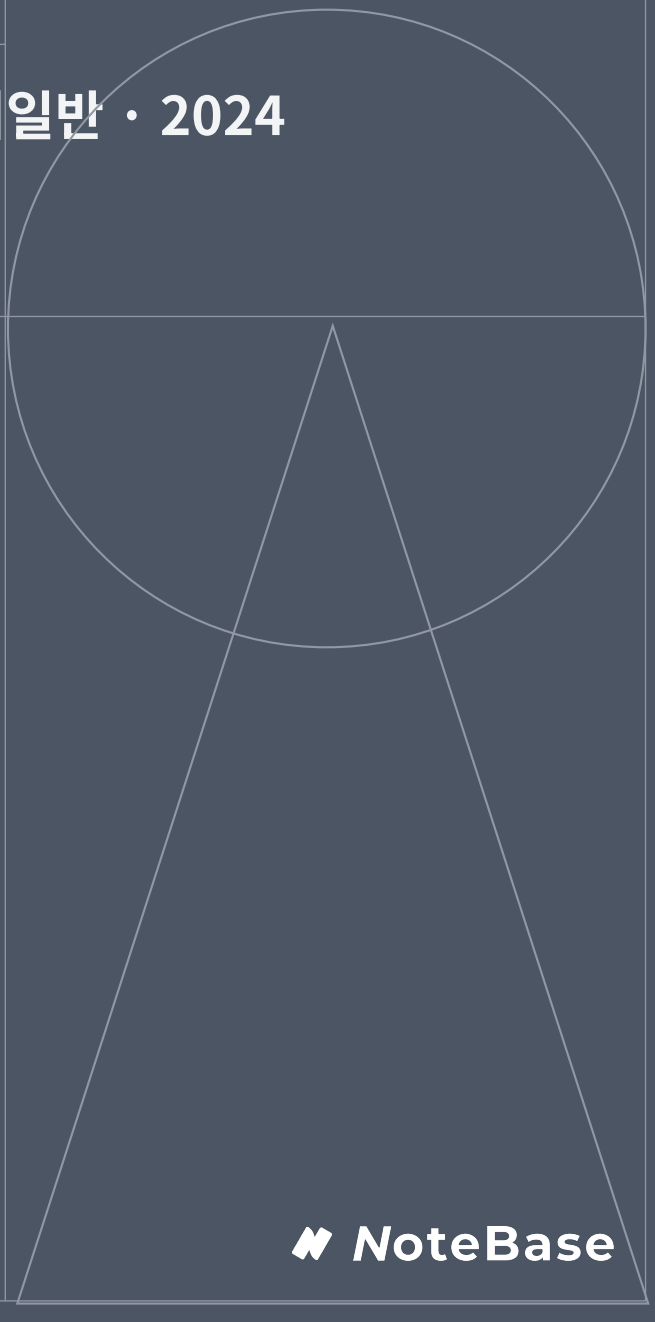
기출OX 해설집

공무원/지방직/9급 · 컴퓨터일반 · 2024



NoteBase

기출OX 해설집

공무원/지방직/9급 · 컴퓨터일반 · 2024

공무원/지방직/9급 · 컴퓨터일반 · 2024

문 1. 이미지 파일 형식에 해당하지 않는 것은?

① WAV

② BMP

③ TIFF

④ JPEG

정답 1번

간단한 해설

WAV는 이미지가 아니라 음성·음악 등 **소리를 저장하는 오디오 파일 형식**이다.

선지별 해설

① X: WAV는 **소리를 저장하는 오디오 파일 형식**이다. 음성이나 음악 등의 데이터를 담으며, 이미지 파일 형식에 해당하지 않는다.

② O: BMP는 그림을 **픽셀 단위의 비트맵 데이터**로 저장하는 이미지 파일 형식이다. 각 픽셀의 색상 정보로 이미지를 표현한다.

③ O: TIFF는 스캔한 문서나 사진 등의 이미지를 저장하는 파일 형식이다. **무손실 압축을 지원**하여 인쇄·출판처럼 화질 보존이 필요한 작업에도 사용된다.

④ O: JPEG는 사진과 같은 이미지를 저장하는 파일 형식이다. 일반적으로 **손실 압축**을 이용하여 파일 크기를 줄이므로 압축률을 높이면 화질이 저하될 수 있다.

NoteBase

문 2. DDL(Data Definition Language) 명령어에 해당하지 않는 것은?

① ALTER

② DROP

③ SELECT

④ CREATE

정답 3번

간단한 해설

SELECT는 저장된 **데이터를 조회**하는 명령어이다. 테이블 같은 데이터베이스 객체의 구조를 정의·변경·삭제하는 **DDL에 해당하지 않는다**.

선지별 해설

DDL은 데이터 정의어로, 데이터베이스 객체의 구조를 정의하는 데 사용하는 명령어이다. 각 선지가 DDL 명령어에 해당하는지를 판단하면 다음과 같다.

① O: ALTER는 이미 존재하는 데이터베이스 객체의 **구조를 변경**하는 DDL 명령어이다. 예를 들어 테이블에 열을 추가하거나 열의 자료형을 변경할 때 사용한다.

② O: DROP은 테이블 등 **데이터베이스 객체 자체를 삭제**하는 DDL 명령어이다. 테이블의 일부 행을 지우는 DELETE와 달리 테이블 자체를 제거한다.

③ X: SELECT는 테이블에서 조건에 맞는 데이터를 조회하는 명령어로, **객체의 구조를 정의하는 DDL이 아니다**. DDL 명령어로 바꾸려면 CREATE, ALTER, DROP 등을 제시해야 한다.

④ O: CREATE는 테이블 등 **데이터베이스 객체를 생성**하는 DDL 명령어이다. 예를 들어 테이블을 만들면서 열 이름과 자료형 등을 정의한다.

NoteBase

문 3. 8진수 $543_{(8)}$과 10진수 $124_{(10)}$의 합을 8진수로 표현한 것은?

① $626_{(8)}$

② $637_{(8)}$

③ $726_{(8)}$

④ $737_{(8)}$

정답 4번

간단한 해설

서로 다른 진수의 수는 **진수를 통일한 뒤** 더한다. $543_{(8)}$은 10진수 355이므로 합은 479이고, 이를 8진수로 바꾸면 $737_{(8)}$이다.

계산 과정

1. 8진수를 10진수로 변환한다. 8진수는 **오른쪽부터 자릿값이 1, 8, 64**이므로 $543_{(8)}$의 값은 $5 \cdot 64 + 4 \cdot 8 + 3 = 355$이다.

2. 같은 진수로 더한다. 두 수를 모두 10진수로 나타냈으므로 합은 355 + 124 = 479이다. 이 479는 합의 10진수 표현이다.

3. 합을 8진수로 변환한다. 479를 8로 나누면 몫 59, 나머지 7이고, 59를 8로 나누면 몫 7, 나머지 3이다. 마지막으로 7을 8로 나누면 몫 0, 나머지 7이다. 각 나머지는 낮은 자리부터 구해지므로 **나머지를 마지막에 구한 것부터 거꾸로** 읽으면 737이 된다.

합의 8진수 표현은 $737_{(8)}$이며, ④와 일치한다.

NoteBase

문 4. OSI 모델에서 데이터 링크 계층의 프로토콜 데이터 단위(protocol data unit)는?

① 비트(bit)

② 패킷(packet)

③ 프레임(frame)

④ 세그먼트(segment)

정답 3번

간단한 해설

데이터 링크 계층은 같은 링크로 연결된 장치 사이에서 데이터를 전달하며, 이때 사용하는 프로토콜 데이터 단위가 **프레임(frame)**이다.

개념 구분

프로토콜 데이터 단위(PDU)는 각 계층의 프로토콜이 처리하는 데이터 묶음이다. 데이터 링크 계층은 같은 링크로 연결된 장치 사이에서 데이터를 전달하며, 이때 사용하는 PDU가 **프레임(frame)**이다. 상위 계층에서 받은 데이터에 링크 전송에 필요한 제어 정보를 붙여 프레임을 구성한다.

다른 데이터 단위와는 **처리하는 계층**으로 구분한다. 비트(bit)는 물리 계층, 패킷(packet)은 네트워크 계층의 데이터 단위이며, 세그먼트(segment)는 전송 계층에서 TCP가 사용하는 데이터 단위이다.

NoteBase

문 5. 이메일 송신 또는 수신을 위한 프로토콜에 해당하지 않는 것은?

① POP3

② SMTP

③ FTP

④ IMAP

정답 3번

간단한 해설

FTP는 이메일 송수신이 아니라 **파일 전송**을 위한 프로토콜이다. 이메일을 보낼 때는 SMTP를, 메일 서버의 메시지를 가져오거나 조회할 때는 POP3 또는 IMAP을 사용한다.

선지별 해설

① O: POP3는 메일 서버에 도착한 이메일을 **사용자 기기로 내려받는** 수신용 프로토콜이다. 서버에 저장된 메시지를 가져와 읽을 수 있게 한다.

② O: SMTP는 이메일을 **발송하고 메일 서버 사이에서 전달하는** 프로토콜이다. 사용자의 메일 프로그램이 발신 메일 서버로 메시지를 보내거나, 발신 메일 서버가 수신 측 메일 서버로 메시지를 전달할 때 사용한다.

③ X: FTP는 클라이언트와 서버 사이에서 **파일을 업로드하거나 다운로드하는** 파일 전송 프로토콜이다. 이메일 송수신용 프로토콜로 분류하지 않으며, 이메일 송신에는 SMTP, 수신에는 POP3 또는 IMAP을 사용한다.

④ O: IMAP은 **서버에 저장된 이메일을 조회하고 관리하는** 수신용 프로토콜이다. 서버의 메일함을 기준으로 메시지와 읽음 상태 등을 관리하므로 여러 기기에서 같은 메일함을 이용하기에 적합하다.

NoteBase

문 6. UML 버전 2.0에서 구조 다이어그램에 해당하는 것만을 모두 고르면?

ㄱ. 활동 다이어그램
ㄴ. 클래스 다이어그램
ㄷ. 컴포넌트 다이어그램
ㄹ. 시퀀스 다이어그램

① ㄱ, ㄴ
② ㄱ, ㄹ
③ ㄴ, ㄷ
④ ㄷ, ㄹ

정답 3번

간단한 해설

클래스 다이어그램과 컴포넌트 다이어그램은 시스템을 이루는 요소와 그 관계를 나타내는 **구조 다이어그램**이다. 활동의 흐름이나 메시지의 시간적 순서를 나타내는 다이어그램은 행위 다이어그램에 속한다.

보기별 해설

ㄱ. 활동 다이어그램 — 행위 다이어그램: 작업의 수행 흐름과 분기·병렬 처리 등을 나타낸다. 시스템의 구성 요소 자체보다 **활동이 진행되는 흐름**을 표현한다.

ㄴ. 클래스 다이어그램 — 구조 다이어그램: 클래스의 속성과 연산, 클래스 사이의 연관·상속 관계 등을 나타낸다. 연산을 표시하더라도 그 실행 과정을 시간순으로 추적하는 것이 아니라 **클래스의 구성과 관계**를 표현한다.

ㄷ. 컴포넌트 다이어그램 — 구조 다이어그램: 시스템을 구성하는 컴포넌트와 이들이 제공하거나 요구하는 인터페이스, 컴포넌트 사이의 의존 관계 등을 나타낸다. 소프트웨어가 **어떤 구성 요소로 나뉘고 서로 연결되는지**를 표현한다.

ㄹ. 시퀀스 다이어그램 — 행위 다이어그램의 하위 범주인 상호작용 다이어그램: 객체 등 참여자 사이에 주고받는 메시지를 **시간적 순서**에 따라 나타낸다. 구성 요소의 정적 관계보다 실행 중의 상호작용에 초점을 둔다.

구조 다이어그램에 해당하는 보기는 ㄴ과 ㄷ이며, 이 조합은 ③에 해당한다.

NoteBase

문 7. 블랙박스 테스트 기법에 해당하는 것은?

① 조건 커버리지(condition coverage)

② 기본 경로 테스트(basis path test)

③ 문장 커버리지(statement coverage)

④ 동등 분할(equivalence partitioning)

정답 4번

간단한 해설

동등 분할은 입력값을 **같은 방식으로 처리될 것으로 예상되는 집합**으로 나누고 대표값을 골라 검사한다. 코드 내부의 실행 구조가 아니라 **입력 조건과 기대 결과**를 기준으로 하므로 블랙박스 테스트에 해당한다.

개념 구분

블랙박스 테스트는 프로그램의 내부 코드 구조를 기준으로 삼지 않고, **명세에 따른 입력과 출력**으로 기능을 검사한다. 화이트박스 테스트는 코드 내부의 문장·조건·제어 흐름을 기준으로 검사할 대상을 정한다.

동등 분할(equivalence partitioning): 입력 영역을 동일하게 처리될 것으로 예상되는 동등 클래스들로 나누고, 각 클래스에서 대표값을 선택한다. 예를 들어 1부터 100까지의 정수만 허용하는 입력이라면, 정수 입력을 '0 이하', '1 이상 100 이하', '101 이상'으로 나눌 수 있다. 각 영역에서 대표값을 골라 정상 입력과 잘못된 입력에 대한 처리를 확인한다. 이 구분은 **입력 명세에서 도출**하므로 블랙박스 테스트 기법이다.

나머지 기법은 내부 코드 구조를 검사 기준으로 삼는 화이트박스 테스트에 속한다. 조건 커버리지는 개별 조건이 참과 거짓이 되는 경우를 검사하고, 문장 커버리지는 실행 가능한 각 문장의 실행 여부를 검사한다.

기본 경로 테스트는 제어 흐름 그래프를 바탕으로 **선형 독립인 경로**를 선정하여 검사한다. 이들은 입력 영역을 동등 클래스로 나누는 동등 분할과 검사 기준이 다르다.

NoteBase

문 8. 운영체제에서 일괄 처리 시스템(batch processing system)에 대한 설명으로 옳은 것은?

① 사용자로부터 작업이 요구되는 즉시 처리한다.

② 일정량 또는 일정 기간의 작업을 모아 한꺼번에 처리한다.

③ 네트워크로 연결된 여러 대의 컴퓨터에서 작업을 분산하여 처리한다.

④ CPU 운영시간을 골고루 할당하여 여러 사용자가 순환하며 작업을 수행한다.

정답 2번

간단한 해설

일괄 처리 시스템은 **일정량 또는 일정 기간의 작업을 모아 한꺼번에 처리**한다. 요청할 때마다 즉시 처리하는 방식과 구별된다.

개념 구분

일괄 처리: 여러 작업을 모아 하나의 묶음으로 처리하는 방식이다. 예를 들어 한 달 동안의 급여 자료를 모아 월말에 급여를 계산하는 작업이 이에 해당한다. ②의 '일정량 또는 일정 기간의 작업을 모아 한꺼번에 처리'가 이 특징을 설명한다.

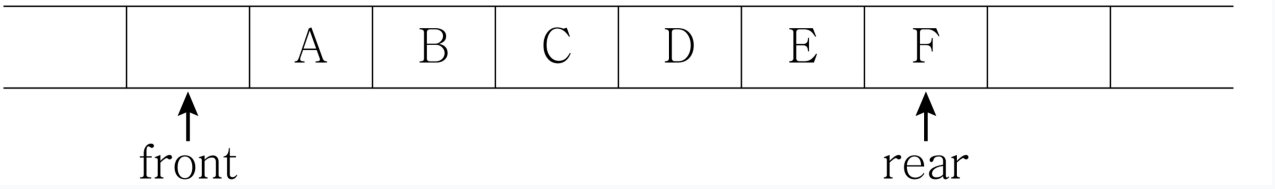
즉시 처리와의 차이: ①은 요청이 들어오면 곧바로 처리하는 방식이다. 일괄 처리는 **작업을 모은 뒤 처리**하므로, 요청 즉시 결과를 제공하는 데 초점을 두지 않는다.

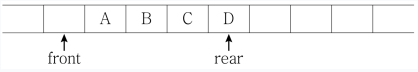
분산 처리와의 차이: ③은 네트워크로 연결된 여러 컴퓨터에 작업을 나누어 맡기는 분산 처리이다. 이는 **여러 컴퓨터에 작업을 배분**한다는 점을 설명하며, 작업을 모아 처리한다는 일괄 처리의 기준과 다르다.

시분할 처리와의 차이: ④는 CPU 사용 시간을 짧게 나누어 여러 사용자의 작업에 번갈아 할당하는 시분할 처리이다. **CPU 시간을 나누어 교대로 실행**하는 방식으로, 모아 둔 작업을 한꺼번에 처리하는 일괄 처리와 구별된다.

NoteBase

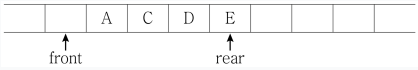
문 9. 다음은 front 다음 위치부터 rear 위치까지 유효한 원소가 들어있는 선형 큐를 보여준다. 두 개의 원소를 제거한 후 큐의 상태는?



① 

② 

③ 

④ 

정답 2번

간단한 해설

큐는 **먼저 들어온 원소부터 제거**하므로 A와 B가 차례로 제거된다. front는 B가 있던 위치로 이동하고 rear는 F 위치에 그대로 있어, 유효한 원소는 C·D·E·F이다.

공통 해설

큐는 먼저 들어온 원소를 먼저 꺼내는 선입선출(FIFO) 구조이다. 이 문제에서는 front가 첫 유효 원소의 바로 앞 위치를, rear가 마지막 유효 원소의 위치를 가리킨다. 따라서 삭제할 때마다 **front를 한 칸 앞으로 이동**하여 그 위치의 원소를 큐에서 제외하며, rear와 나머지 원소의 위치는 바뀌지 않는다.

처음 유효한 원소는 A·B·C·D·E·F이다. 첫 번째 삭제에서는 A를 제거하고 front가 A의 원래 위치로 이동한다. 두 번째 삭제에서는 B를 제거하고 front가 B의 원래 위치로 이동한다. 이때 front 바로 다음의 C부터 rear가 가리키는 F까지가 유효하므로, 남은 원소는 C·D·E·F이고 rear는 F 위치를 유지한다.

NoteBase

문 10. 운영체제의 목적으로 옳지 않은 것은?

① 신뢰도(reliability) 향상

② 처리량(throughput) 향상

③ 응답 시간(response time) 증가

④ 사용 가능도(availability) 향상

정답 3번

간단한 해설

③의 '응답 시간 증가'는 '응답 시간 **단축**'으로 고쳐야 한다. 사용자의 요청에 시스템이 반응하기까지 걸리는 시간이 짧을수록 더 신속하게 서비스를 제공할 수 있다.

선지별 해설

① O: 신뢰도는 시스템이 주어진 조건에서 오류나 고장 없이 정확하게 동작하는 정도이다. 운영체제는 오류를 감지하고 처리하며 프로그램 간의 잘못된 간섭을 막아 **안정적이고 정확한 동작**을 돕는다.

② O: 처리량은 **단위 시간에 처리하는 작업의 양**이다. 운영체제가 CPU와 입출력 장치 등 자원을 효율적으로 배분하면 같은 시간에 더 많은 작업을 처리할 수 있으므로 처리량 향상은 운영체제의 목적에 해당한다.

③ X: 응답 시간은 사용자가 요청한 뒤 시스템의 반응이 나타날 때까지 걸리는 시간이다. 이 시간이 증가하면 사용자가 더 오래 기다려야 하므로, 운영체제가 추구하는 방향은 응답 시간의 증가가 아니라 **단축**이다.

④ O: 사용 가능도는 사용자가 필요할 때 시스템과 자원을 사용할 수 있는 정도이다. 운영체제는 자원을 관리하고 장애에 대처하여 **필요할 때 서비스를 이용할 수 있도록** 하는 것을 목표로 한다.

NoteBase

문 11. 현재의 출력값이 현재의 입력값에 의해서만 결정되는 논리회로에 해당하지 않는 것은?

① 반가산기(half adder)

② 링 카운터(ring counter)

③ 멀티플렉서(multiplexer)

④ 디멀티플렉서(demultiplexer)

정답 2번

간단한 해설

링 카운터는 플립플롭에 저장된 상태를 클록에 따라 순환시키는 **순서논리회로**이다. 출력이 현재 입력뿐 아니라 **이전에 저장된 상태**에도 의존하므로 조합논리회로에 해당하지 않는다.

선지별 해설

현재 입력만으로 출력이 결정되는 회로를 조합논리회로라 한다. 순서논리회로는 기억소자를 가지므로, 현재 입력뿐 아니라 저장된 상태도 동작에 영향을 준다.

① O: 반가산기는 두 1비트 입력을 더하여 합과 자리올림을 출력하는 조합논리회로이다. 예를 들어 두 입력이 모두 1이면 합은 0, 자리올림은 1이다. 이전 연산 결과를 기억하지 않고 **현재의 두 입력만으로** 출력이 결정된다.

② X: 링 카운터는 여러 플립플롭을 연결하고 마지막 플립플롭의 출력을 처음으로 되돌려, 저장된 비트 패턴을 클록에 따라 순환시키는 **순서논리회로**이다. 예를 들어 4비트 링 카운터를 1000으로 초기화하면 클록마다 0100 → 0010 → 0001 → 1000처럼 상태가 순환할 수 있다. 다음 상태가 이전에 저장된 상태에 의존하므로, 현재 입력만으로 출력이 결정되는 조합논리회로로 분류할 수 없다.

③ O: 멀티플렉서는 여러 데이터 입력 중 하나를 선택하여 하나의 출력으로 전달하는 조합논리회로이다. **현재의 선택 신호**가 어느 입력을 전달할지 정하므로, 데이터 입력과 선택 신호가 주어지면 이전 상태와 관계없이 출력이 결정된다.

④ O: 디멀티플렉서는 하나의 데이터 입력을 여러 출력 중 선택된 출력으로 전달하는 조합논리회로이다. **현재의 선택 신호**가 전달할 출력 경로를 정하며, 출력은 현재의 데이터 입력과 선택 신호로 결정된다.

NoteBase

문 12. 캐시기억장치에 대한 설명으로 옳지 않은 것은?

① 주로 SRAM을 사용하여 구현된다.

② 주기억장치보다 용량은 작지만 접근 속도가 빠르다.

③ 성능 향상을 위해 지역성의 원리(principle of locality)를 이용한다.

④ 직접 사상(direct mapping) 방식을 사용하면, 특정 주기억장치 블록을 여러 개의 캐시기억장치 블록으로 사상할 수 있다.

정답 4번

간단한 해설

직접 사상에서는 특정 주기억장치 블록이 들어갈 캐시 위치가 **하나로 고정**된다. 여러 주기억장치 블록이 같은 캐시 위치에 대응할 수는 있지만, 하나의 주기억장치 블록을 여러 캐시 위치에 배치할 수는 없다.

선지별 해설

① O: 캐시기억장치는 빠른 접근이 필요하므로 주로 **SRAM**으로 구현한다. SRAM은 전원이 공급되는 동안 주기적인 재충전 없이 데이터를 유지하며, 주기억장치에 주로 쓰이는 DRAM보다 접근 속도가 빠르다.

② O: 캐시기억장치는 주기억장치보다 **용량은 작고 접근 속도는 빠르다**. 빠른 SRAM은 비트당 비용이 높으므로 비교적 작은 용량으로 구성하고, 자주 사용하는 데이터와 명령어를 저장하여 주기억장치 접근에 드는 시간을 줄인다.

③ O: 지역성은 프로그램이 **최근 접근한 위치나 그 주변 위치에 다시 접근하는 경향**이다. 최근 사용한 데이터나 명령어를 다시 사용하는 시간적 지역성과, 접근한 주소의 인접 주소를 사용하는 공간적 지역성이 있다. 캐시는 이러한 경향을 이용해 다시 필요할 가능성이 큰 내용을 보관하여 평균 메모리 접근 시간을 줄인다.

④ X: 직접 사상은 각 주기억장치 블록이 들어갈 수 있는 캐시 블록을 **오직 하나로 정하는 방식**이다. 따라서 '여러 개의 캐시기억장치 블록으로 사상할 수 있다'는 '정해진 하나의 캐시기억장치 블록으로 사상된다'로 고쳐야 한다. 예를 들어 두 주기억장치 블록 A와 B가 같은 캐시 위치에 대응하면, A를 보관한 상태에서 B를 그 위치에 가져올 때 A가 교체된다.

NoteBase

문 13. 데이터 크기에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 1바이트(byte)는 8비트이다.
ㄴ. 1니블(nibble)은 2비트이다.
ㄷ. 워드(word) 크기는 컴퓨터 시스템에 따라 다를 수 있다.

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

정답 2번

간단한 해설

1니블은 1바이트의 절반인 **4비트**이므로, '2비트'라는 설명은 틀리다.

보기별 해설

ㄱ. O: 비트(bit)는 0 또는 1을 나타내는 데이터의 최소 단위이고, 바이트(byte)는 **8비트를 묶은 단위**이다. 따라서 1바이트는 8비트이다.

ㄴ. X: 니블(nibble)은 **4비트**를 묶은 단위이다. 1바이트가 8비트이므로 1바이트는 2니블에 해당한다. '1니블은 2비트'를 '1니블은 4비트'로 고쳐야 한다.

ㄷ. O: 워드(word)는 CPU가 한 번에 처리하는 기본 데이터 단위로, 그 크기는 **컴퓨터 시스템의 구조에 따라 달라질 수 있다**. 예를 들어 워드 크기가 32비트인 시스템과 64비트인 시스템이 있으므로, 워드를 항상 일정한 비트 수로 고정할 수 없다.

옳은 보기는 ㄱ과 ㄷ이므로, 이 조합에 해당하는 선택지는 ②이다.

NoteBase

문 14. 다음 C 프로그램의 출력 결과는?

```
#include <stdio.h>

int repeat(int a, int b) {
    if (b == 0)
        return a;
    else if (b % 2 == 0)
        return repeat(a + a, b / 2);
    else
        return repeat(a + a, b / 2) + a;
}

int main() {
    printf("%d", repeat(3, 6));
    return 0;
}
```

① 12

② 24

③ 30

④ 42

정답 4번

간단한 해설

재귀 호출을 마친 뒤에는 **각 호출에서 더하기로 한 a값**을 되돌아오며 더해야 한다. b가 0인 호출의 반환값 24에 12와 6이 더해져 출력값은 42가 된다.

NoteBase

계산 과정

repeat의 두 매개변수 a와 b는 정수이다. b % 2는 b를 2로 나눈 나머지이므로 짝수와 홀수를 구분한다. b / 2는 정수 나눗셈이므로, 이 문제의 양수 b에서는 소수 부분을 버린 몫만 남긴다. b가 0이면 a를 반환하고, 그 외에는 a를 두 배로 하여 재귀 호출한다. 이때 **b가 홀수인 호출에서만 현재 a를 추가로 더한다**.

repeat(3, 6): b가 6으로 짝수이므로 repeat(6, 3)을 호출하고, 그 반환값을 그대로 반환한다.

repeat(6, 3): b가 3으로 홀수이다. 3을 2로 나눈 정수 몫은 1이므로 repeat(12, 1)을 호출한다. 이 호출이 끝나면 **현재 호출의 a값인 6**을 더해 반환한다.

repeat(12, 1): b가 1로 홀수이다. 1을 2로 나눈 정수 몫은 0이므로 repeat(24, 0)을 호출한다. 이 호출이 끝나면 현재 호출의 a값인 12를 더해 반환한다.

repeat(24, 0): b가 0이므로 **재귀 호출을 멈추고 a값 24를 반환**한다.

반환값을 거슬러 계산하면 repeat(12, 1)은 24 + 12 = 36, repeat(6, 3)은 36 + 6 = 42를 반환한다. repeat(3, 6)도 42를 그대로 반환하므로 printf가 출력하는 값은 42이며, ④의 값과 일치한다.

NoteBase

문 15. IPv4 주소 체계의 A 클래스 주소에서 호스트 ID의 비트 수는?

① 8

② 16

③ 24

④ 32

정답 3번

간단한 해설

A 클래스 주소는 전체 32비트 중 첫 8비트를 네트워크 ID로, 나머지 **24비트를 호스트 ID**로 사용한다.

공통 해설

IPv4 주소는 총 32비트이며, 8비트 단위인 옥텟 네 개로 구성된다. 네트워크 ID는 주소가 속한 네트워크를 식별하고, 호스트 ID는 그 네트워크 안의 개별 호스트를 구별한다.

별도로 서브넷을 나누지 않은 A 클래스의 기본 구조에서는 **첫 번째 옥텟이 네트워크 ID**이고, 뒤의 세 옥텟이 호스트 ID이다. 따라서 호스트 ID의 길이는 세 옥텟에 해당하는 24비트이다.

NoteBase

문 16. 블록체인에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 비트코인은 블록체인 기술을 기반으로 만들어진 암호화폐이다.
ㄴ. 블록체인 유형에는 퍼블릭 블록체인, 프라이빗 블록체인 등이 있다.
ㄷ. 블록체인에서 사용되는 합의 알고리즘에는 작업 증명(PoW: Proof of Work), 지분 증명(PoS: Proof of Stake) 등이 있다.

① ㄱ, ㄴ
② ㄱ, ㄷ
③ ㄴ, ㄷ
④ ㄱ, ㄴ, ㄷ

정답 4번

간단한 해설

블록체인의 합의 방식은 하나로 고정되지 않으며, **작업 증명(PoW)과 지분 증명(PoS)** 등이 사용된다. 합의 알고리즘은 참여자들이 어떤 블록을 유효한 것으로 받아들일지 결정하는 규칙이다.

보기별 해설

ㄱ. O: 비트코인은 **블록체인을 기반으로 한 암호화폐**이다. 거래 내역을 블록 단위로 묶고 이전 블록의 해시값을 통해 연결하며, 네트워크 참여자들이 거래 기록을 공유하고 검증한다.

ㄴ. O: 블록체인은 참여 허용 범위와 운영 방식에 따라 퍼블릭 블록체인, 프라이빗 블록체인 등으로 구분한다. 퍼블릭 블록체인은 원칙적으로 누구나 참여할 수 있고, 프라이빗 블록체인은 **운영 주체가 허가한 참여자**를 중심으로 구성된다.

ㄷ. O: 작업 증명(PoW)과 지분 증명(PoS)은 블록체인의 대표적인 합의 방식이다. 작업 증명은 정해진 조건을 만족하는 해시값을 찾는 **계산 작업**을 수행하도록 하고, 지분 증명은 참여자가 맡긴 **암호화폐 지분**을 바탕으로 블록 제안·검증에 참여하도록 한다. 비트코인은 작업 증명을 사용하지만, 모든 블록체인이 같은 합의 방식을 사용하는 것은 아니다.

ㄱ, ㄴ, ㄷ이 모두 옳으므로 이들을 모두 포함한 조합은 ④이다.

NoteBase

문 17. 다음은 프로세스가 준비 큐에 도착하는 시간과 프로세스를 처리하는데 필요한 실행시간을 보여준다. 비선점 SJF(Shortest Job First) 스케줄링 알고리즘을 사용한 경우, P1, P2, P3, P4 프로세스 중에서 두 번째로 실행되는 프로세스는? (단, 프로세스 간 문맥 교환에 따른 오버헤드는 무시하며, 주어진 4개의 프로세스 외에 처리할 다른 프로세스는 없다고 가정한다)

프로세스	도착시간	실행시간
P1	0	6
P2	1	4
P3	2	1
P4	3	2

① P1

② P2

③ P3

④ P4

정답 3번

간단한 해설

비선점 방식이므로 먼저 실행된 P1은 시각 6까지 계속 실행된다. 그때 준비 큐에서 기다리는 P2·P3·P4 중 **실행시간이 가장 짧은 P3**가 두 번째로 실행된다.

NoteBase

계산 과정

비선점 SJF는 CPU가 다음 프로세스를 선택할 때 **이미 준비 큐에 도착한 프로세스 중 실행시간이 가장 짧은 것**을 고른다. 한 번 실행을 시작하면 더 짧은 프로세스가 도착해도 현재 프로세스의 실행을 중단하지 않는다.

첫 번째 실행: 시각 0에는 도착시간이 0인 P1만 준비 큐에 있으므로 P1이 실행된다. P1의 실행시간은 6이고 문맥 교환 오버헤드는 무시하므로, 종료 시각은 0 + 6 = 6이다. P2·P3·P4가 각각 시각 1·2·3에 도착해도 **P1이 끝날 때까지 대기**한다.

두 번째 실행: 시각 6에는 P2·P3·P4가 모두 도착해 준비 큐에서 기다리고 있다. 이들의 실행시간은 각각 4·1·2이므로 가장 짧은 1의 실행시간을 가진 P3가 선택된다. 먼저 도착한 P2보다 P3가 앞서는 것은 도착 순서가 아니라 실행시간을 비교하기 때문이다.

따라서 두 번째로 실행되는 프로세스는 P3이며, 선택지 ③에 해당한다.

NoteBase

문 18. 다음 C 언어로 작성된 코드의 시간 복잡도는? (단, n은 임의의 양의 정수이다)

```
for (i = 0; i < n; i++)
for (j = 0; j < 500; j++)
printf("i * j = %d\n", i * j);
```

① $\Theta(n)$

② $\Theta(n^2)$

③ $\Theta(\log n)$

④ $\Theta(n\log n)$

정답 1번

간단한 해설

안쪽 반복문의 실행 횟수는 **n과 무관하게 500번으로 고정**되어 있다. 바깥 반복만 n번 늘어나므로 전체 실행 시간은 n에 비례하는 $\Theta(n)$이다.

공통 해설

시간 복잡도의 Θ 표기는 입력 크기 n이 충분히 커질 때 실행 시간이 어떤 증가율에 비례하는지를 나타낸다. 상수 배의 차이는 증가율을 바꾸지 않는다. 따라서 일정한 비용의 작업을 n번 반복하면 시간 복잡도는 $\Theta(n)$이다.

이 코드에서 바깥 반복문의 i는 0부터 n − 1까지 변하므로 본문을 n번 실행한다. 바깥 반복을 할 때마다 j는 다시 0으로 초기화되고, 0부터 499까지 변하며 안쪽 본문을 500번 실행한다. **500은 n이 커져도 변하지 않는 상수**이므로 바깥 반복 한 번에 드는 비용은 일정하다. 이 일정한 작업을 n번 수행하므로 전체 시간 복잡도는 $\Theta(n)$이다. 중첩 반복문이라도 안쪽 반복 횟수가 n에 따라 늘어나지 않으면 이차 시간이 되지 않는다.

NoteBase

문 19. 페이지 테이블(page table)을 사용하는 가상기억장치 컴퓨터 시스템에서 TLB(Translation Lookaside Buffer)에 대한 설명으로 옳은 것은?

① 페이지 테이블의 캐시로서 동작한다.

② 한 시스템 내에 여러 개가 존재할 수 없다.

③ TLB 실패(miss)가 발생할 때마다 페이지 부재(page fault)가 발생한다.

④ 물리 주소(physical address)를 가상 주소(virtual address)로 빠르게 변환하기 위한 것이다.

정답 1번

간단한 해설

TLB는 **페이지 테이블의 주소 변환 정보를 저장하는 고속 캐시**이다. 필요한 변환 정보가 TLB에 있으면 페이지 테이블을 다시 조회하지 않고 물리 주소를 구할 수 있다.

선지별 해설

① O: 페이지 테이블은 가상 페이지 번호와 물리 프레임 번호의 대응 정보를 담는다. TLB는 그중 일부를 빠르게 조회할 수 있도록 저장하는 **페이지 테이블의 캐시**이므로, 주소 변환에 필요한 페이지 테이블 접근을 줄인다.

② X: 한 시스템에 **여러 개의 TLB가 존재할 수 있다**. 예를 들어 여러 코어가 있는 프로세서에서는 각 코어가 TLB를 가질 수 있고, 명령어용과 데이터용 TLB를 나누어 둘 수도 있다.

③ X: **TLB 실패와 페이지 부재는 다르다**. TLB 실패는 필요한 주소 변환 정보가 TLB에 없다는 뜻이다. 이때 페이지 테이블을 확인하여 해당 페이지가 주기억장치에 있으면 변환 정보를 얻을 수 있으므로 페이지 부재가 발생하지 않는다. 해당 페이지가 주기억장치에 없는 경우에는 페이지 부재가 발생한다.

④ X: 주소 변환 방향이 반대이다. TLB는 **가상 주소를 물리 주소로 변환**하는 데 사용한다. 가상 주소의 페이지 번호에 대응하는 물리 프레임 번호를 찾고, 페이지 내부의 위치를 나타내는 변위는 그대로 유지하여 물리 주소를 구성한다.

NoteBase

문 20. 다음 파이썬 프로그램의 출력 결과는?

```
student_list = ['A', 'B', 'C', 'D']
student_score = ['92', '85', '77', '54']
student_grade = []
i = 0
for _ in range(len(student_score)):
   try:
      if student_score[_] >= 90:
         student_grade.append('A+')
         i+=1
      elif student_score[_] >= 80:
         student_grade.append('B+')
         i+=1
      elif student_score[_] >= 70:
         student_grade.append('C+')
         i+=1
      else:
         student_grade.append('D+')
         i+=1
   except: student_grade.append('F')
print("%s, %s" % (student_list[i], student_grade[i]))
```

① A, D+

② A, F

③ D, D+

④ D, F

정답 2번

간단한 해설

점수는 **문자열**이므로 정수 90과 비교할 때 TypeError가 발생하여 'F'가 추가된다. 예외 때문에 **i += 1이 실행되지 않아** i는 0으로 유지되므로 첫 학생과 첫 등급인 A, F가 출력된다.

NoteBase

공통 해설

파이썬에서 문자열과 정수는 >=로 크기를 비교할 수 없다. student_score의 원소는 따옴표로 둘러싸인 '92', '85', '77', '54'이므로 모두 문자열이다. 따라서 매 반복에서 처음 만나는 student_score[_] >= 90은 참·거짓을 판단하기 전에 TypeError를 발생시킨다. 숫자 모양의 문자열도 이 비교에서 자동으로 정수로 변환되지는 않는다.

try 안에서 예외가 발생하면 **그 뒤의 try 문장을 건너뛰고 except로 이동**한다. 따라서 등급 'A+', 'B+', 'C+', 'D+'를 추가하는 문장과 각 분기의 i += 1은 실행되지 않고, except의 student_grade.append('F')만 실행된다. 반복은 점수의 개수만큼 네 번 이루어지므로 student_grade는 ['F', 'F', 'F', 'F']가 된다. 이때 반복 변수 _는 0부터 3까지 바뀌지만, 별도로 초기화한 i는 계속 0이다.

마지막 print는 student_list[i]와 student_grade[i]를 차례로 출력한다. i가 0이므로 각각 첫 원소인 'A'와 'F'를 가져오며, 출력 결과는 A, F이다.

NoteBase