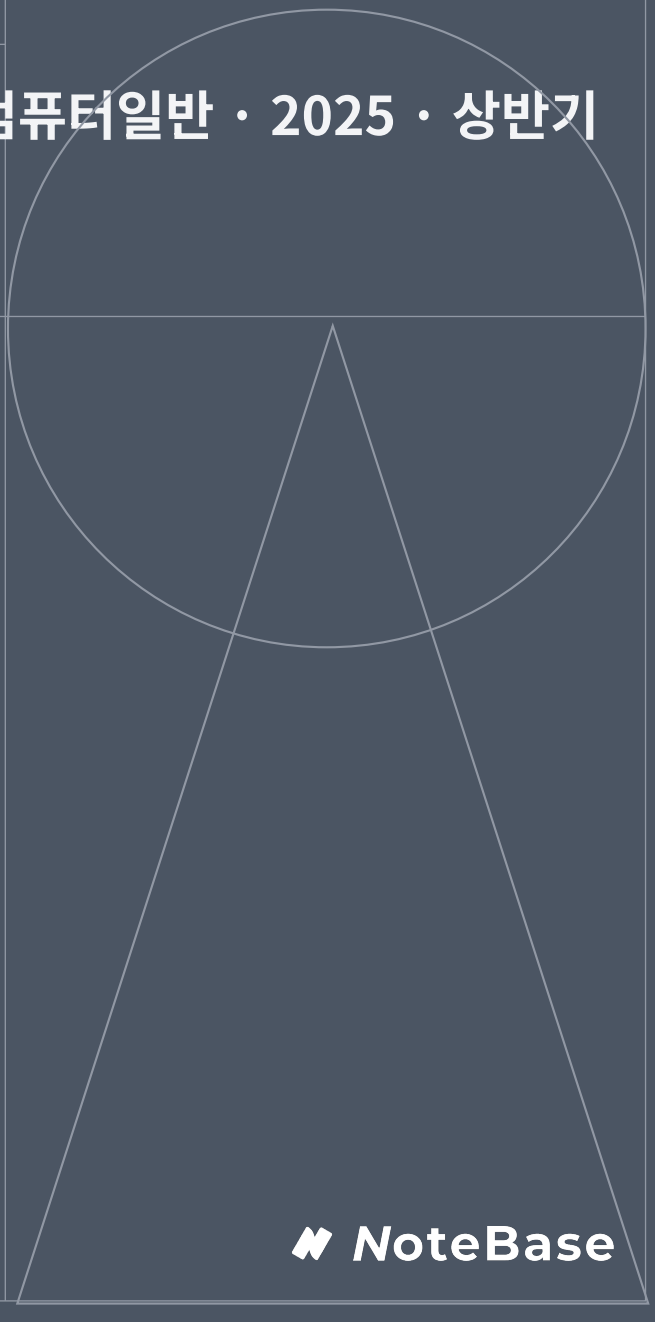
기출OX 해설집

공무원/해경/일반직 9급 · 컴퓨터일반 · 2025 · 상반기



NoteBase

기출OX 해설집

공무원/해경/일반직 9급 · 컴퓨터일반 · 2025 · 상반기

공무원/해경/일반직 9급 · 컴퓨터일반 · 2025 · 상반기

문 1. 다음 진리표에 해당하는 논리 게이트는?

입력(A)	입력(B)	출력(X)
0	0	1
0	1	1
1	0	1
1	1	0

① AND

② OR

③ NAND

④ XOR

정답 3번

출제 포인트

이 문제는 두 입력의 진리표에서 출력이 0이 되는 입력 조합을 확인하여 논리 게이트를 판별하는 문제이다.

자료 읽는 순서

1. 입력 A와 B의 네 가지 조합을 확인한다.
2. 출력 X가 0이 되는 조합을 찾는다.
3. 확인한 출력 패턴을 각 논리 게이트의 진리표와 대조한다.

핵심 근거

표에서 출력 X는 A와 B가 모두 1일 때만 0이고, 나머지 세 조합에서는 모두 1이다. 이는 AND의 출력을 반전한 NAND의 진리표이며, 논리식은 $X = \overline{A \cdot B}$이다.

NoteBase

선지별 해설

① X — AND는 A와 B가 모두 1일 때만 출력이 1이다. 제시된 표는 같은 입력에서 0을 출력하므로 AND가 아니다. 틀린 표현 ‘AND’는 ‘NAND’로 고쳐야 한다.
② X — OR는 A와 B가 모두 0일 때만 출력이 0이다. 제시된 표는 (0, 0)에서 1을 출력하므로 OR가 아니다. 틀린 표현 ‘OR’는 ‘NAND’로 고쳐야 한다.
③ O — NAND는 A와 B가 모두 1일 때만 출력이 0이고, (0, 0), (0, 1), (1, 0)에서는 모두 1이다. 제시된 표와 정확히 일치한다.
④ X — XOR는 두 입력이 서로 다를 때만 출력이 1이고, 두 입력이 같으면 0이다. 제시된 표는 (0, 0)에서도 1을 출력하므로 XOR가 아니다. 틀린 표현 ‘XOR’는 ‘NAND’로 고쳐야 한다.

오답 포인트

(1, 1)에서 출력이 0이라는 한 행만 보고 XOR와 혼동하기 쉽다. XOR는 (0, 0)에서도 0이지만, NAND는 (0, 0)에서 1이라는 차이가 있다.

다시 나오면 이렇게 판단

출력이 0인 입력 조합을 먼저 찾는다. 두 입력이 모두 1일 때만 0이면 NAND이고, 두 입력이 같을 때 0이면 XOR이다.

암기 포인트

NAND는 AND의 반전이므로 ‘둘 다 1’인 경우에만 0이다.

NoteBase

문 2. 다음 <보기>는 CPU의 명령 사이클을 구성하는 마이크로 오퍼레이션을 나타낸 것이다. 인출 사이클(fetch cycle)의 실행 순서로 옳은 것은? (단, PC는 Program Counter, IR은 Instruction Register, AC는 Accumulator, MAR은 Memory Address Register, MBR은 Memory Buffer Register를 의미한다.)

<보기>

㉠ $IR \leftarrow MBR$

㉡ $MAR \leftarrow PC$

㉢ $AC \leftarrow MBR$

㉣ $MBR \leftarrow M[MAR],\ PC \leftarrow PC + 1$

① ㉠-㉡-㉢

② ㉡-㉣-㉠

③ ㉢-㉠-㉡

④ ㉡-㉣-㉢

정답 2번

출제 포인트

이 문제는 CPU가 주기억장치에서 다음 명령어를 가져와 명령어 레지스터에 저장하는 인출 사이클의 마이크로 오퍼레이션 순서를 묻는다.

올바른 순서

1. ㉡ $MAR \leftarrow PC$ — 다음에 인출할 명령어의 주소를 PC에서 MAR로 보낸다.
2. ㉣ $MBR \leftarrow M[MAR],\ PC \leftarrow PC + 1$ — MAR이 가리키는 기억장치의 명령어를 MBR로 읽고, PC는 다음 명령어 주소로 증가시킨다.
3. ㉠ $IR \leftarrow MBR$ — MBR에 읽어 둔 명령어를 IR로 옮긴다.

왜 이 순서인가: 기억장치를 읽으려면 먼저 MAR에 주소가 있어야 하고, IR에 명령어를 넣으려면 먼저 그 명령어가 MBR에 읽혀 있어야 하기 때문이다. ㉢ $AC \leftarrow MBR$은 데이터를 누산기 AC로 옮기는 동작이므로 인출 사이클의 필수 순서가 아니다.

NoteBase

선지별 해설

① X — ㉠에서 IR로 옮길 명령어가 MBR에 아직 준비되지 않았는데 ㉠을 먼저 수행하므로 틀리다. 또한 ㉤은 인출 사이클의 필수 동작이 아니다. 틀린 순서 ㉠-㉡-㉤은 올바르게는 ㉡-㉢-㉠이다.
② O — PC의 주소를 MAR에 저장한 뒤, 해당 주소의 명령어를 MBR로 읽으면서 PC를 증가시키고, 마지막으로 MBR의 명령어를 IR에 저장한다. 따라서 ㉡-㉢-㉠이 올바르다.
③ X — ㉤은 MBR의 내용을 AC로 옮기는 동작이므로 명령어 인출 과정에 해당하지 않으며, ㉠도 기억장치 읽기 전에 수행할 수 없다. 틀린 순서 ㉤-㉠-㉡은 올바르게는 ㉡-㉢-㉠이다.
④ X — ㉡ 다음 ㉢까지는 맞지만, 마지막 ㉣은 읽어 온 명령어를 AC로 보내므로 틀리다. 틀린 마지막 단계 ㉣ *AC* ← *MBR*은 ㉠ *IR* ← *MBR*로 고쳐야 하며, 전체 순서는 ㉡-㉢-㉠이다.

오답 포인트

MBR에서 값을 받는다는 공통점 때문에 ㉠과 ㉤을 혼동하기 쉽다. 인출한 것은 명령어이므로 최종 목적지는 명령어 레지스터 IR이며, 연산 데이터를 저장하는 누산기 AC가 아니다.

다시 나오면 이렇게 판단

인출 사이클은 주소 지정 *PC* → *MAR*, 기억장치 읽기 *M*[*MAR*] → *MBR*, 명령어 저장 *MBR* → *IR*의 의존관계로 판단한다. PC 증가는 기억장치 읽기와 함께 수행될 수 있다.

암기 포인트

인출은 PC-MAR-MBR-IR 순서이며, AC는 인출의 목적지가 아니다.

NoteBase

문 3. 기억장치 접근 방법에 대한 유형 중에서 직접 접근(direct access) 방법에 해당하는 것은?

① 워드 내의 특정 비트들과 원하는 비트들을 비교하여 일치하는 단어에 접근한다. 캐시(cache)가 대표적 장치이다.

② 기억장치에 데이터가 저장되는 순서에 따라 순차적으로 접근한다. 자기 테이프가 대표적 장치이다.

③ 저장된 모든 데이터에 접근하는데 소요되는 시간이 항상 일정한 방식이다. RAM과 같은 반도체 기억장치가 대표적 장치이다.

④ 기억장소 근처로 이동한 다음 순차적 검색을 통하여 최종적으로 원하는 데이터에 접근한다. 디스크가 대표적 장치이다.

정답 4번

출제 포인트

기억장치의 순차 접근, 직접 접근, 임의 접근, 연관 접근을 데이터 위치를 찾는 방식으로 구별하는 문제이다.

핵심 개념

순차 접근은 저장 순서대로 데이터를 탐색하는 방식이다. 직접 접근은 대략적인 위치로 먼저 이동한 뒤 그 위치에서 순차적으로 원하는 데이터를 찾는 방식이다. 임의 접근은 현재 위치나 저장 순서와 관계없이 주소로 기억장소에 접근하는 방식이다. 연관 접근은 주소가 아니라 기억된 내용의 일부를 비교하여 일치하는 워드를 찾는 방식이다.

선지별 해설

① X — 특정 비트와 원하는 비트를 비교하여 일치하는 워드를 찾는 것은 직접 접근이 아니라 연관 접근이다. 틀린 표현은 '직접 접근'에 해당한다는 점이며, 올바르게는 '연관 접근 방식에 대한 설명'이다.
② X — 저장 순서에 따라 차례로 접근하고 자기 테이프를 대표 장치로 드는 것은 순차 접근이다. 틀린 표현은 '직접 접근'에 해당한다는 점이며, 올바르게는 '순차 접근 방식에 대한 설명'이다.
③ X — 저장 위치와 관계없이 거의 일정한 시간에 접근하며 RAM을 대표 장치로 드는 것은 임의 접근이다. 틀린 표현은 '직접 접근'에 해당한다는 점이며, 올바르게는 '임의 접근 방식에 대한 설명'이다.
④ O — 디스크는 읽기·쓰기 헤드를 원하는 트랙 부근으로 먼저 이동한 뒤 해당 영역에서 원하는 데이터의 위치를 찾아 접근하므로 직접 접근 방식에 해당한다.

오답 포인트

'직접 접근'이라는 이름 때문에 RAM처럼 곧바로 주소에 접근하는 임의 접근과 혼동하기 쉽다. 이 문항에서 직접 접근의 결정적 표현은 '근처로 이동한 다음 순차적 검색'이며, 대표 장치는 디스크이다.

NoteBase

다시 나오면 이렇게 판단

저장 순서대로 찾으면 순차 접근, 대략적인 위치로 이동한 뒤 다시 찾으면 직접 접근, 주소로 곧바로 찾으면 임의 접근, 내용의 비트를 비교하면 연관 접근으로 판단한다.

암기 포인트

테이프는 순차, 디스크는 직접, RAM은 임의, 내용 비교는 연관 접근이다.

NoteBase

문 4. OSI 7계층 구조에서 다음 <보기>에 해당하는 역할을 수행하는 계층은?

<보기>
◯ 하단의 네트워크 서비스와 상단의 사용자 서비스 간의 인터페이스 기능을 담당한다.
◯ 전체 메시지의 종단간(end-to-end) 전송을 수행하는 역할을 담당한다.
◯ TCP, UDP 등의 프로토콜과 관련된 계층으로 에러 복구, 흐름 제어 등을 담당하며 두 시스템 간의 신뢰성 있는 데이터 전송을 보장한다.

① 전송 계층(transport layer)
② 네트워크 계층(network layer)
③ 데이터 링크 계층(data link layer)
④ 세션 계층(session layer)

정답 1번

출제 포인트

OSI 7계층에서 종단 간 전송, 흐름 제어, 오류 복구와 TCP·UDP를 담당하는 계층을 구별하는 문제이다.

핵심 개념

전송 계층은 서로 다른 호스트에서 동작하는 응용 프로세스 사이에 종단 간 데이터 전송 서비스를 제공하는 계층이다. TCP와 UDP가 대표 프로토콜이며, 특히 TCP는 오류 복구와 흐름 제어를 통해 신뢰성 있는 전송을 제공한다.
네트워크 계층은 패킷의 논리 주소 지정과 경로 선택을, 데이터 링크 계층은 인접 노드 사이의 프레임 전송을, 세션 계층은 응용 프로그램 간 세션의 설정·유지·종료와 동기화를 담당한다.

선지별 해설

① O — 전송 계층은 상위 계층과 네트워크 계층 사이에서 종단 간 전송을 담당하며 TCP·UDP와 관련된다. 보기의 오류 복구, 흐름 제어, 신뢰성 있는 데이터 전송도 전송 계층의 대표 기능이므로 맞다.
② X — 틀린 부분은 '네트워크 계층'이다. 네트워크 계층은 IP를 바탕으로 논리 주소 지정과 패킷의 경로 선택을 담당한다. 보기처럼 전체 메시지의 종단 간 전송과 TCP·UDP를 담당하는 계층은 전송 계층이다.
③ X — 틀린 부분은 '데이터 링크 계층'이다. 데이터 링크 계층은 인접한 노드 사이에서 프레임 단위 전송, 물리 주소 지정, 오류 검출 등을 담당한다. 두 종단 시스템 사이의 TCP·UDP 기반 전송을 담당하는 계층은 전송 계층이다.
④ X — 틀린 부분은 '세션 계층'이다. 세션 계층은 응용 프로그램 사이의 대화 세션을 설정·유지·종료하고 동기점을 관리한다. 종단 간 전송과 TCP·UDP를 담당하는 계층은 전송 계층이다.

NoteBase

오답 포인트

‘종단 간 전송’을 네트워크 계층의 경로 선택과 혼동하거나, ‘오류·흐름 제어’라는 표현만 보고 데이터 링크 계층을 고르기 쉽다. 그러나 TCP·UDP와 전체 메시지의 종단 간 전송이 함께 제시되면 전송 계층을 가리킨다.

다시 나오면 이렇게 판단

계층을 묻는 문제에서는 대표 단서를 먼저 연결한다. TCP·UDP와 종단 간 전송은 전송 계층, IP·라우팅은 네트워크 계층, 프레임·MAC·인접 노드 전송은 데이터 링크 계층, 대화 세션의 설정·유지·종료는 세션 계층으로 판단한다.

암기 포인트

TCP·UDP와 종단 간 전송은 전송 계층이다.

NoteBase

문 5. 다음 <보기>에 해당하는 프로세스 상태를 올바르게 나열한 것은?

<보기>
㉠ 프로세스가 어떤 이벤트가 일어나기를 기다리는 상태
㉡ 프로세서에 할당되기를 기다리는 상태
㉢ 프로세서를 점유하여 명령어들이 실행되는 상태

① ㉠: 준비, ㉡: 실행, ㉢: 대기
② ㉠: 대기, ㉡: 준비, ㉢: 실행
③ ㉠: 준비, ㉡: 대기, ㉢: 실행
④ ㉠: 대기, ㉡: 실행, ㉢: 준비

정답 2번

출제 포인트

프로세스의 대기·준비·실행 상태를 각각 무엇을 기다리는지와 프로세서를 점유하고 있는지로 구분하는 문제이다.

핵심 개념

대기 상태: 입출력 완료나 특정 이벤트의 발생을 기다리는 상태이다.
준비 상태: 실행에 필요한 조건은 갖추었지만 프로세서를 할당받기 위해 기다리는 상태이다.
실행 상태: 프로세서를 점유하여 명령어를 수행하는 상태이다.
따라서 ㉠은 대기, ㉡은 준비, ㉢은 실행이다.

NoteBase

선지별 해설

① X
㉠을 준비, ㉡을 실행, ㉢을 대기로 연결한 부분이 모두 틀렸다. 올바르게는 ㉠은 이벤트를 기다리는 대기 상태, ㉡은 프로세서 할당을 기다리는 준비 상태, ㉢은 프로세서를 점유한 실행 상태이다.

② O
㉠의 이벤트 대기는 대기 상태, ㉡의 프로세서 할당 대기는 준비 상태, ㉢의 명령어 수행은 실행 상태에 각각 해당한다.

③ X
㉠을 준비, ㉡을 대기로 연결한 부분이 틀렸다. 올바르게는 ㉠이 대기 상태이고 ㉡이 준비 상태이며, ㉢의 실행 상태 연결은 맞다.

④ X
㉠을 대기로 연결한 것은 맞지만 ㉡을 실행, ㉢을 준비로 연결한 부분이 틀렸다. 올바르게는 ㉡이 준비 상태이고 ㉢이 실행 상태이다.

오답 포인트

㉠과 ㉡은 모두 무언가를 기다리므로 혼동하기 쉽다. 이벤트를 기다리면 대기 상태이고, 프로세서 할당만 기다리면 준비 상태이다.

다시 나오면 이렇게 판단

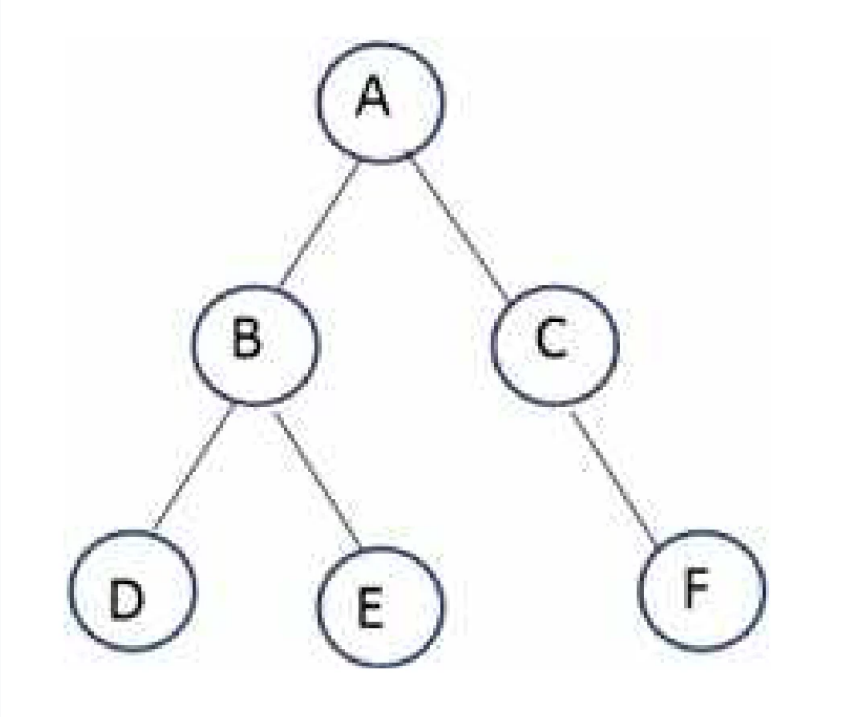
기다리는 대상부터 확인한다. 입출력 완료나 이벤트를 기다리면 대기, 프로세서를 기다리면 준비, 프로세서를 실제로 점유하여 명령어를 수행하면 실행으로 판단한다.

암기 포인트

이벤트 대기＝대기, CPU 대기＝준비, CPU 점유＝실행이다.

NoteBase

문 6. 다음 이진 트리를 중위(inorder) 순회한 경우 3번째, 5번째로 방문되는 노드를 나열한 것으로 가장 옳은 것은?



① D, C

② E, C

③ B, C

④ C, E

정답 2번

출제 포인트

이 문제는 이진 트리의 중위 순회가 왼쪽 서브트리, 루트, 오른쪽 서브트리 순서로 진행된다는 원리를 묻는 문제이다.

NoteBase

올바른 순서

중위 순회의 기본 순서는 왼쪽 자식(서브트리) → 현재 노드 → 오른쪽 자식(서브트리)이다.
주어진 트리에서 A의 왼쪽 서브트리인 B 부분을 먼저 중위 순회하면 D → B → E가 된다. 이어서 루트 A를 방문하고, A의 오른쪽 서브트리인 C 부분을 중위 순회하면 C → F가 된다.
따라서 전체 방문 순서는 D → B → E → A → C → F이며, 3번째 노드는 E, 5번째 노드는 C이다.
B보다 D가 먼저 방문되는 이유는 D가 B의 왼쪽 자식이고, E가 B보다 나중에 방문되는 이유는 E가 B의 오른쪽 자식이기 때문이다. 같은 규칙에 따라 오른쪽 서브트리에서는 왼쪽 자식이 없는 C를 먼저 방문한 뒤 오른쪽 자식 F를 방문한다.

선지별 해설

① X — D와 C는 각각 1번째와 5번째로 방문된다. 틀린 부분은 3번째 노드를 D로 제시한 것이며, 올바르게는 3번째 노드는 E이고 5번째 노드는 C이다.
② O — 전체 순서가 D → B → E → A → C → F이므로 3번째는 E, 5번째는 C이다.
③ X — B와 C는 각각 2번째와 5번째로 방문된다. 틀린 부분은 3번째 노드를 B로 제시한 것이며, 올바르게는 3번째 노드는 E이고 5번째 노드는 C이다.
④ X — C와 E는 각각 5번째와 3번째로 방문된다. 틀린 부분은 두 노드의 방문 위치를 거꾸로 제시한 것이며, 올바르게는 3번째 노드는 E이고 5번째 노드는 C이다.

오답 포인트

중위 순회를 루트부터 시작한다고 생각하거나, 왼쪽 서브트리의 노드를 단순히 위에서 아래로 읽으면 순서가 어긋난다. 특히 ④는 실제 3번째와 5번째 노드인 E와 C의 위치만 서로 바꾼 함정이다.

다시 나오면 이렇게 판단

각 노드마다 왼쪽 서브트리를 끝까지 처리한 뒤 현재 노드를 방문하고, 마지막에 오른쪽 서브트리를 처리한다. 전체 방문 순서를 먼저 한 줄로 적은 다음 요구된 번째의 노드를 고르면 된다.

암기 포인트

중위 순회는 왼쪽 → 루트 → 오른쪽이며, 주어진 트리의 순서는 D-B-E-A-C-F이다.

NoteBase

문 7. PCM(Pulse Code Modulation) 디지털화 과정으로 가장 옳은 것은?

① 표본화(Sampling)-양자화(Quantization)-부호화(Encoding)

② 양자화(Quantization)-표본화(Sampling)-부호화(Encoding)

③ 부호화(Encoding)-양자화(Quantization)-표본화(Sampling)

④ 표본화(Sampling)-부호화(Encoding)-양자화(Quantization)

정답 1번

출제 포인트

이 문제는 아날로그 신호를 PCM 디지털 신호로 바꾸는 표본화, 양자화, 부호화의 처리 순서를 묻는 문제이다.

올바른 순서

1. 표본화(Sampling) — 연속적인 아날로그 신호를 일정한 시간 간격으로 측정하여 시간축을 이산화한다.
2. 양자화(Quantization) — 각 표본의 연속적인 진폭값을 미리 정한 유한한 단계 중 하나로 근사한다.
3. 부호화(Encoding) — 양자화된 단계값을 이진 부호로 변환한다.

표본값이 있어야 그 진폭을 양자화할 수 있고, 양자화된 단계값이 있어야 이를 이진 부호로 표현할 수 있으므로 표본화 → 양자화 → 부호화 순서가 된다.

선지별 해설

① O — 표본화로 시간축을 이산화한 뒤 양자화로 진폭을 이산화하고, 마지막으로 그 값을 부호화하므로 올바른 순서이다.

② X — '양자화-표본화'의 선후관계가 바뀌었다. 양자화는 먼저 얻은 표본의 진폭값을 유한한 단계로 바꾸는 과정이므로 표본화가 선행해야 한다. 올바르게는 '표본화-양자화-부호화'이다.

③ X — '부호화-양자화-표본화'는 전체 순서가 거꾸로 되어 있다. 부호화할 대상인 양자화값은 표본화와 양자화를 거친 뒤에 만들어진다. 올바르게는 '표본화-양자화-부호화'이다.

④ X — '부호화-양자화'의 선후관계가 바뀌었다. 이진 부호는 양자화된 단계값에 부여하므로 양자화가 부호화보다 먼저이다. 올바르게는 '표본화-양자화-부호화'이다.

오답 포인트

표본화와 양자화를 모두 '신호를 불연속적으로 만드는 과정'으로 보아 순서를 혼동하기 쉽다. 표본화는 시간을 이산화하고, 양자화는 표본의 진폭을 이산화하며, 부호화는 그 결과를 비트로 표현한다는 차이가 핵심이다.

NoteBase

다시 나오면 이렇게 판단

PCM 순서 문제에서는 먼저 어느 시점의 값을 읽을지 정하는 표본화, 다음으로 읽은 값의 단계를 정하는 양자화, 마지막으로 단계값을 비트로 나타내는 부호화의 의존관계를 확인한다.

암기 포인트

PCM은 '표-양-부', 즉 표본화 → 양자화 → 부호화이다.

NoteBase

문 8. 다음 <보기>는 프로세스의 일반적인 메모리 구조의 각 영역을 설명한 것이다. 이 중 스택(stack) 영역에 해당하는 설명만을 모아놓은 것은?

<보기>
㉠ 이 영역에 데이터를 저장하기 위해서 C는 malloc(), C++은 new를 사용
㉡ 지역(local) 변수, 매개변수(parameter), 리턴 값 등 잠시 사용되었다가 사라지는 데이터를 저장
㉢ 전역(global) 변수, 정적(static) 변수, 배열, 구조체 등을 저장
㉣ 함수 호출 시 생성되고 함수가 끝나면 반환

① ㉠, ㉡
② ㉡, ㉢
③ ㉠, ㉣
④ ㉡, ㉣

정답 4번

출제 포인트

프로세스 메모리에서 스택 영역과 힙·데이터 영역을 구분하는 문제이다. 스택은 함수 호출과 함께 생성되는 일시적 데이터를 저장한다.

핵심 개념

스택 영역: 함수 호출 과정에서 지역 변수, 매개변수 등 일시적인 데이터를 저장하며 함수가 끝나면 해당 스택 프레임이 반환된다.
힙 영역: 실행 중 동적으로 할당한 메모리를 저장하는 영역으로, C의 malloc()이나 C++의 new로 확보한다.
데이터 영역: 전역 변수와 정적 변수처럼 프로그램 실행 동안 유지되는 데이터를 저장한다.
따라서 ㉠은 힙, ㉡과 ㉣은 스택, ㉢은 데이터 영역에 관한 설명이다.

NoteBase

선지별 해설

① X
㉠은 malloc()과 new를 이용한 동적 할당을 설명하므로 힙 영역에 해당한다. ㉡만 스택 설명이고 ㉣이 빠져 있다. 틀린 조합 '㉠, ㉡'은 '㉡, ㉣'로 고쳐야 한다.

② X
㉡은 지역 변수와 매개변수 등 일시적 데이터를 저장하므로 스택에 해당하지만, ㉢은 전역 변수와 정적 변수 등을 저장하는 데이터 영역의 설명이다. 틀린 조합 '㉡, ㉢'은 '㉡, ㉣'로 고쳐야 한다.

③ X
㉠은 힙 영역, ㉣은 스택 영역에 해당하므로 둘 모두가 스택 설명인 조합이 아니다. 틀린 조합 '㉠, ㉣'은 '㉡, ㉣'로 고쳐야 한다.

④ O
㉡은 함수 실행 중 잠시 쓰이는 지역 변수·매개변수 등을 설명하고, ㉣은 함수 호출 때 생성되어 함수 종료 때 반환되는 성질을 설명한다. 둘 다 스택 영역의 특징이다.

오답 포인트

㉠의 '데이터를 저장'이라는 표현만 보고 스택으로 판단하기 쉽다. 그러나 malloc()과 new는 동적 메모리 할당의 대표 표지이므로 ㉠은 힙이다. 또한 ㉢의 배열과 구조체는 선언 위치와 저장 기간에 따라 영역이 달라질 수 있지만, 이 보기에서는 전역 변수·정적 변수와 함께 제시되어 데이터 영역을 가리킨다.

다시 나오면 이렇게 판단

malloc()·new가 보이면 힙, 지역 변수·매개변수·함수 호출과 종료가 보이면 스택, 전역 변수·정적 변수가 보이면 데이터 영역으로 분류한다.

암기 포인트

동적 할당은 힙, 함수의 일시적 실행 정보는 스택, 전역·정적 데이터는 데이터 영역이다.

NoteBase

문 9. 다음은 두 릴레이션 R과 S에 대해 관계 대수 연산을 실행한 결과이다. 여기에 적용된 관계 대수 연산으로 가장 옳은 것은?

R

A	B	C
1	a	10
2	b	20
3	c	20

S

C	D
20	x
30	y

결과

A	B	C
2	b	20
3	c	20

① $R \times S$

② $R \bowtie S$

③ R ⊠ S

④ $R \ltimes S$

정답 4번

간단한 해설

결과에는 S와 공통 속성 C의 값이 일치하는 R의 행만 남고, 속성도 **R의 A·B·C만 유지**된다. 이는 왼쪽 릴레이션의 일치하는 행만 반환하는 **세미조인**의 특징이다.

공통 해설

세미조인 $R \ltimes S$은 공통 속성의 값이 같은 S의 튜플이 존재하는 R의 튜플만 골라 내는 연산이다. 튜플은 릴레이션의 한 행을 뜻하며, 결과에는 **왼쪽 릴레이션 R의 속성만** 남는다.

표에서 공통 속성은 C이다. R의 (1, a, 10)은 S에 C가 10인 행이 없어 제외된다. R의 (2, b, 20)과 (3, c, 20)은 모두 S의 (20, x)와 C 값이 같으므로 남는다. 따라서 결과의 두 행은 (2, b, 20), (3, c, 20)이고, 속성은 A·B·C이다.

이 결과에는 **S의 D 속성이 없다**. 카티션 곱, 자연 조인, 오른쪽 외부 조인(R ⊠ S)은 S의 D 속성도 결과에 포함하므로 주어진 결과와 구별된다. S와의 일치 여부로 R의 행을 걸러 내면서 R의 속성만 남긴 결과는 세미조인에 해당한다.

NoteBase

문 10. 다음 직원 테이블에 대해 <보기>의 SQL문을 실행한 결과 테이블의 카디널리티(cardinality)는?

직원

번호	이름	소속
1	김경감	100함
2	이경위	P-20정
3	박순경	100함
4	정경위	300함

< 보 기 >

```
SELECT DISTINCT 소속
FROM 직원;
```

① 1
② 2
③ 3
④ 4

정답 3번

출제 포인트

이 문제는 SELECT DISTINCT가 중복 값을 제거한다는 점을 적용하여 결과 테이블의 행 수인 카디널리티를 구하는 문제이다.

자료 읽는 순서

1. SELECT 절에서 조회하는 속성이 소속임을 확인한다.
2. DISTINCT가 있으므로 소속 열의 중복 값을 제거한다.
3. 남은 서로 다른 소속 값의 개수를 센다.

NoteBase

핵심 근거

직원 테이블의 소속 값은 차례로 100함, P-20정, 100함, 300함이다. 중복된 100함을 하나로 합치면 100함, P-20정, 300함의 3개 값만 남는다. 따라서 결과 테이블의 카디널리티는 3이다.

선지별 해설

① X — 카디널리티가 1이라는 선택지이다. DISTINCT는 모든 행을 하나로 만드는 것이 아니라 같은 소속 값만 하나로 합치므로, 올바른 카디널리티는 3이다.
② X — 카디널리티가 2라는 선택지이다. 소속 열에는 100함 외에도 P-20정과 300함이 있으므로, 올바른 카디널리티는 3이다.
③ O — 중복을 제거한 소속 값은 100함, P-20정, 300함의 3개이므로 결과 테이블의 카디널리티는 3이다.
④ X — 카디널리티가 4라는 선택지이다. 원본 테이블은 4행이지만 100함이 두 번 나타나므로 DISTINCT 적용 후에는 한 행이 줄어든다. 올바른 카디널리티는 3이다.

오답 포인트

원본 직원 수 4를 그대로 세어 ④를 고르거나, DISTINCT를 단순히 행을 줄이는 기능으로만 이해해 ①이나 ②를 고르기 쉽다. DISTINCT는 지정된 열에서 값이 완전히 같은 행만 중복으로 제거한다.

다시 나오면 이렇게 판단

SELECT DISTINCT 열이 나오면 해당 열의 값만 나열한 뒤 같은 값을 하나씩만 남긴다. 그때 남은 행의 수가 결과 테이블의 카디널리티이다.

암기 포인트

DISTINCT는 중복 제거, 카디널리티는 행의 수이다.

NoteBase

문 11. IPv6에 관한 설명으로 가장 옳지 않은 것은?

① IPv6 주소의 비트 수는 IPv4 주소 비트 수의 2배이다.

② 128비트의 길이를 갖는다.

③ IPv6 각 패킷의 기본 헤더는 40바이트로 고정된다.

④ IPv6은 유니캐스트, 멀티캐스트 주소뿐만 아니라 새로운 주소 형태인 애니캐스트 주소가 도입되었다.

정답 1번

출제 포인트

IPv4와 IPv6의 주소 길이, 기본 헤더 크기, 주소 유형을 구분하는 문제이다.

핵심 개념

IPv4는 32비트 주소를 사용하고, IPv6는 128비트 주소를 사용한다. 따라서 IPv6 주소의 비트 수는 IPv4의 4배이다. IPv6의 기본 헤더는 40바이트의 고정 길이이며, 주소 유형으로 유니캐스트·멀티캐스트·애니캐스트를 사용한다.

선지별 해설

① X — 틀린 표현은 'IPv4 주소 비트 수의 2배'이다. IPv4는 32비트, IPv6는 128비트이므로 올바르게는 'IPv4 주소 비트 수의 4배'이다.
② O — IPv6 주소의 길이는 128비트이다.
③ O — IPv6 패킷의 기본 헤더는 40바이트의 고정 길이이다. 추가 정보는 확장 헤더로 분리한다.
④ O — IPv6은 유니캐스트와 멀티캐스트뿐 아니라 애니캐스트 주소 유형도 사용한다. 애니캐스트는 같은 주소가 지정된 여러 인터페이스 가운데 라우팅상 가까운 하나로 패킷을 전달한다.

오답 포인트

①의 '2배'는 주소 길이의 숫자만 어렴풋이 기억할 때 선택하기 쉬운 함정이다. 128을 32로 나누면 4이므로 2배가 아니라 4배이다.

다시 나오면 이렇게 판단

주소 길이는 IPv4 32비트, IPv6 128비트를 먼저 대입해 배수를 계산하고, IPv6 기본 헤더는 40바이트, 주소 유형은 유니캐스트·멀티캐스트·애니캐스트로 판단한다.

암기 포인트

IPv4 32비트, IPv6 128비트·기본 헤더 40바이트·유니/멀티/애니캐스트이다.

NoteBase

문 12. 다음의 중위(infix) 표기식을 후위(postfix) 표기식으로 변환한 것으로 가장 올바른 것은?

$(A + B) * (C - D/E)$

① ABCDE+*-/

② AB+CD-E/*

③ AB+CDE/-*

④ AB+*CD-E/

정답 3번

출제 포인트

중위 표기식의 괄호와 연산자 우선순위를 유지하면서 각 연산자를 두 피연산자 뒤로 옮겨 후위 표기식으로 변환하는 문제이다.

올바른 순서

1. $A + B$를 변환하면 AB+가 된다.
2. D/E를 먼저 변환하면 DE/가 된다.
3. $C - D/E$는 C와 DE/ 뒤에 뺄셈 연산자를 놓으므로 CDE/-가 된다.
4. 두 부분식 뒤에 곱셈 연산자를 놓으면 최종 결과는 AB+CDE/-*가 된다.

나눗셈이 뺄셈보다 우선이고, 곱셈은 두 괄호 부분식이 모두 완성된 뒤 수행되므로 이 순서가 된다.

선지별 해설

① X — ABCDE+*-/는 첫 번째 부분식 $A + B$를 나타내는 AB+가 앞에 완성되지 않는다. 틀린 배열 ABCDE+*-/은 올바르게 AB+CDE/-*로 고쳐야 한다.

② X — CD-E/는 $(C - D)/E$의 순서이므로 원식의 $C - (D/E)$와 다르다. 틀린 부분 CD-E/은 올바르게 CDE/-로 고쳐야 하며, 전체 식은 AB+CDE/-*이다.

③ O — AB+가 $A + B$, DE/가 D/E, CDE/-가 $C - D/E$를 나타내고 마지막 $*$가 두 부분식을 곱하므로 원식과 일치한다.

④ X — AB+*에서는 오른쪽 피연산식이 나오기 전에 이항 연산자 $*$가 등장하므로 올바른 후위 표기식이 아니다. 또한 CD-E/는 $(C - D)/E$를 뜻한다. 곱셈 연산자는 마지막에 놓고 전체를 AB+CDE/-*로 고쳐야 한다.

오답 포인트

②처럼 $C - D/E$를 $(C - D)/E$로 잘못 묶거나, ④처럼 바깥쪽 곱셈 연산자를 오른쪽 부분식이 완성되기 전에 놓는 것이 핵심 함정이다.

NoteBase

다시 나오면 이렇게 판단

가장 안쪽이거나 우선순위가 높은 연산부터 부분식을 후위 표기로 완성하고, 각 이항 연산자를 왼쪽 부분식과 오른쪽 부분식 뒤에 붙인다.

암기 포인트

후위 표기는 피연산자 두 개를 먼저 쓰고 연산자를 나중에 쓰며, 바깥 연산자일수록 뒤에 놓인다.

NoteBase

문 13. 다음은 어떤 정렬 알고리즘의 수행 과정을 보인 것이다. 수행된 정렬 알고리즘으로 가장 옳은 것은?

초기값	3	5	1	2	4
1단계	1	5	3	2	4
2단계	1	2	3	5	4
3단계	1	2	3	5	4
4단계	1	2	3	4	5

① 버블 정렬

② 선택 정렬

③ 삽입 정렬

④ 힙 정렬

정답 2번

출제 포인트

선택 정렬은 매 단계마다 미정렬 구간에서 최솟값을 찾아 그 구간의 맨 앞 원소와 교환하고, 정렬된 구간을 왼쪽부터 한 칸씩 확장한다. 제시된 배열에서 1, 2, 3, 4가 차례로 제자리에 확정되는지를 판별하는 문제이다.

올바른 순서

1단계: [3, 5, 1, 2, 4]에서 전체 최솟값 1을 첫 원소 3과 교환하여 [1, 5, 3, 2, 4]가 된다.
2단계: 미정렬 구간 [5, 3, 2, 4]의 최솟값 2를 그 구간의 첫 원소 5와 교환하여 [1, 2, 3, 5, 4]가 된다.
3단계: 미정렬 구간 [3, 5, 4]의 최솟값 3이 이미 맨 앞에 있으므로 배열이 [1, 2, 3, 5, 4]로 유지된다.
4단계: 미정렬 구간 [5, 4]의 최솟값 4를 5와 교환하여 [1, 2, 3, 4, 5]가 된다.
왜 이 순서인가: 매 단계에서 선택한 최솟값은 현재 미정렬 구간의 맨 앞자리에 확정되므로 다음 단계는 그 오른쪽 구간만 다시 탐색한다.

NoteBase

선지별 해설

① X — 버블 정렬은 인접한 두 원소를 차례로 비교·교환하며, 오름차순의 한 패스가 끝나면 미정렬 구간의 최댓값이 오른쪽 끝에 확정된다. 제시된 1단계처럼 전체 최솟값 1을 곧바로 첫 자리로 옮기는 방식이 아니다. 올바르게는 '인접 원소를 반복 교환하여 큰 값을 오른쪽부터 확정하는 버블 정렬'이다.
② O — 선택 정렬은 미정렬 구간의 최솟값을 선택해 그 구간의 첫 원소와 교환한다. 제시된 과정도 1, 2, 3, 4를 왼쪽부터 차례로 확정하므로 선택 정렬이다.
③ X — 삽입 정렬은 현재 원소를 이미 정렬된 왼쪽 부분의 알맞은 위치에 삽입하여 정렬 구간을 확장한다. 제시된 과정처럼 미정렬 구간 전체에서 최솟값을 찾아 앞자리와 교환하는 방식이 아니다. 올바르게는 '현재 원소를 정렬된 앞부분에 삽입하는 삽입 정렬'이다.
④ X — 힙 정렬은 먼저 최대 힙 또는 최소 힙을 구성한 뒤 루트와 배열 끝 원소를 교환하고 남은 힙을 다시 조정한다. 제시된 과정에는 힙 구성과 재조정이 없으며, 미정렬 구간의 최솟값을 직접 선택한다. 올바르게는 '힙을 구성하고 루트 교환과 힙 재구성을 반복하는 힙 정렬'이다.

오답 포인트

1단계에서 최솟값 1이 맨 앞으로 이동한 모습만 보고 단순한 교환 정렬로 혼동하기 쉽다. 결정적인 단서는 2단계에서도 남은 구간의 최솟값 2가 두 번째 자리에 오고, 3단계에서는 최솟값 3이 이미 제자리에 있어 배열이 그대로 유지된다는 점이다.

다시 나오면 이렇게 판단

각 단계 뒤에 어느 위치가 확정되는지를 본다. 미정렬 구간의 최솟값이 왼쪽부터 하나씩 확정되면 선택 정렬이고, 인접 교환으로 최댓값이 오른쪽부터 확정되면 버블 정렬이다.

암기 포인트

선택 정렬은 '남은 것 중 최솟값을 선택하여 맨 앞으로'이다.

NoteBase

문 14. 다음 <보기>에서 자바(JAVA)에 해당하는 설명만으로 모아놓은 것은? (단, 자바(JAVA) 9버전 이전으로 한정한다.)

<보기>
㉠ 쓰레기 수집 기능을 수행한다.
㉡ 변수에 대해 타입 선언 없이 사용할 수 있다.
㉢ 가상 머신이 동일한 실행 환경을 제공하므로 아키텍처 중립적이다.
㉣ 포인터를 이용하여 메모리의 내용에 직접 접근할 수 있다.

① ㉠, ㉢
② ㉠, ㉣
③ ㉡, ㉢
④ ㉢, ㉣

정답 1번

출제 포인트

자바의 자동 메모리 관리, 정적 타입 체계, JVM에 의한 플랫폼 독립성, 포인터 미지원이라는 핵심 특성을 구별하는 문제이다.

핵심 개념

㉠ O: 자바의 가비지 컬렉터는 더 이상 사용하지 않는 객체가 차지한 메모리를 자동으로 회수한다.
㉡ X: 자바 9 이전에는 변수를 사용할 때 자료형을 명시해야 하는 정적 타입 언어이다. 지역 변수 타입 추론을 위한 var도 자바 10에서 도입되었으므로 문제의 범위에는 해당하지 않는다.
㉢ O: 자바 소스 코드는 바이트코드로 컴파일되고, 각 운영체제와 하드웨어에 맞는 JVM이 이를 실행하므로 바이트코드는 아키텍처 중립적이다.
㉣ X: 자바는 C나 C++처럼 포인터를 이용해 임의의 메모리 주소에 직접 접근하는 기능을 제공하지 않는다. 객체는 참조를 통해 다룬다.

선지별 해설

① O: ㉠의 쓰레기 수집 기능과 ㉢의 JVM 기반 아키텍처 중립성은 모두 자바의 특성이므로 옳은 조합이다.
② X: ㉣의 '포인터를 이용한 메모리 직접 접근'이 틀렸다. 올바른 조합은 ㉠, ㉢이다.
③ X: ㉡의 '타입 선언 없이 변수 사용'이 틀렸다. 자바 9 이전에는 변수의 자료형을 명시하며, 올바른 조합은 ㉠, ㉢이다.
④ X: ㉣의 '포인터를 이용한 메모리 직접 접근'이 틀렸고, 옳은 ㉠도 빠져 있다. 올바른 조합은 ㉠, ㉢이다.

NoteBase

오답 포인트

‘참조’와 ‘포인터를 통한 메모리 직접 접근’을 같은 것으로 보는 것이 함정이다. 자바의 참조는 객체를 가리키지만 주소 연산이나 임의의 메모리 직접 접근을 허용하지 않는다. 또한 자바 10의 var를 자바 9 이전에도 사용할 수 있다고 혼동하면 ㉡을 옳다고 판단하게 된다.

다시 나오면 이렇게 판단

자바 특성은 ‘가비지 컬렉션 O, JVM 기반 플랫폼 독립성 O, 타입 없는 변수 선언 X, 포인터에 의한 직접 메모리 접근 X’로 판별한다. 버전 조건이 있으면 해당 기능의 도입 시점도 함께 확인한다.

암기 포인트

자바는 GC와 JVM을 사용하고, 자바 9 이전에는 명시적 타입 선언이 필요하며, 포인터 직접 접근은 지원하지 않는다.

NoteBase

문 15. 어떤 정부 기관에서 네이버 클라우드를 이용함과 동시에 기관 기밀 자료의 처리를 위해 오픈스택(OpenStack)을 이용한 온프레미스(on-premise) 클라우드를 구축하여 같이 사용하고 있다면, 이러한 클라우드 전개 방식을 지칭하는 용어로 가장 옳은 것은?

① 프라이빗 클라우드(private cloud)

② 퍼블릭 클라우드(public cloud)

③ 하이브리드 클라우드(hybrid cloud)

④ 멀티 클라우드(multicloud)

정답 3번

출제 포인트

이 문제는 공개 클라우드와 기관 전용 온프레미스 클라우드를 함께 사용하는 사례에 알맞은 클라우드 전개 방식을 구별하는 문제이다.

사례 정리

핵심 사실 1: 정부 기관은 외부 사업자가 제공하는 네이버 클라우드를 이용한다.
핵심 사실 2: 기밀 자료 처리를 위해 OpenStack 기반의 기관 전용 온프레미스 클라우드도 구축한다.
판단에 중요한 조건: 공개 클라우드와 프라이빗 클라우드라는 서로 다른 환경을 함께 사용한다.

핵심 개념

하이브리드 클라우드는 프라이빗 클라우드와 퍼블릭 클라우드처럼 서로 다른 클라우드 환경을 결합하여 사용하는 전개 방식이다. 공개 서비스의 활용성과 전용 환경의 통제·보안 필요를 함께 충족하려는 경우에 적용된다.

적용 기준

클라우드가 설치된 장소만 보는 것이 아니라 어떤 전개 환경들이 결합되었는지를 판단한다. 이 사례는 외부의 퍼블릭 클라우드와 기관 내부의 프라이빗 클라우드를 병행하므로 하이브리드 클라우드에 해당한다.

NoteBase

선지별 해설

① X — 프라이빗 클라우드는 한 기관이 전용으로 사용하는 클라우드 환경을 뜻한다. 이 사례에는 기관 전용 온프레미스 클라우드뿐 아니라 네이버의 퍼블릭 클라우드도 포함되므로 전체 전개 방식을 프라이빗 클라우드라고 할 수 없다. 틀린 표현 '프라이빗 클라우드'는 '하이브리드 클라우드'로 고쳐야 한다.
② X — 퍼블릭 클라우드는 클라우드 사업자가 다수 이용자에게 제공하는 공유형 서비스이다. 네이버 클라우드만 이용한다면 이에 해당할 수 있지만, 사례는 기관 전용 온프레미스 클라우드까지 함께 사용한다. 틀린 표현 '퍼블릭 클라우드'는 '하이브리드 클라우드'로 고쳐야 한다.
③ O — 네이버의 퍼블릭 클라우드와 OpenStack 기반의 기관 전용 프라이빗 클라우드를 함께 사용하므로 하이브리드 클라우드이다.
④ X — 멀티 클라우드는 일반적으로 둘 이상의 클라우드 사업자가 제공하는 클라우드 서비스를 함께 활용하는 방식에 초점이 있다. 이 사례의 핵심은 여러 외부 사업자의 서비스를 이용하는 것이 아니라 퍼블릭 환경과 자체 프라이빗 환경을 결합한 것이다. 틀린 표현 '멀티 클라우드'는 '하이브리드 클라우드'로 고쳐야 한다.

오답 포인트

OpenStack과 네이버 클라우드라는 서로 다른 이름이 등장한다는 이유만으로 ④ 멀티 클라우드를 고르기 쉽다. 그러나 OpenStack은 이 사례에서 기관 전용 온프레미스 환경을 구축하는 기술이며, 결정적 단서는 퍼블릭 클라우드와 프라이빗 클라우드의 결합이다.

다시 나오면 이렇게 판단

사례에 외부 공개형 클라우드와 조직 전용 클라우드가 함께 나오면 하이브리드 클라우드로 판단한다. 서로 다른 여러 클라우드 사업자의 서비스를 함께 쓴다는 점이 중심이면 멀티 클라우드로 판단한다.

암기 포인트

퍼블릭과 프라이빗의 결합은 하이브리드, 여러 클라우드 사업자의 조합은 멀티 클라우드이다.

NoteBase

문 16. 다음 <보기> 중 객체지향 언어의 설명으로 옳은 것은 모두 몇 개인가?

<보기>
㉠ 객체는 재사용이 가능하나 클래스는 재사용이 불가능하다.
㉡ 다형성(polymorphism)을 지원한다.
㉢ 추상메소드는 메소드의 선언만 하고 그 내용은 기술하지 않는 메소드이다.
㉣ 상속(inheritance)이란 속성과 관련된 오퍼레이션을 클래스 안에 묶어서 하나로 취급하는 것이다.

① 1개
② 2개
③ 3개
④ 4개

정답 2번

출제 포인트

객체지향의 핵심 특성인 재사용성, 다형성, 추상 메소드, 상속과 캡슐화의 정의를 구별하고 옳은 설명의 개수를 판단하는 문제이다.

핵심 개념

클래스: 객체의 속성과 동작을 정의하는 설계도로, 여러 객체를 생성하거나 상속하는 방식으로 재사용할 수 있다.
다형성: 같은 메시지나 인터페이스가 객체의 실제 종류에 따라 서로 다른 동작을 수행하는 성질이다.
추상 메소드: 선언만 있고 구현 내용이 없는 메소드이다.
상속: 상위 클래스의 속성과 메소드를 하위 클래스가 물려받아 확장하거나 재사용하는 것이다.
캡슐화: 데이터인 속성과 그 데이터를 처리하는 연산을 하나의 단위로 묶고 내부 구현을 감추는 것이다.

보기 판단

㉠ X — '클래스는 재사용이 불가능하다'가 틀린 표현이다. 클래스는 여러 객체를 만드는 설계도로 재사용할 수 있고, 상속을 통해 기존 클래스를 확장하여 재사용할 수도 있다.
㉡ O — 객체지향 언어는 같은 인터페이스가 객체의 실제 종류에 따라 다르게 동작하는 다형성을 지원한다.
㉢ O — 추상 메소드는 선언만 하고 구체적인 구현 내용은 기술하지 않는 메소드이다.
㉣ X — '상속이란 속성과 관련된 오퍼레이션을 클래스 안에 묶는 것'이 틀린 표현이다. 이는 캡슐화의 설명이며, 상속은 상위 클래스의 속성과 메소드를 하위 클래스가 물려받는 것이다.
따라서 옳은 것은 ㉡, ㉢의 2개이다.

NoteBase

선지별 해설

① X — 옳은 보기는 1개가 아니라 ㉡과 ㉢의 2개이다. ‘1개’는 ‘2개’로 고쳐야 한다.
② O — ㉡과 ㉢이 옳으므로 옳은 보기의 수는 2개이다.
③ X — 옳은 보기는 3개가 아니라 ㉡과 ㉢의 2개이다. ‘3개’는 ‘2개’로 고쳐야 한다.
④ X — 옳은 보기는 4개가 아니라 ㉡과 ㉢의 2개이다. ‘4개’는 ‘2개’로 고쳐야 한다.

오답 포인트

㉣은 상속과 캡슐화의 정의를 바꾸어 놓은 함정이다. 또한 객체지향의 재사용성은 클래스에도 적용되므로 ㉠의 ‘클래스는 재사용이 불가능하다’는 표현에 주의해야 한다.

다시 나오면 이렇게 판단

각 보기에서 핵심 동사를 확인한다. ‘같은 인터페이스로 다른 동작’은 다형성, ‘선언만 있고 구현 없음’은 추상 메소드, ‘상위 클래스의 구성 요소를 물려받음’은 상속, ‘데이터와 연산을 하나로 묶음’은 캡슐화로 판단한다.

암기 포인트

다형성은 하나의 인터페이스·여러 동작, 추상 메소드는 선언만, 상속은 물려받기, 캡슐화는 묶어서 감추기이다.

NoteBase

문 17. 다음 <보기> 중 컴파일러(compiler) 방식만으로 처리되는 언어는 모두 몇 개인가? (단, 프로그래밍 언어의 일반적인 경우로 한정한다.)

<보기>
㉠ FORTRAN
㉡ BASIC
㉢ 파이썬(python)
㉣ C++
㉤ COBOL

① 1개
② 2개
③ 3개
④ 4개

정답 3번

출제 포인트

이 문제는 프로그래밍 언어를 일반적인 처리 방식에 따라 컴파일러 방식과 인터프리터 방식으로 구분하고, 컴파일러 방식으로 처리되는 언어의 수를 묻는다.

핵심 개념

컴파일러 방식은 소스 프로그램 전체를 번역하여 목적 프로그램을 만든 뒤 실행하는 방식이다. 인터프리터 방식은 프로그램을 실행하면서 명령을 차례로 해석하여 처리하는 방식이다.
일반적인 시험 분류에서 FORTRAN, C++, COBOL은 컴파일러 방식이고, BASIC과 파이썬은 인터프리터 방식이다. 따라서 해당 언어는 ㉠, ㉣, ㉤의 3개이다.

선지별 해설

① X — ‘1개’가 틀린 수치이다. 컴파일러 방식으로 분류되는 언어는 FORTRAN, C++, COBOL의 3개이다.
② X — ‘2개’가 틀린 수치이다. BASIC과 파이썬을 제외한 3개가 컴파일러 방식이다.
③ O — FORTRAN, C++, COBOL의 3개이므로 옳다.
④ X — ‘4개’가 틀린 수치이다. BASIC과 파이썬은 일반적인 시험 분류에서 인터프리터 방식이므로 컴파일러 방식에 포함하지 않으며, 올바른 수는 3개이다.

NoteBase

오답 포인트

파이썬이 실행 과정에서 바이트코드로 변환된다는 사실이나 BASIC에 컴파일 구현도 있다는 사실을 떠올려 모두 컴파일러 방식에 넣기 쉽다. 그러나 문항은 '일반적인 경우'로 한정하므로 전통적인 시험 분류를 적용한다.

다시 나오면 이렇게 판단

언어별 대표 처리 방식을 먼저 분류한다. FORTRAN·C++·COBOL은 컴파일러, BASIC·파이썬은 인터프리터로 묶은 뒤 요구된 집합의 개수를 센다.

암기 포인트

컴파일 3개는 FORTRAN·C++·COBOL, 인터프리터 2개는 BASIC·파이썬이다.

NoteBase

문 18. 다음 <보기>의 저장장치 중 접근(access) 속도가 가장 빠른 것과 가장 느린 것은?

<보기>
㉠ 캐시메모리 ㉡ 주기억장치
㉢ SSD(Solid State Disk) ㉣ 레지스터

① 가장 빠른 것: 캐시메모리, 가장 느린 것: 주기억장치
② 가장 빠른 것: 캐시메모리, 가장 느린 것: SSD
③ 가장 빠른 것: 레지스터, 가장 느린 것: 주기억장치
④ 가장 빠른 것: 레지스터, 가장 느린 것: SSD

정답 4번

출제 포인트

레지스터, 캐시메모리, 주기억장치, SSD로 이루어진 저장계층에서 접근 속도의 상대적 순서를 판단하는 문제이다.

핵심 개념

레지스터: CPU 내부에서 연산에 필요한 값을 직접 보관하는 가장 빠른 저장장치이다.
캐시메모리: CPU와 주기억장치의 속도 차이를 줄이기 위해 자주 사용하는 데이터와 명령어를 임시 저장한다.
주기억장치: 실행 중인 프로그램과 데이터를 저장하며, 캐시메모리보다 느리고 SSD보다 빠르다.
SSD: 비휘발성 보조기억장치로, 제시된 장치 중 접근 속도가 가장 느리다.
접근 속도는 일반적으로 레지스터 > 캐시메모리 > 주기억장치 > SSD 순이다.

NoteBase

선지별 해설

① X
가장 빠른 것을 캐시메모리, 가장 느린 것을 주기억장치라고 한 부분이 모두 틀렸다. 올바르게는 가장 빠른 것이 레지스터이고 가장 느린 것이 SSD이다.

② X
가장 느린 것이 SSD라는 부분은 맞지만, 가장 빠른 것이 캐시메모리라는 부분이 틀렸다. 올바르게는 가장 빠른 것이 레지스터이다.

③ X
가장 빠른 것이 레지스터라는 부분은 맞지만, 가장 느린 것이 주기억장치라는 부분이 틀렸다. 올바르게는 가장 느린 것이 SSD이다.

④ O
CPU 내부의 레지스터가 제시된 장치 중 가장 빠르고, 보조기억장치인 SSD가 가장 느리므로 옳다.

오답 포인트

캐시메모리가 주기억장치보다 빠르다는 사실만 떠올려 캐시메모리를 전체 장치 중 가장 빠르다고 고르기 쉽다. 그러나 CPU 내부의 레지스터가 캐시메모리보다 빠르다. 또한 SSD는 빠른 보조기억장치이지만 주기억장치보다 접근 속도가 느리다.

다시 나오면 이렇게 판단

CPU에 가까울수록 접근 속도가 빠르다는 저장계층의 원리를 적용한다. CPU 내부의 레지스터에서 시작하여 캐시메모리, 주기억장치, 보조기억장치 순으로 내려갈수록 접근 속도가 느려진다.

암기 포인트

속도 순서: 레지스터 > 캐시메모리 > 주기억장치 > SSD이다.

NoteBase

문 19. 다음 <보기> 중 화이트박스 테스트의 종류로 옳은 것은 모두 몇 개인가?

<보기>
㉠ 동등분할(equivalence partitioning)
㉡ 루프시험(loop test)
㉢ 기초경로시험(basic path test)
㉣ 조건시험(condition test)
㉤ 비교검사(comparison test)

① 1개
② 2개
③ 3개
④ 4개

정답 3번

출제 포인트

화이트박스 테스트와 블랙박스 테스트의 대표 기법을 구분하고, 화이트박스 테스트에 해당하는 항목의 개수를 판단하는 문제이다.

핵심 개념

화이트박스 테스트는 프로그램의 내부 구조와 제어 흐름을 바탕으로 경로, 조건, 반복 구조 등을 검사하는 구조적 테스트이다. 루프 시험, 기초 경로 시험, 조건 시험이 이에 해당한다.
블랙박스 테스트는 내부 구현을 보지 않고 입력과 출력의 관계 및 명세 충족 여부를 검사하는 기능적 테스트이다. 동등 분할과 비교 검사가 이에 해당한다.
따라서 화이트박스 테스트는 ㉡, ㉢, ㉣의 3개이다.

NoteBase

선지별 해설

① X
화이트박스 테스트는 1개가 아니라 ㉡ 루프 시험, ㉢ 기초 경로 시험, ㉣ 조건 시험의 3개이다. 틀린 수치인 '1개'는 '3개'로 고쳐야 한다.

② X
화이트박스 테스트에 해당하는 항목은 2개가 아니라 3개이다. 틀린 수치인 '2개'는 '3개'로 고쳐야 한다.

③ O
㉡은 반복 구조, ㉢은 제어 흐름의 독립 경로, ㉣은 논리 조건을 내부 구조에 근거하여 검사하므로 모두 화이트박스 테스트이다. 따라서 모두 3개이다.

④ X
㉠ 동등 분할과 ㉤ 비교 검사는 블랙박스 테스트이므로 화이트박스 테스트는 4개가 될 수 없다. 틀린 수치인 '4개'는 '3개'로 고쳐야 한다.

오답 포인트

㉤의 '비교 검사'라는 표현만 보고 프로그램 내부의 값을 비교하는 기법으로 오해하기 쉽다. 비교 검사는 동일한 입력에 대해 여러 구현의 출력 결과를 비교하는 블랙박스 테스트이고, 내부 제어 구조를 검사하는 기법이 아니다.

다시 나오면 이렇게 판단

경로·조건·루프처럼 프로그램 내부의 제어 구조를 직접 검사하면 화이트박스, 입력 영역을 나누거나 출력 결과를 비교하면 블랙박스로 판단한다.

암기 포인트

화이트박스는 루프·경로·조건, 블랙박스는 동등 분할·비교 검사이다.

NoteBase

문 20. 다음 <보기>에서 정의하는 용어를 가장 바르게 짝지은 것은?

<보기>
㉠ 공격자는 사용자의 합법적 도메인을 탈취하거나 도메인 네임 시스템(DNS) 또는 프락시 서버의 주소를 변조하여, 사용자가 진짜 사이트로 오인하여 접속하도록 유도한 후 개인정보를 훔친다.
㉡ 유명 은행이나 전자상거래업체 등의 홈페이지와 동일하게 보이는 위장 홈페이지를 만든 후, 인터넷 이용자에게 전자우편을 보내 위장 홈페이지를 접속하게 하여 개인정보를 입력하도록 유인하는 사기 수법이다.
㉢ 신뢰하는 시스템으로 가장하기 위해 고유 식별자를 위장하여 상대를 속이는 기법이다. 최근에는 공격자가 자신의 흔적을 노출하지 않기 위해 사용한다. 해킹 시 자신의 IP 주소, DNS 이름, MAC 주소 등과 같은 고유 식별자를 위장하여 역추적을 어렵게 한다.

① ㉠: 파밍(Pharming), ㉡: 피싱(Phishing), ㉢: 스푸핑(Spoofing)
② ㉠: 파밍(Pharming), ㉡: 피싱(Phishing), ㉢: 스니핑(Sniffing)
③ ㉠: 하이재킹(Hijacking), ㉡: 피싱(Phishing), ㉢: 스푸핑(Spoofing)
④ ㉠: 하이재킹(Hijacking), ㉡: 스미싱(Smishing), ㉢: 스니핑(Sniffing)

정답 1번

출제 포인트

공격 방식의 핵심 단서인 접속 경로 변조, 위장 홈페이지를 이용한 정보 입력 유도, 고유 식별자 위장을 각각 파밍·피싱·스푸핑과 연결하는 문제이다.

핵심 개념

파밍(Pharming): DNS나 프락시 서버의 주소 등을 변조하여 사용자가 정상 주소를 입력해도 가짜 사이트로 접속하게 하는 공격이다. 키워드는 '접속 경로·주소 변조'이다.
피싱(Phishing): 신뢰할 만한 기관을 사칭한 전자우편과 위장 홈페이지 등으로 사용자가 개인정보를 직접 입력하도록 속이는 수법이다. 키워드는 '사칭 메시지와 위장 사이트'이다.
스푸핑(Spoofing): IP 주소, DNS 이름, MAC 주소 등의 식별 정보를 위조하여 신뢰할 수 있는 시스템이나 사용자인 것처럼 가장하는 기법이다. 키워드는 '식별자 위장'이다.
스니핑(Sniffing)은 네트워크 패킷을 엿보는 행위이고, 하이재킹(Hijacking)은 세션이나 시스템의 제어권을 가로채는 행위이며, 스미싱(Smishing)은 문자메시지를 이용한 피싱이다.

NoteBase

선지별 해설

① O — ㉠은 DNS·프락시 서버 주소 등을 변조해 가짜 사이트로 유도하므로 파밍, ㉡은 전자우편과 위장 홈페이지로 개인정보 입력을 유도하므로 피싱, ㉢은 IP 주소·DNS 이름·MAC 주소 등 고유 식별자를 위장하므로 스푸핑이다. 세 용어가 모두 바르게 연결되었다.
② X — 틀린 부분은 ㉢을 스니핑으로 연결한 것이다. 스니핑은 네트워크상에서 오가는 패킷을 몰래 엿보는 행위이다. ㉢처럼 고유 식별자를 위장하는 기법은 스푸핑이므로 '㉠ 파밍, ㉡ 피싱, ㉢ 스푸핑'으로 고쳐야 한다.
③ X — 틀린 부분은 ㉠을 하이재킹으로 연결한 것이다. 하이재킹은 통신 세션이나 시스템의 제어권을 가로채는 공격이다. ㉠처럼 DNS나 프락시 서버 주소를 변조해 가짜 사이트로 유도하는 수법은 파밍이므로 '㉠ 파밍, ㉡ 피싱, ㉢ 스푸핑'으로 고쳐야 한다.
④ X — ㉠을 하이재킹, ㉡을 스미싱, ㉢을 스니핑으로 연결한 부분이 모두 틀렸다. ㉠은 접속 경로를 변조하는 파밍, ㉡은 전자우편과 위장 홈페이지를 이용하는 피싱, ㉢은 식별자를 위장하는 스푸핑이다. 따라서 '㉠ 파밍, ㉡ 피싱, ㉢ 스푸핑'으로 고쳐야 한다.

오답 포인트

㉠과 ㉡은 모두 가짜 사이트로 개인정보를 빼내지만, ㉠은 DNS·프락시 서버 주소 등 접속 경로의 변조가 핵심인 파밍이고 ㉡은 전자우편으로 위장 홈페이지 접속과 정보 입력을 유도하는 피싱이다. 또한 ㉢의 'IP·DNS·MAC 주소 위장'을 패킷 도청인 스니핑과 혼동하기 쉽다.

다시 나오면 이렇게 판단

주소·DNS 변조로 자동 유도되면 파밍, 기관 사칭 메시지와 가짜 사이트에서 입력을 유도하면 피싱, IP·DNS·MAC 등 식별자를 위조하면 스푸핑으로 판단한다. 패킷 엿보기는 스니핑, 세션 제어권 탈취는 하이재킹, 문자메시지를 이용한 피싱은 스미싱으로 구분한다.

암기 포인트

파밍은 경로 변조, 피싱은 입력 유도, 스푸핑은 신분 위장이다.

NoteBase