속성 값이 더 이상 분해되지 않는 원자 값으로 구성되는 정규형이다. 문제에서 이 조건을 명시했으므로 주어진 릴레이션은 제1정규형을 만족한다.

제2정규형은 제1정규형을 만족하면서, 후보키에 속하지 않는 모든 속성이 후보키 전체에 완전 함수 종속되어야 한다. 즉, 그러한 속성이 **후보키의 일부만으로 결정되는 부분 함수 종속이 없어야 한다**.

주어진 복합키는 $\{A, B\}$이고, 키에 속하지 않는 C에 대해 $A \rightarrow C$가 성립한다. 이는 A와 B를 모두 알아야 C가 결정되는 것이 아니라, A만 알아도 C가 결정된다는 뜻이다. 따라서 C는 복합키에 부분 함수 종속되어 제2정규형 조건을 위반한다. $\{A, B\} \rightarrow D$와 $\{A, B\} \rightarrow E$가 성립하더라도 C의 부분 함수 종속은 없어지지 않는다.

제3정규형과 보이스/코드 정규형도 제2정규형의 조건을 충족해야 하므로 성립할 수 없다. 따라서 주어진 릴레이션이 만족하는 가장 높은 정규형은 **제1정규형**이다.

NoteBase

문 9. 인터넷의 전송 계층에서 사용하는 프로토콜로 TCP와 UDP가 있다. TCP와 UDP 모두에서 제공하지 않는 기능은?

① 연결 설정(Connection Setup)

② 오류 검출(Error Detection)

③ 지연시간 보장(Delay Guarantee)

④ 혼잡 제어(Congestion Control)

정답 3번

간단한 해설

TCP와 UDP 모두 데이터가 **정해진 시간 안에 도착하는 것을 보장하지 않는다**. TCP가 재전송이나 혼잡 제어를 수행하더라도 전송 지연의 상한이 보장되는 것은 아니다.

선지별 해설

① 연결 설정: TCP는 데이터를 전송하기 전에 **3방향 핸드셰이크로 연결을 설정**한다. UDP는 사전 연결 설정 없이 데이터그램을 전송한다. 따라서 연결 설정은 TCP가 제공하는 기능이다.

② 오류 검출: TCP와 UDP는 모두 **체크섬을 이용한 오류 검출**을 지원한다. 수신 측은 체크섬을 검사하여 전송 중 데이터가 손상되었는지 확인한다. 오류 검출은 손상 여부를 알아내는 기능이므로, UDP가 손실된 데이터를 자체적으로 재전송하지 않는다는 사실과 구별해야 한다.

③ 지연시간 보장: **TCP와 UDP 모두 제공하지 않는다**. 데이터가 도착하기까지 걸리는 시간은 네트워크의 혼잡과 라우터의 대기 시간 등에 따라 달라진다. TCP의 재전송은 손실된 데이터를 다시 보내는 기능이며, 일정 시간 이내의 도착을 보장하는 기능은 아니다.

④ 혼잡 제어: TCP는 네트워크가 혼잡해지면 송신량을 줄이는 등 **전송량을 조절**한다. UDP 자체에는 이러한 혼잡 제어 기능이 없다. 따라서 혼잡 제어는 TCP가 제공하는 기능이다.

두 프로토콜 모두 제공하지 않는 기능은 지연시간 보장이다.

NoteBase

문 10. 유비쿼터스를 응용한 컴퓨팅 기술에 대한 설명으로 옳지 않은 것은?

① 엑조틱 컴퓨팅(Exotic Computing)은 스스로 생각하여 현실세계와 가상세계를 연계해 주는 컴퓨팅 기술이다.

② 노매딕 컴퓨팅(Nomadic Computing)은 장소에 상관없이 다양한 정보기기가 편재되어 있어 사용자가 정보기기를 휴대할 필요가 없는 컴퓨팅 기술이다.

③ 디스포절 컴퓨팅(Disposable Computing)은 컴퓨터가 센서 등을 통해 사용자의 상황을 인식하여 사용자가 필요로 하는 정보를 제공해 주는 컴퓨팅 기술이다.

④ 웨어러블 컴퓨팅(Wearable Computing)은 컴퓨터를 옷이나 안경처럼 착용할 수 있게 해줌으로써 컴퓨터를 인간의 몸의 일부로 여길 수 있도록 하는 컴퓨팅 기술이다.

정답 3번

간단한 해설

③의 '센서 등을 통해 사용자의 상황을 인식하여 필요한 정보를 제공한다'는 설명은 **감지 컴퓨팅(Sentient Computing)**에 해당한다. 디스포절 컴퓨팅은 **한 번 쓰고 버릴 수 있을 만큼 저렴한 컴퓨터**를 구현하려는 기술이다.

선지별 해설

① O: 엑조틱 컴퓨팅은 컴퓨터가 스스로 판단하여 **현실세계와 가상세계를 연계**하는 기술이다. 물리적인 현실과 컴퓨터 속 전자공간을 연결한다는 점에 초점을 둔다.

② O: 노매딕 컴퓨팅은 장소를 옮겨도 그곳에 갖추어진 정보기기를 이용할 수 있는 환경을 뜻한다. 주변에 다양한 정보기기가 편재되어 있으므로 **자신의 정보기기를 따로 휴대할 필요가 없다**는 설명과 연결된다.

③ X: 센서로 사용자의 상황을 파악하고 그 상황에 맞는 정보를 제공하는 것은 **감지 컴퓨팅**이다. 디스포절 컴퓨팅은 '일회용 컴퓨팅'으로, 한 번 사용하고 버릴 수 있을 만큼 컴퓨터의 가격을 낮추어 여러 물건에 컴퓨팅 기능을 적용하려는 기술이다. 두 기술은 각각 상황 인식과 저비용 구현에 초점을 둔다.

④ O: 웨어러블 컴퓨팅은 컴퓨터를 옷이나 안경처럼 **몸에 착용하는 형태**로 구현하는 기술이다. 몸에 지니고 사용하는 방식이므로 컴퓨터를 신체의 일부처럼 이용할 수 있게 한다.

NoteBase

문 11. 컴퓨터 명령어 처리 시 필요한 유효 주소(Effective Address)를 찾기 위한 주소 지정 방식에 대한 설명으로 옳지 않은 것은?

① 즉시 주소 지정 방식(Immediate Addressing Mode)은 유효 데이터가 명령어 레지스터 내에 있다.

② 간접 주소 지정 방식(Indirect Addressing Mode)으로 유효 데이터에 접근하는 경우 주기억장치 최소접근횟수는 2이다.

③ 상대 주소 지정 방식(Relative Addressing Mode)은 프로그램 카운터와 명령어 내의 주소필드 값을 결합하여 유효 주소를 도출한다.

④ 레지스터 주소 지정 방식(Register Addressing Mode)은 직접 주소 지정 방식(Direct Addressing Mode)보다 유효 데이터 접근속도가 느리다.

정답 4번

간단한 해설

레지스터 주소 지정 방식은 CPU 내부 레지스터에서 데이터를 바로 읽으므로, 주기억장치에서 데이터를 읽는 직접 주소 지정 방식보다 **데이터 접근속도가 빠르다**. ④의 '느리다'는 두 방식의 속도 관계를 반대로 설명한 것이다.

선지별 해설

① O: 즉시 주소 지정 방식은 **명령어 자체에 피연산자 값**이 들어 있는 방식이다. 명령어를 인출하여 명령어 레지스터에 저장하면 그 안에 사용할 데이터도 있으므로, 데이터를 얻기 위해 주기억장치를 추가로 읽을 필요가 없다.

② O: 주기억장치를 이용한 간접 주소 지정 방식에서는 명령어의 주소필드가 가리키는 곳에서 먼저 **실제 데이터가 있는 주소**를 읽고, 그 주소의 주기억장치에서 데이터를 읽는다. 따라서 명령어 인출을 제외한 데이터 접근에는 최소 2회의 주기억장치 접근이 필요하다.

③ O: 상대 주소 지정 방식은 **프로그램 카운터의 값에 주소필드의 변위를 더해** 유효 주소를 구한다. 주소필드가 절대 주소를 직접 나타내는 것이 아니라, 프로그램 카운터를 기준으로 데이터가 얼마나 떨어져 있는지를 나타낸다.

④ X: 레지스터 주소 지정 방식은 주소필드가 지정한 CPU 내부 레지스터에서 데이터를 읽는다. 직접 주소 지정 방식은 주소필드가 가리키는 주기억장치에서 데이터를 읽는다. 레지스터는 주기억장치보다 접근속도가 빠르므로, 선지의 '느리다'는 **'빠르다'로 고쳐야 한다**.

NoteBase

문 12. 컴퓨터 시스템에서 교착상태의 해결 방안에 대한 설명으로 옳지 않은 것은?

① 교착상태가 발생할 가능성을 사전에 없앤다.

② 하나의 프로세스만이 한 시점에서 하나의 자원을 사용할 수 있게 한다.

③ 교착상태가 탐지되면, 교착상태와 관련된 프로세스와 자원을 시스템으로부터 제거한다.

④ 교착상태가 발생할 가능성을 인정하고, 교착상태가 발생하려고 할 때 이를 회피하도록 한다.

정답 2번

간단한 해설

하나의 자원을 한 번에 하나의 프로세스만 사용하게 하는 것은 **상호 배제**이다. 이는 교착상태의 발생 필요조건 중 하나이므로, 그 자체를 교착상태의 해결 방안으로 볼 수 없다.

선지별 해설

① O: 교착상태 예방에 해당한다. 교착상태는 상호 배제, 점유 대기, 비선점, 순환 대기의 네 조건이 모두 성립해야 발생할 수 있다. 예방은 이 중 **하나 이상이 성립하지 않도록** 자원 사용 규칙을 정하여 발생 가능성을 사전에 없애는 방법이다.

② X: 한 자원을 한 시점에 하나의 프로세스만 사용하게 하는 것은 **상호 배제라는 발생 필요조건**이다. 예를 들어 두 프로세스가 서로 다른 자원을 하나씩 독점한 채 상대가 가진 자원을 기다리면 교착상태가 발생할 수 있다. 따라서 상호 배제를 보장하는 것만으로 교착상태가 해결되지는 않는다. 예방하려면 공유 가능한 자원에서는 상호 배제를 없애거나, 점유 대기 등 다른 필요조건을 성립하지 않게 해야 한다.

③ O: 교착상태 탐지 후 회복에 해당한다. 교착상태가 발생했는지 검사한 뒤, 관련 프로세스를 종료하거나 자원을 선점하여 기다림의 고리를 끊는다. 프로세스를 종료하면 점유하던 자원이 회수된다. 여기서 자원의 제거는 장치를 물리적으로 없앤다는 뜻이 아니라 **교착상태에 묶인 자원을 회수**하는 것으로 이해한다.

④ O: 교착상태 회피에 해당한다. 자원 요청을 받았을 때 이를 허용해도 모든 프로세스가 순서대로 실행을 마칠 수 있는 **안전 상태가 유지되는지** 확인한다. 안전 상태를 보장할 수 없는 요청은 보류하여 교착상태에 빠질 위험을 피한다. 발생 조건 자체를 사전에 없애는 예방과 달리, 회피는 자원 할당 여부를 판단하여 위험한 상태로의 진입을 막는다.

NoteBase

문 13. 다음 Java 프로그램은 3의 배수를 제외한 1부터 10까지 정수의 누적 합이 10을 초과하는 최초 시점에서의 합을 출력하는 프로그램이다. ㉠과 ㉡에 들어가는 내용으로 적절한 것은?

```
public class JavaApplication {
    public static void main(String[] args) {
        int i = 0, sum = 0;
        while(i < 10) {
            i++;
            if(i % 3 == 0) ㉠_____;
            if(sum > 10)   ㉡_____;
            sum += i;
        }
        System.out.println("sum=" + sum);
    }
}
```

① ㉠: break
㉡: goto

② ㉠: continue
㉡: break

③ ㉠: final
㉡: continue

④ ㉠: return
㉡: break

정답 2번

간단한 해설

3의 배수일 때는 **continue**로 이번 반복의 덧셈을 건너뛰고, 누적 합이 10을 넘으면 **break**로 while문을 종료해야 한다.

NoteBase

빈칸별 해설

㉠: continue. i % 3 == 0은 i를 3으로 나눈 나머지가 0, 즉 3의 배수라는 뜻이다. continue는 **현재 반복의 나머지 부분을 건너뛰고** while문의 조건을 다시 검사하게 한다. 따라서 아래의 sum += i가 실행되지 않아 3의 배수가 합산에서 제외된다. i++는 이 검사보다 앞에 있으므로 다음 반복에서도 i는 계속 증가한다. 여기서 break를 쓰면 첫 3의 배수인 3에서 반복 전체가 끝나고, return을 쓰면 main 메서드가 끝나 반복문 뒤의 출력문도 실행되지 않는다.

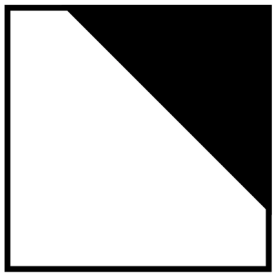
㉡: break. sum > 10이면 더 이상 수를 더하지 않고 **while문을 빠져나와** 뒤의 출력문을 실행해야 한다. break는 자신을 둘러싼 반복문을 종료한다. continue를 쓰면 이번 반복만 건너뛰고 반복 자체는 계속되므로 종료 목적에 맞지 않는다. 실제로 1과 2를 더하면 sum은 3이고, 3은 건너뛴다. 이어 4를 더하면 7, 5를 더하면 12가 된다. sum > 10 검사는 덧셈보다 앞에 있으므로 5를 더한 직후에는 종료하지 않는다. 다음 i=6에서는 continue로 건너뛰고, i=7에서 sum > 10이 참이 되어 break로 종료한다. 이때 출력되는 내용은 sum=12이다.

㉠에 continue, ㉡에 break를 넣는 조합은 ②에 해당한다.

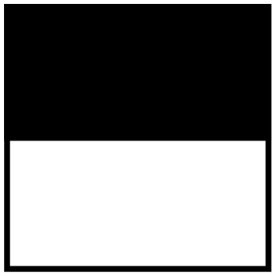
NoteBase

문 14. 다음과 같은 압축되지 않은 비트맵 형식의 이미지를 RLE (Run Length Encoding) 방식을 이용하여 압축했을 때 압축률이 가장 작은 것은? (단, 모든 이미지의 가로와 세로의 길이는 동일하고, 가로 방향 우선으로 픽셀을 읽어 처리한다)

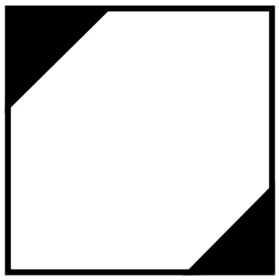
①



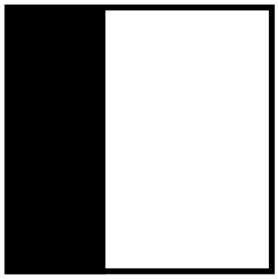
②



③



④



정답 4번

NoteBase

간단한 해설

RLE는 **연속된 같은 색의 픽셀을 색과 반복 횟수로 묶어** 저장한다. ④는 가로로 읽을 때 모든 행의 내부에서 검은색과 흰색이 나뉘므로, 기록할 구간이 많아 **압축 효과가 가장 작다**.

선지별 해설

RLE는 같은 값이 연속되는 구간인 '런(run)'마다 픽셀 값과 반복 횟수를 기록하는 방식이다. 예를 들어 검은 픽셀 100개가 이어지면 검은색과 반복 횟수 100으로 나타낼 수 있다. 같은 크기의 흑백 이미지에서는 가로 방향으로 읽을 때 색이 바뀌는 지점이 많을수록 런이 잘게 나뉘어 저장할 정보가 많아진다. 이 문제는 구체적인 압축 파일 크기를 계산하는 대신, 각 그림에서 같은 색이 이어지는 구간을 비교한다.

①: 오른쪽 위의 검은 영역이 대각선으로 내려온다. 대각선이 지나는 행은 내부가 흰색 구간과 검은색 구간으로 나뉘지만, 대각선 아래쪽에는 **내부가 흰색으로만 이어지는 행**이 있다. 따라서 모든 행의 내부가 두 색으로 나뉘는 ④보다 런을 적게 기록할 수 있다.

②: 검은 영역과 흰 영역이 **위아래로 나뉘어** 있다. 가로로 읽으면 위쪽 행의 내부는 검은색으로, 아래쪽 행의 내부는 흰색으로 길게 이어진다. 가로로 이어지는 같은 색을 크게 묶을 수 있으므로 ④보다 압축 효과가 크다.

③: 왼쪽 위와 오른쪽 아래에 검은 삼각형 영역이 있다. 위쪽과 아래쪽의 일부 행에서는 내부의 색이 바뀌지만, **가운데에는 내부가 흰색으로만 이어지는 행**이 있다. 따라서 모든 행의 내부가 두 색으로 나뉘는 ④보다 런을 적게 기록할 수 있다.

④: 검은 영역과 흰 영역이 **좌우로 나뉘어** 있다. 가로로 읽으면 모든 행의 내부에서 검은색 구간 다음에 흰색 구간이 나타나므로 두 구간을 따로 기록해야 한다. 검은색이 세로로 길게 이어져 있어도 가로 우선으로 읽기 때문에 그 세로 영역 전체를 하나의 런으로 묶을 수 없다.

따라서 가로 우선으로 읽을 때 기록해야 할 런이 가장 많은 ④가 저장 공간을 줄이는 효과, 즉 이 문항에서 묻는 압축률이 가장 작다.

NoteBase

문 15. 다중 스레드(Multithread)에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 스레드는 프로세스보다 더 큰 CPU의 실행 단위이다.
ㄴ. 단일 CPU 컴퓨터에서 작업을 수행하는 스레드들은 CPU 자원을 공유한다.
ㄷ. 스레드는 프로세스와 마찬가지로 독립적인 PC(Program Counter)를 가진다.
ㄹ. 프로세스 간의 문맥교환은 동일 프로세스에 있는 스레드 간의 문맥교환에 비해 비용면에서 효과적이다.

① ㄱ, ㄴ
② ㄱ, ㄹ
③ ㄴ, ㄷ
④ ㄴ, ㄹ

정답 3번

간단한 해설

스레드들은 **CPU 사용 시간을 나누어 쓰지만**, 각자의 실행 위치를 나타내는 **PC는 따로 가진다**. 같은 프로세스의 스레드끼리는 주소 공간을 공유하므로, 문맥교환 비용도 일반적으로 프로세스 간 전환보다 작다.

보기별 해설

ㄱ. X: 스레드는 프로세스 안에서 실행 흐름을 이루는 **기본 실행 단위**이다. 하나의 프로세스에 여러 스레드가 존재할 수 있으므로, 프로세스보다 더 큰 실행 단위라는 설명은 반대이다.

ㄴ. O: 단일 CPU에서 여러 스레드가 작업을 수행할 때는 **CPU 사용 시간을 나누어 쓴다**. 한 스레드가 CPU를 사용하다가 다른 스레드로 실행이 전환되는 방식으로 CPU 자원을 공유한다.

ㄷ. O: PC(Program Counter)는 다음에 실행할 명령어의 주소를 나타낸다. 스레드마다 실행 중인 위치가 다를 수 있으므로 **각자의 PC와 레지스터 상태**를 가진다. 같은 프로세스의 스레드가 코드와 데이터를 공유하더라도 실행 위치까지 같아야 하는 것은 아니다.

ㄹ. X: 문맥교환은 실행 중인 작업의 상태를 저장하고 다음 작업의 상태를 복원하는 과정이다. 서로 다른 프로세스로 전환할 때는 일반적으로 주소 공간 전환에 따른 추가 처리가 필요하다. 반면 동일 프로세스의 스레드들은 주소 공간을 공유하므로 **동일 프로세스 내 스레드 간 문맥교환이 일반적으로 더 적은 비용**으로 이루어진다.

옳은 보기는 ㄴ, ㄷ이므로 ③의 조합과 일치한다.

NoteBase

문 16. 구매 방법에 따른 소프트웨어 분류에 대한 설명으로 옳은 것은?

① 프리웨어(Freeware)는 라이선스 없이 무료로 배포되어, 영리목적 기관에서도 자유롭게 배포할 수 있는 소프트웨어이다.

② 라이트웨어(Liteware)는 상용 소프트웨어의 일부 기능만을 사용할 수 있도록 하여, 낮은 가격에 판매되는 소프트웨어이다.

③ 오픈소스 소프트웨어(Open Source Software)는 프로그램 소스가 공개되어 있으나, 저작권자의 동의 없는 임의 수정은 불가능하다.

④ 셰어웨어(Shareware)는 시범적으로 사용자에게 무료로 제공한 후 일정 기간이 지나면, 유용성에 따라서 구매하도록 하는 소프트웨어이다.

정답 4번

간단한 해설

셰어웨어는 **구매 전에 무료로 시험 사용**하여 유용성을 판단하게 하는 소프트웨어이다. 정해진 시험 사용 기간이 지난 뒤 계속 사용하려면 라이선스 비용을 지불한다.

선지별 해설

① X: 프리웨어는 사용료 없이 제공되는 소프트웨어이지만, **무료라는 것이 라이선스가 없다는 뜻은 아니다**. 저작권자가 정한 이용·배포 조건은 유지되므로, 영리목적 기관의 이용이나 재배포가 허용되는지는 해당 라이선스를 확인해야 한다.

② X: 라이트웨어는 상용 소프트웨어의 일부 기능을 제한하여 **무료로 제공하는 버전**을 말한다. 제한된 기능을 써 보게 하여 완전한 기능을 갖춘 유료 버전의 구매를 유도하므로, '낮은 가격에 판매된다'는 부분이 잘못되었다.

③ X: 오픈소스 소프트웨어는 소스 코드가 공개될 뿐 아니라, 라이선스 조건에 따라 **수정과 재배포도 허용**된다. 허용된 범위에서 수정할 때 저작권자의 개별 동의를 다시 받을 필요는 없다. 다만 수정·배포 시 지켜야 할 라이선스 조건까지 없어지는 것은 아니다.

④ O: 셰어웨어는 사용자가 구매 전에 프로그램의 기능과 유용성을 평가하도록 무료 시험 사용을 허용한다. 선지처럼 시험 사용 기간을 둔 경우, 그 기간이 지난 뒤 **계속 사용하려면 비용을 지불**해야 한다. 무료 시험 사용은 구매 여부를 판단할 기회이며, 영구적인 무료 사용 허락과는 구별된다.

NoteBase

문 17. 프로세스 상태 전이에서 준비(Ready) 상태로 전이되는 상황만을 모두 고르면? (단, 동일한 우선순위의 프로세스가 준비 상태로 한 개 이상 대기하고 있다)

ㄱ. 실행 상태에 있는 프로세스가 우선순위가 높은 프로세스에 의해 선점되었을 때
ㄴ. 블록된(Blocked) 상태에 있는 프로세스가 요청한 입출력 작업이 완료되었을 때
ㄷ. 실행 상태에 있는 프로세스가 작업을 마치지 못하고 시간 할당량을 다 썼을 때

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

정답 4번

간단한 해설

준비 상태는 **실행에 필요한 조건을 갖추고 CPU 배정을 기다리는 상태**이다. CPU를 빼앗기거나 시간 할당량을 다 쓴 프로세스는 다시 CPU를 기다리고, 입출력이 끝난 프로세스도 대기 원인이 해소되어 준비 상태로 이동한다.

보기별 해설

준비(Ready) 상태의 프로세스는 CPU를 배정받으면 실행할 수 있다. 실행(Running) 상태는 CPU를 사용 중인 상태이고, 블록(Blocked) 상태는 입출력 완료 등 특정 사건을 기다려 CPU를 배정받아도 당장 실행할 수 없는 상태이다.

ㄱ. O: 우선순위가 높은 다른 프로세스에 의해 선점되면 현재 프로세스는 CPU를 내놓는다. 실행에 필요한 다른 조건은 갖추고 있으므로 **실행 → 준비**로 전이되어 다음 CPU 배정을 기다린다.

ㄴ. O: 요청한 입출력 작업이 끝나면 블록 상태에 머물게 한 대기 원인이 해소된다. 다만 입출력 완료 자체가 CPU 배정을 뜻하지는 않으므로 **블록 → 준비**로 전이하고, 이후 스케줄러에 의해 CPU를 배정받으면 실행 상태가 된다.

ㄷ. O: 작업이 끝나지 않았어도 시간 할당량을 모두 쓰면 CPU를 다른 프로세스에 넘기고 **실행 → 준비**로 전이한다. 문제에서 동일한 우선순위의 프로세스가 준비 상태로 대기하고 있으므로, 기존 프로세스는 CPU를 계속 독점하지 않고 다음 실행 차례를 기다린다. 작업을 마친 것이 아니므로 종료 상태로 전이하지 않는다.

ㄱ, ㄴ, ㄷ은 모두 준비 상태로 전이되는 상황이므로 이를 모두 포함한 ④에 해당한다.

NoteBase

문 18. CSMA/CD(Carrier Sense Multiple Access with Collision Detection)에 대한 설명으로 옳은 것만을 고르면?

ㄱ. 버스형 토폴로지에 많이 사용한다.
ㄴ. 데이터 전송 시간 및 충돌에 의한 데이터 지연 시간을 정확히 예측할 수 있다.
ㄷ. 데이터를 전송하기 전에 통신회선의 사용 여부를 확인하고 전송하는 방식이다.
ㄹ. 전송할 데이터가 없을 때에도 토큰이 통신회선을 회전하면서 점유하는 단점이 있다.

① ㄱ, ㄷ
② ㄱ, ㄹ
③ ㄴ, ㄷ
④ ㄴ, ㄹ

정답 1번

간단한 해설

CSMA/CD는 여러 장치가 공유하는 회선에서 **전송 전에 회선의 사용 여부를 확인**하는 방식으로, 버스형 이더넷에 사용된다. 충돌이 발생하면 임의의 시간을 기다렸다가 재전송하므로 지연 시간을 정확히 예측할 수 없다.

NoteBase

보기별 해설

ㄱ. O: 버스형 토폴로지는 여러 장치가 하나의 공통 전송 매체를 공유한다. 둘 이상의 장치가 동시에 전송하면 충돌할 수 있으므로, **공유 회선의 사용 여부와 충돌을 감지**하는 CSMA/CD가 전통적인 버스형 이더넷에 사용된다.

ㄴ. X: CSMA/CD에서는 장치들이 회선 사용을 경쟁하며, 충돌이 발생하면 전송을 중단하고 **임의의 대기 시간 후 재전송**한다. 충돌 여부와 재시도에 따른 대기 시간이 달라지므로 데이터 전송 시간과 충돌에 의한 지연 시간을 정확히 예측할 수 없다.

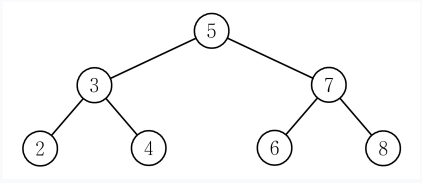
ㄷ. O: CSMA의 'Carrier Sense'는 **전송 전에 회선이 사용 중인지 감지**한다는 뜻이다. 회선이 비어 있으면 전송하고, 사용 중이면 기다린다. 다만 신호가 전파되는 데 시간이 걸리므로 서로 다른 장치가 회선이 비었다고 판단해 거의 동시에 전송하면 충돌할 수 있으며, 이를 전송 중에 감지하는 기능이 CD이다.

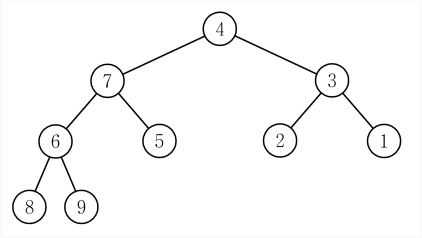
ㄹ. X: 전송할 데이터가 없어도 토큰이 순환하는 것은 **토큰 패싱 방식**의 특성이다. 토큰 패싱에서는 전송 권한을 나타내는 토큰을 가진 장치가 데이터를 전송한다. CSMA/CD는 토큰을 사용하지 않고, 회선의 사용 여부를 감지하여 전송을 시도한다.

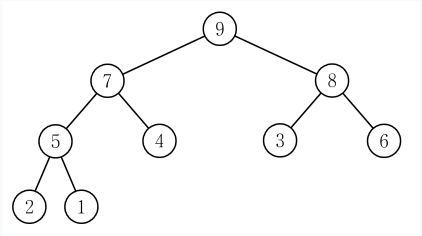
옳은 보기는 ㄱ, ㄷ이며, 이 조합은 ①에 해당한다.

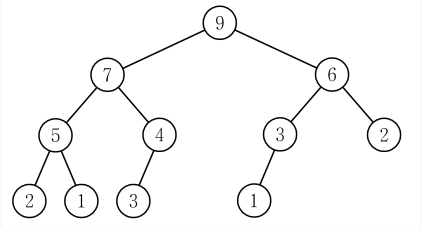
NoteBase

문 19. 최대 히프 트리(Heap Tree)로 옳은 것은?

① 

② 

③ 

④ 

정답 3번

간단한 해설

루트가 9이고 그 자식이 7과 8인 트리는 **완전이진트리**이며, 모든 연결에서 **부모의 값이 자식의 값 이상**이다. 최대 히프는 이 두 조건을 함께 만족해야 한다.

NoteBase

공통 해설

최대 히프는 **모든 부모의 값이 자식의 값 이상인 완전이진트리**이다. 완전이진트리는 마지막 레벨을 제외한 모든 레벨이 꽉 차 있고, 마지막 레벨의 노드가 왼쪽부터 빈자리 없이 채워진 이진트리이다. 따라서 값의 대소 관계와 노드가 채워진 모양을 모두 확인해야 한다.

루트가 9이고 그 자식이 7과 8인 트리에서는 9가 7·8보다 크고, 7은 자식 5·4보다 크며, 8은 자식 3·6보다 크다. 5도 자식 2·1보다 크므로 모든 부모·자식 관계가 조건을 만족한다. 마지막 레벨에는 맨 왼쪽부터 2와 1이 놓여 있어 완전이진트리 조건도 만족한다.

①과 ②는 완전이진트리 모양이지만 부모·자식의 대소 관계가 맞지 않는다. ①은 루트 5보다 오른쪽 자식 7이 크고, ②는 루트 4보다 왼쪽 자식 7이 크다. 최대 히프에서는 **자식이 부모보다 클 수 없다**.

④는 모든 부모의 값이 자식의 값 이상이지만, **마지막 레벨에 빈자리가 있다**. 노드 4의 오른쪽 자식 자리가 비어 있는데 그보다 오른쪽에 있는 노드 6의 왼쪽 자식인 노드 3에는 왼쪽 자식 1이 있다. 마지막 레벨을 왼쪽부터 빈자리 없이 채워야 하므로 완전이진트리가 아니며, 최대 히프에도 해당하지 않는다.

NoteBase

문 20. 다음 식과 논리적으로 같은 것은?

$(x + y \geq z$ AND $(x + y \geq z$ OR $x - y \leq z)$ AND $x - y > z)$ OR $x + y < z$

① $x + y < z$

② $x - y > z$

③ $x + y \geq z$ OR $x - y \leq z$

④ $x + y < z$ OR $x - y > z$

정답 4번

간단한 해설

괄호 안에서 $x + y \geq z$가 참이면 그 조건을 포함한 OR 식도 참이므로, **중복된 OR 조건을 없앨 수 있다**. 남은 식을 정리하면 $x + y < z$ OR $x - y > z$가 된다.

계산 과정

1. 조건을 기호로 나타내기: A를 $x + y \geq z$, B를 $x - y > z$라고 두자. 그러면 $x + y < z$는 A, $x - y \leq z$는 B이다. AND는 두 조건이 모두 참일 때, OR는 적어도 하나가 참일 때 참이다.

2. 괄호 안의 중복 조건 없애기: 괄호 안에는 A, A B, B가 AND로 연결되어 있다. **흡수법칙**에 따라 A(A B)=A이다. A가 참이면 OR 식도 참이고, A가 거짓이면 AND 식 전체가 거짓이므로 이 등식이 성립한다. 따라서 괄호 안은 A B로, 전체 식은 (A B) A로 줄어든다.

3. 전체 식 정리하기: 분배법칙을 적용하면 (A B) A는 (A A)(B A)와 같다. 여기서 **한 조건과 그 부정을 OR로 연결하면 항상 참**이므로 A A는 참이다. 참과 다른 조건을 AND로 연결하면 그 조건의 값이 그대로 남으므로, 전체 식은 B A, 즉 A B가 된다.

4. 원래 조건으로 되돌리기: A는 $x + y < z$, B는 $x - y > z$이다. 따라서 주어진 식은 $x + y < z$ OR $x - y > z$와 논리적으로 같으며, 이는 ④의 식이다.

NoteBase