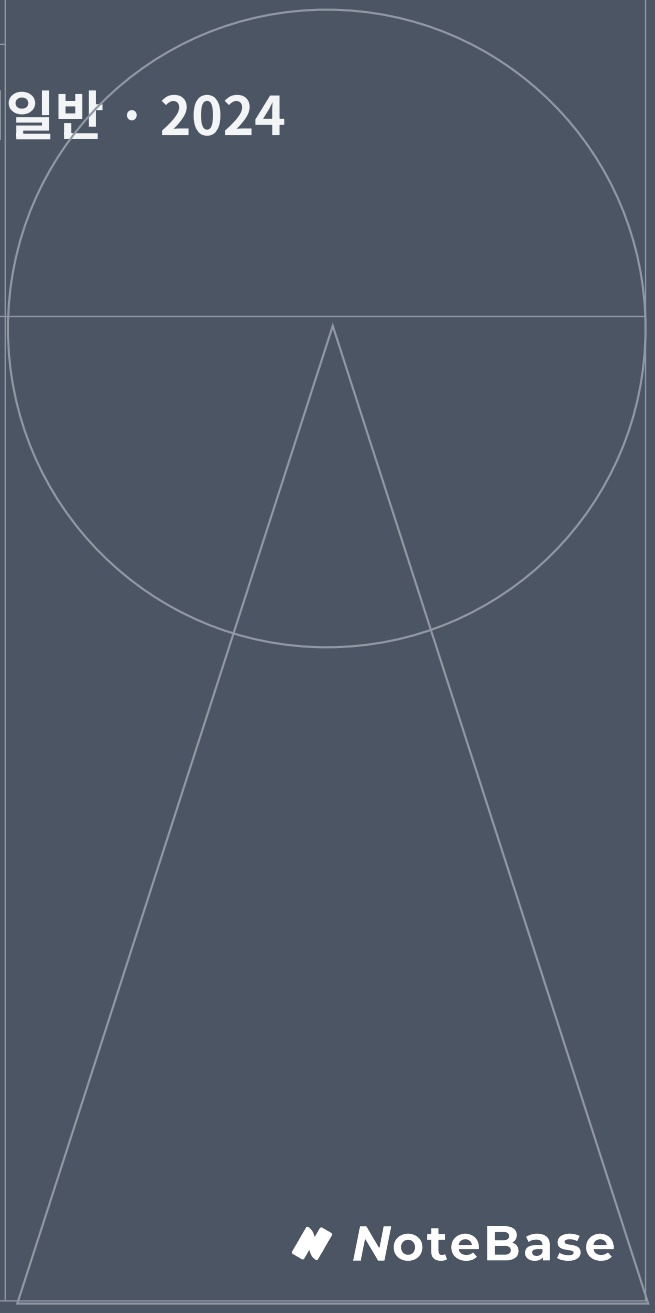
기출OX 해설집

공무원/군무원/9급 · 컴퓨터일반 · 2024



NoteBase

기출OX 해설집

공무원/군무원/9급 · 컴퓨터일반 · 2024

공무원/군무원/9급 · 컴퓨터일반 · 2024

1. 다음 운영체제의 프로세스 스케줄링 방법에 대한 설명 중 가장 적절하지 않은 것은?
   ① 라운드 로빈(Round Robin) 스케줄링은 여러 프로세스를 일정 순서에 따라 단위시간씩 실행시킨다.
   ② FCFS(First Come First Serve) 스케줄링은 프로세스의 평균 대기 시간을 최적화하는 실행 방법이다.
   ③ 우선순위 스케줄링은 프로세스별 등급에 따라 높은 순위를 먼저 실행시킨다.
   ④ 다단계 큐 스케줄링은 상위 단계 큐의 프로세스를 하위 단계 큐의 프로세스보다 먼저 실행시킨다.

정답 2번

출제 포인트

라운드 로빈, FCFS, 우선순위, 다단계 큐 스케줄링의 처리 순서와 핵심 특성을 구별하는 문제이다.

핵심 개념

라운드 로빈(Round Robin): 준비 큐의 프로세스에 일정한 시간 할당량을 차례로 주며 순환 실행하는 선점형 방식이다.
FCFS(First Come First Served): 준비 큐에 먼저 도착한 프로세스부터 실행하는 비선점형 방식이다. 짧은 작업이 긴 작업 뒤에서 오래 기다리는 호위 효과가 생길 수 있으므로 평균 대기 시간을 최소화하지 않는다.
우선순위 스케줄링: 각 프로세스에 부여된 우선순위에 따라 우선순위가 높은 프로세스를 먼저 실행하는 방식이다.
다단계 큐 스케줄링: 프로세스를 특성에 따라 여러 준비 큐로 나누고 큐마다 별도의 스케줄링 방식을 적용하며, 큐 사이에는 우선순위를 둘 수 있다.

NoteBase

선지별 해설

① O — 라운드 로빈은 여러 프로세스에 일정한 시간 할당량을 차례로 부여하고, 시간 안에 끝나지 않은 프로세스를 준비 큐의 뒤로 보내 순환 실행한다.
② X — 틀린 표현은 'FCFS 스케줄링은 프로세스의 평균 대기 시간을 최적화한다'이다. FCFS는 도착 순서대로 처리할 뿐 평균 대기 시간을 최소화하지 않으며, 실행 시간이 긴 프로세스가 앞에 오면 뒤의 짧은 프로세스들이 오래 기다릴 수 있다. 올바르게는 '실행 시간이 가장 짧은 프로세스를 먼저 선택하는 SJF 스케줄링이 각 실행 시간을 알고 있다는 조건에서 평균 대기 시간을 최소화한다'이다.
③ O — 우선순위 스케줄링은 프로세스별 우선순위를 비교하여 우선순위가 높은 프로세스를 먼저 실행한다.
④ O — 다단계 큐 스케줄링은 준비 큐를 여러 단계로 나누며, 큐 사이에 고정 우선순위를 적용하는 경우 상위 우선순위 큐의 프로세스를 하위 우선순위 큐의 프로세스보다 먼저 실행한다.

오답 포인트

②는 '먼저 온 프로세스를 먼저 처리한다'는 FCFS의 공정하고 단순한 규칙을 '평균 대기 시간 최적화'와 혼동하게 만든다. 평균 대기 시간 최소화는 FCFS가 아니라 SJF의 대표적 성질이다.

다시 나오면 이렇게 판단

스케줄링 설명에서 무엇을 기준으로 다음 프로세스를 고르는지 확인한다. 도착 순서는 FCFS, 짧은 실행 시간은 SJF, 시간 할당량의 순환은 라운드 로빈, 등급은 우선순위, 여러 고정 큐의 구분은 다단계 큐에 대응한다.

암기 포인트

FCFS는 도착 순서, SJF는 짧은 작업 우선과 평균 대기 시간 최소화, RR은 시간 할당량 순환이다.

NoteBase

2. 다음 무선 전송 규격 중 2.4GHz와 5GHz 주파수 대역을 모두 지원하는 것은?

① 802.11a ② 802.11g

③ 802.11n ④ 802.11ac

정답 3번

출제 포인트

IEEE 802.11 무선 LAN 규격이 지원하는 주파수 대역을 구분하는 문제이다.

핵심 개념

802.11a와 802.11ac는 5*GHz* 대역을 사용하고, 802.11g는 2.4*GHz* 대역을 사용한다. 802.11n은 2.4*GHz*와 5*GHz* 두 대역을 모두 지원하는 규격이다.

선지별 해설

① X — 802.11a는 5*GHz* 대역만 지원하므로 두 대역을 모두 지원하지 않는다. 틀린 표현은 '2.4*GHz*와 5*GHz* 주파수 대역을 모두 지원'이며, 올바르게는 '5*GHz* 대역을 지원'이다.
② X — 802.11g는 2.4*GHz* 대역만 지원한다. 틀린 표현은 '2.4*GHz*와 5*GHz* 주파수 대역을 모두 지원'이며, 올바르게는 '2.4*GHz* 대역을 지원'이다.
③ O — 802.11n은 2.4*GHz*와 5*GHz* 대역을 모두 지원하므로 문제의 조건에 맞는다.
④ X — 802.11ac는 5*GHz* 대역을 사용하므로 두 대역을 모두 지원하는 규격이 아니다. 틀린 표현은 '2.4*GHz*와 5*GHz* 주파수 대역을 모두 지원'이며, 올바르게는 '5*GHz* 대역을 지원'이다.

오답 포인트

802.11ac가 비교적 최신 규격이라는 이유만으로 두 대역을 모두 지원한다고 판단하기 쉽다. 그러나 이 문제에서는 세대가 아니라 규격 자체의 주파수 대역을 기준으로 구분하며, 두 대역을 모두 지원하는 것은 802.11n이다.

다시 나오면 이렇게 판단

먼저 각 규격을 주파수 대역으로 분류한다. 802.11g는 2.4*GHz*, 802.11a와 802.11ac는 5*GHz*, 802.11n은 2.4*GHz*와 5*GHz* 모두에 해당한다.

암기 포인트

g는 2.4*GHz*, a·ac는 5*GHz*, n은 두 대역 모두이다.

NoteBase

3. 다음 암호화에 관련된 설명 중 가장 적절하지 않은 것은?
① 단방향 암호화는 암호화된 결과로부터 원문을 복호화하는 용도로 사용하지 않는다.
② 대칭키 암호화는 키를 가진 경우 원문 복호화가 가능하다.
③ 공개키 암호화는 개인키를 이용한 원문 복호화가 불가능하다.
④ 공개키 암호화 기법은 비밀번호 없는 SSH (Secure Shell) 접속에 응용된다.

정답 3번

출제 포인트

단방향 암호화(해시), 대칭키 암호화, 공개키 암호화의 복호화 가능 여부와 공개키 방식의 SSH 활용을 구별하는 문제이다.

핵심 개념

단방향 암호화(해시)는 입력값을 고정된 형태의 해시값으로 바꾸며, 그 결과에서 원문을 되돌리는 복호화를 전제로 하지 않는다.
대칭키 암호화는 암호화와 복호화에 같은 비밀키를 사용하는 방식이다.
공개키 암호화는 공개키와 개인키의 쌍을 사용하는 방식으로, 공개키로 암호화한 데이터는 대응하는 개인키로 복호화할 수 있다. SSH의 공개키 인증에서는 서버에 공개키를 두고 사용자가 개인키 보유를 증명하여 비밀번호 입력 없이 인증할 수 있다.

선지별 해설

① O — 단방향 암호화는 암호화된 결과로부터 원문을 복호화하는 용도로 사용하지 않는다. 해시값은 입력값의 일치 여부나 무결성을 확인하는 데 활용된다.
② O — 대칭키 암호화에서는 암호화에 사용한 비밀키를 가진 경우 그 키로 암호문을 복호화할 수 있다.
③ X — '개인키를 이용한 원문 복호화가 불가능하다'는 부분이 틀렸다. 올바르게는 공개키로 암호화한 데이터는 대응하는 개인키를 이용해 복호화할 수 있다.
④ O — SSH는 공개키와 개인키의 쌍을 이용한 공개키 인증을 지원하므로, 공개키가 서버에 등록되어 있고 사용자가 대응하는 개인키를 보유하면 비밀번호 입력 없는 접속에 활용할 수 있다.

NoteBase

오답 포인트

③은 공개키가 공개되어 있다는 사실을 '개인키로도 복호화할 수 없다'는 뜻으로 바꾸어 놓았다. 공개키 방식의 핵심은 공개키와 개인키가 서로 대응한다는 점이며, 공개키로 암호화한 데이터의 복호화에는 대응하는 개인키가 사용된다.

다시 나오면 이렇게 판단

복호화 가능 여부를 기준으로 구분한다. 해시는 원문 복원을 전제로 하지 않고, 대칭키 방식은 같은 비밀키로 암호화·복호화하며, 공개키 방식은 서로 대응하는 공개키·개인키 쌍을 사용한다. SSH의 비밀번호 없는 접속은 공개키 암호화 자체로 비밀번호를 바꾸는 것이 아니라 개인키 보유를 확인하는 공개키 인증의 활용이다.

암기 포인트

해시는 복호화 없음, 대칭키는 같은 키, 공개키 방식은 공개키·개인키 한 쌍, SSH 무암호 접속은 공개키 인증이다.

NoteBase

4. 다음 중 하이퍼바이저(Hypervisor)에 관련된 설명으로 가장 적절하지 않은 것은?
   ① 가상머신을 동작시키기 위한 기반 소프트웨어이다.
   ② KVM(Kernel-based Virtual Machine)은 리눅스 커널에 통합된 하이퍼바이저로, 가상머신의 게스트(Guest) 운영체제는 리눅스로 제한된다.
   ③ 가상머신이 사용하는 하드웨어 자원을 관리, 할당한다.
   ④ 한 하이퍼바이저 위에서 각기 다른 운영체제의 가상머신 병행 실행이 가능하다.

정답 2번

출제 포인트

하이퍼바이저의 역할과 KVM에서 호스트 운영체제와 게스트 운영체제를 구분할 수 있는지를 묻는 문제이다.

핵심 개념

하이퍼바이저는 물리적 하드웨어 자원을 가상화하여 여러 가상머신을 생성·실행하고, 각 가상머신에 자원을 배분·관리하는 기반 소프트웨어이다.
KVM은 Linux 커널에 통합된 하이퍼바이저이지만, 그 위의 가상머신에서 실행되는 게스트 운영체제가 Linux로만 제한되는 것은 아니다.

NoteBase

선지별 해설

① O — 하이퍼바이저는 가상머신을 생성하고 동작시키는 가상화 기반 소프트웨어이므로 적절한 설명이다.

② X — 틀린 부분은 '게스트 운영체제는 Linux로 제한된다'이다. KVM이 Linux 커널에 통합되어 있다는 것은 호스트 측 구현에 관한 설명이며, 게스트 운영체제로는 Linux뿐 아니라 Windows 등도 실행할 수 있다. 올바르게는 'KVM은 Linux 커널에 통합된 하이퍼바이저이며, 게스트 운영체제는 Linux로만 제한되지 않는다'이다.

③ O — 하이퍼바이저는 CPU, 메모리, 저장장치 등의 물리적 하드웨어 자원을 가상화하고 각 가상머신에 할당·관리한다.

④ O — 하이퍼바이저는 가상머신을 서로 격리하여 한 물리 시스템에서 서로 다른 운영체제를 사용하는 여러 가상머신을 동시에 실행할 수 있게 한다.

오답 포인트

②의 'Linux 커널에 통합'이라는 표현을 '게스트 운영체제도 Linux만 가능'하다는 뜻으로 오해하기 쉽다. Linux는 KVM이 통합된 호스트 측 기반이고, 가상머신 안에서 실행되는 게스트 운영체제의 종류와는 구분해야 한다.

다시 나오면 이렇게 판단

하이퍼바이저 문항에서는 먼저 호스트와 게스트를 구분한다. 하이퍼바이저가 설치되거나 통합된 운영체제는 호스트 측 조건이고, 가상머신 내부의 운영체제는 게스트이므로 두 조건을 동일하게 제한하는 선지는 틀린 설명으로 판단한다.

암기 포인트

KVM은 Linux 커널 기반이지만, 게스트 운영체제는 Linux 전용이 아니다.

NoteBase

5. 다음 중 EPROM(Erasable Programmable ROM)에 대한 설명으로 가장 적절하지 않은 것은?
① 비휘발성 메모리이다.
② 데이터 갱신 횟수의 제한이 없다.
③ 데이터를 갱신하려면 저장되어 있는 데이터를 지워야 한다.
④ 플래시 메모리는 EPROM의 한 종류이다.

정답 2번

출제 포인트

EPROM의 비휘발성, 소거 후 재기록 방식, 기록·소거 횟수의 한계를 구분하는 문제이다.

핵심 개념

EPROM은 전원이 꺼져도 내용이 유지되며, 저장된 내용을 소거한 뒤 다시 기록할 수 있는 비휘발성 ROM이다. 다만 소거와 기록을 반복할 수 있는 횟수에는 물리적인 한계가 있다. 플래시 메모리는 전기적으로 소거·재기록하는 EEPROM 계열이며, 이 문항에서는 소거 후 재기록 가능한 ROM의 넓은 분류에 포함된다.

선지별 해설

① O — EPROM은 전원을 차단해도 저장 내용이 사라지지 않는 비휘발성 메모리이다.
② X — 틀린 표현은 '데이터 갱신 횟수의 제한이 없다'이다. 메모리 셀은 소거와 기록을 반복하면서 열화되므로 갱신 가능 횟수에는 한계가 있다. 올바르게는 '데이터 갱신 횟수에는 제한이 있다'이다.
③ O — EPROM의 데이터를 갱신하려면 기존 내용을 먼저 소거한 뒤 새로운 데이터를 기록해야 한다. 전통적인 EPROM은 자외선으로 저장 내용을 소거한 후 다시 프로그램한다.
④ O — 플래시 메모리는 전기적으로 소거하고 다시 기록할 수 있는 EEPROM 계열의 비휘발성 메모리이며, 이 문항의 넓은 EPROM 분류에 속한다.

오답 포인트

'재기록할 수 있다'는 성질을 '횟수 제한 없이 무한히 재기록할 수 있다'로 확대하면 안 된다. 재기록 가능 여부와 소거·기록 내구성의 한계는 서로 다른 개념이다.

다시 나오면 이렇게 판단

EPROM 관련 선지는 먼저 비휘발성인지, 기존 데이터를 소거한 뒤 재기록하는지, 소거·기록 횟수에 한계가 있는지를 차례로 확인한다. '무제한', '영구적으로 반복 가능' 같은 절대 표현이 나오면 내구성 한계를 점검한다.

NoteBase

암기 포인트

EPROM은 비휘발성이고 소거 후 재기록할 수 있지만, 갱신 횟수는 무제한이 아니다.

NoteBase

6. 다음 중 파일의 허가권을 설정하는 리눅스 명령어에 대한 설명으로 가장 적절한 것은?

```
chmod 640 sample
```

① chmod 명령어는 루트 사용자만이 사용할 수 있다.
② sample 파일의 소유자는 sample 파일에 대해 읽기, 쓰기, 실행하기가 가능하다.
③ sample 파일의 그룹에 속한 사용자는 sample 파일에 대해 읽기만 가능하다.
④ sample 파일의 모든 사용자는 읽기가 가능하다.

정답 3번

출제 포인트

chmod의 숫자 모드에서 소유자·그룹·기타 사용자의 읽기, 쓰기, 실행 권한을 해석할 수 있는지를 묻는 문제이다.

핵심 개념

chmod 640 sample에서 세 숫자는 차례로 파일 소유자, 파일 소유 그룹, 기타 사용자의 권한을 뜻한다.
읽기(r)는 4, 쓰기(w)는 2, 실행(x)은 1이므로 6은 읽기+쓰기(rw-), 4는 읽기만(r--), 0은 아무 권한 없음(---)이다.
따라서 sample의 권한은 소유자 rw-, 그룹 r--, 기타 사용자 ---이다.

선지별 해설

① X — '루트 사용자만 사용할 수 있다'가 틀렸다. chmod는 파일 소유자가 자신의 파일 권한을 변경할 때 사용할 수 있으며, 루트 사용자도 권한을 변경할 수 있다.
② X — '읽기, 쓰기, 실행하기가 가능하다'에서 실행하기가 틀렸다. 소유자 권한 6은 4+2이므로 읽기와 쓰기만 가능하고 실행은 불가능하다.
③ O — 그룹 권한 4는 읽기 권한만 뜻하므로 sample 파일의 그룹에 속한 사용자는 읽기만 가능하다.
④ X — '모든 사용자는 읽기가 가능하다'가 틀렸다. 기타 사용자 권한은 0이므로 파일 소유자도 아니고 소유 그룹에도 속하지 않은 사용자는 읽기를 포함한 어떤 권한도 없다.

NoteBase

오답 포인트

②에서 6을 읽기·쓰기·실행 권한으로 오해하거나, ④에서 소유자와 그룹에 읽기 권한이 있다는 이유로 기타 사용자까지 읽을 수 있다고 확대 해석하기 쉽다. 각 숫자는 서로 다른 사용자 범주의 권한을 나타낸다.

다시 나오면 이렇게 판단

세 자리 숫자를 소유자-그룹-기타 사용자 순서로 나눈 뒤, 각 숫자를 4(읽기), 2(쓰기), 1(실행)의 합으로 분해하여 판단한다.

암기 포인트

chmod 숫자 권한은 사용자 순서가 소유자-그룹-기타이고, 권한 값은 읽기 4·쓰기 2·실행 1이다. 640은 rw- / r-- / ---이다.

NoteBase

7. TCP/IP 통신 방식에 대한 설명으로 가장 적절하지 않은 것은?
① TCP는 연결 생성, 데이터 전송, 연결 종료의 세 단계 절차로 진행된다.
② 데이터그램(Datagram) 방식은 목적지까지 고정된 경로 없이 패킷단위로 전송하는 방식이다.
③ 가상 회선(Virtual Circuit)과 데이터그램 방식은 전송 계층(Transport layer)의 프로토콜이다.
④ IP는 TCP보다 상위에 위치하여 패킷의 전송에 대응하는 계층이다.

정답 4번

출제 포인트

TCP/IP에서 TCP와 IP의 역할 및 계층 관계, 그리고 가상 회선 방식과 데이터그램 방식의 차이를 구분하는 문제이다.

핵심 개념

TCP: 전송 계층에서 연결 설정, 신뢰성 있는 데이터 전송, 연결 종료를 담당하는 연결 지향 프로토콜이다.
IP: TCP보다 하위인 인터넷 계층에서 주소 지정과 패킷 전달을 담당하는 비연결형 프로토콜이다.
가상 회선 방식은 논리적 연결을 먼저 설정한 뒤 같은 경로로 패킷을 보내고, 데이터그램 방식은 고정 경로를 미리 설정하지 않고 각 패킷을 독립적으로 전송한다. 이 둘은 일반적으로 네트워크 계층의 서비스 또는 패킷 교환 방식으로 다루며, TCP·UDP와 같은 전송 계층 프로토콜과는 구별된다.

선지별 해설

① O - TCP 통신은 연결 설정, 데이터 전송, 연결 종료의 세 단계로 이루어진다.
② O - 데이터그램 방식은 목적지까지 고정된 경로를 미리 설정하지 않으며, 각 패킷이 독립적으로 전달된다.
③ X - 가상 회선과 데이터그램을 '전송 계층의 프로토콜'이라고 한 부분이 부정확하다. 일반적인 계층 구분에서는 네트워크 계층의 서비스 또는 패킷 교환 방식이며, TCP·UDP와 같은 전송 계층 프로토콜이 아니다.
④ X - 틀린 부분은 'IP는 TCP보다 상위에 위치'한다는 표현이다. 올바르게는 TCP가 전송 계층에 있고 IP가 그 아래 인터넷 계층에 있으므로, IP는 TCP보다 하위에서 패킷의 주소 지정과 전달을 담당한다. 공식 채점 답은 ④이지만, 엄밀한 계층 구분에 따르면 ③도 부적절하여 문항에는 복수 오답 가능성이 있다.

NoteBase

오답 포인트

‘IP가 패킷 전송을 담당한다’는 뒷부분만 보고 ④를 맞는 설명으로 판단하기 쉽다. 그러나 앞부분의 계층 관계가 반대이다. TCP는 IP 위에서 동작하며, 데이터그램의 무경로 설정 특성과 TCP/IP 계층 순서를 서로 다른 판단 기준으로 보아야 한다.

다시 나오면 이렇게 판단

먼저 계층 순서를 ‘응용 계층－전송 계층(TCP)－인터넷 계층(IP)－네트워크 접근 계층’으로 확인한다. 이어서 가상 회선은 사전 연결 설정, 데이터그램은 고정 경로 없는 독립 패킷 전송이라는 기준으로 판단한다.

암기 포인트

TCP는 IP 위에서 연결과 신뢰성을 담당하고, IP는 아래에서 주소 지정과 패킷 전달을 담당한다. 가상 회선은 연결 후 전송, 데이터그램은 연결 없이 독립 전송이다.

NoteBase

8. 다음 중 아래 데이터베이스 모델링에 대한 설명으로 가장 적절한 것은?

> 국방전자는 배터리를 이용하는 다양한 전자제품을 생산하고 있다. 각 제품이 동작하는 전압은 배터리 사용 개수에 따라 달려 있으므로, 제품의 동작 전압 규격은 배터리 연결로 만들어지는 특정 전압값으로만 가능하다. 제품과 동작 전압을 데이터베이스화할 때 전압값이 가질 수 있는 제한 조건을 정의하는 것이다.

① 시스템 카탈로그(System Catalog)
② 도메인(Domain)
③ 일반 집합 연산자
④ 엔터티(Entity)

정답 2번

출제 포인트

이 문제는 데이터베이스에서 한 속성이 가질 수 있는 값의 범위와 제약을 나타내는 도메인의 개념을 묻는 문제이다.

핵심 개념

도메인(Domain)은 하나의 속성이 가질 수 있는 모든 허용 가능한 값의 집합이다. 데이터형, 자릿수, 값의 범위, 형식 등의 조건으로 속성값을 제한한다.
시스템 카탈로그는 데이터베이스 객체와 구조에 관한 메타데이터를 저장하고, 일반 집합 연산자는 합집합·교집합·차집합 등의 집합 처리를 수행하며, 엔터티는 현실 세계에서 독립적으로 식별할 수 있는 대상이다.

NoteBase

선지별 해설

① X — 시스템 카탈로그(System Catalog)는 테이블, 속성, 제약조건 등 데이터베이스의 정의 정보를 저장하는 메타데이터 저장소이므로, 전압값 자체의 허용 범위를 뜻하지 않는다. 틀린 표현은 '시스템 카탈로그'이며, 올바르게는 '도메인'이다.
② O — 도메인(Domain)은 특정 속성이 가질 수 있는 허용값의 집합이다. 동작 전압 속성을 배터리 연결로 만들 수 있는 특정 전압값으로 제한하는 조건에 해당한다.
③ X — 일반 집합 연산자는 합집합, 교집합, 차집합처럼 집합을 결합하거나 비교하는 연산이므로 속성값의 허용 범위를 정의하는 개념이 아니다. 틀린 표현은 '일반 집합 연산자'이며, 올바르게는 '도메인'이다.
④ X — 엔터티(Entity)는 제품처럼 독립적으로 식별하고 정보를 저장할 현실 세계의 대상이므로, 동작 전압 속성에 허용되는 값의 범위를 뜻하지 않는다. 틀린 표현은 '엔터티'이며, 올바르게는 '도메인'이다.

오답 포인트

제품은 엔터티가 될 수 있으므로 ④를 고르기 쉽지만, 문제는 제품 자체가 아니라 '동작 전압값이 가질 수 있는 제한 조건'을 묻는다. 값의 허용 범위라는 표현이 도메인을 가리킨다.

다시 나오면 이렇게 판단

문장에서 '속성이 가질 수 있는 값의 범위·형식·제한 조건'을 찾으면 도메인으로 판단한다. '관리 대상'은 엔터티, '데이터베이스 정의 정보'는 시스템 카탈로그, '집합의 결합·비교'는 집합 연산자로 구분한다.

암기 포인트

도메인은 속성값의 허용 범위이다. 엔터티는 대상, 시스템 카탈로그는 정의 정보, 집합 연산자는 집합 처리이다.

NoteBase

9. 다음 중 문자 인코딩 방법에 대한 설명으로 가장 적절한 것은?

① 아스키(ASCII) 코드는 8비트로 구성되어 2^8(=256)개의 문자 표현이 가능하다.

② EBCDIC(Entended Binary-Coded Decimal Interchange Code)는 7비트로 구성되어 2^7(=128)개의 문자 표현이 가능하다.

③ BCD(Binary-Coded Decimal) 코드는 5비트로 구성되어 2^5(=32)개의 문자 표현이 가능하다.

④ 유니코드(Unicode)는 16비트로 구성되어 2^{16}(=65,536)개의 문자 표현이 가능하다.

정답 4번

출제 포인트

ASCII, EBCDIC, BCD, Unicode의 기본 비트 수와 표현 범위를 구별하는 문제이다.

핵심 개념

ASCII는 영문자·숫자·기호 등을 표현하는 표준 7비트 문자 코드로 128개를 나타낸다. EBCDIC은 주로 IBM 대형 컴퓨터에서 사용된 8비트 문자 코드로 256개를 나타낸다. BCD는 십진수 한 자리를 4비트 이진수로 표현하는 코드로, 유효한 값은 0000부터 1001까지의 10개이다. Unicode는 여러 언어의 문자를 통일하여 표현하기 위한 국제 문자 코드 체계이며, 이 문항의 교과서적 분류에서는 16비트로 $2^{16} = 65,536$개를 표현하는 것으로 본다.

선지별 해설

① X — 틀린 표현은 '8비트'와 '$2^8 = 256$개'이다. 표준 ASCII는 7비트로 구성되어 $2^7 = 128$개를 표현한다. 8비트 256개는 확장 ASCII에 해당한다.

② X — 틀린 표현은 '7비트'와 '$2^7 = 128$개'이다. EBCDIC은 8비트로 구성되어 $2^8 = 256$개를 표현한다.

③ X — 틀린 표현은 '5비트'와 '$2^5 = 32$개'이다. BCD는 십진수 한 자리를 4비트로 표현하며, 16가지 비트 조합 중 0000부터 1001까지의 10가지만 사용한다.

④ O — 이 문항의 기본 분류 기준에서 Unicode는 16비트 코드로 설명되며, $2^{16} = 65,536$개의 문자를 표현할 수 있으므로 가장 적절하다.

NoteBase

오답 포인트

①의 ASCII와 확장 ASCII를 혼동하거나, ②의 EBCDIC 비트 수와 ASCII 비트 수를 서로 바꾸어 기억하기 쉽다. 또한 BCD의 4비트가 만드는 16가지 조합을 모두 사용하는 것이 아니라 십진 숫자 0부터 9까지 10개만 사용한다는 점이 함정이다.

다시 나오면 이렇게 판단

먼저 각 코드의 기본 비트 수를 대응시킨다. 표준 ASCII는 7비트, EBCDIC은 8비트, BCD는 십진 한 자리당 4비트, 이 문항의 교과서적 Unicode 분류는 16비트이다. 그다음 표현 수가 비트 수에 맞는지 확인하고, BCD는 유효한 십진 숫자가 10개뿐이라는 예외를 적용한다.

암기 포인트

ASCII 7비트·128개, EBCDIC 8비트·256개, BCD 4비트·10개, 교과서적 Unicode 16비트·65,536개로 연결한다.

NoteBase

10. 다음 중 논리회로에 대한 설명으로 가장 적절하지 않은 것은?

① 반가산기(half adder)는 두 개의 입력과 두 개의 출력으로 합(sum)과 자리올림(carry)을 얻는다.

② 멀티플렉서(multiplexer)는 여러 개의 입력 중 하나만 출력에 전달해준다.

③ T 플립플롭(flip-flop)은 입력 신호가 0이면 출력 상태가 반전된다.

④ D 플립플롭(flip-flop)은 입력 신호가 0이면 출력 상태가 0이다.

정답 3번

출제 포인트

반가산기와 멀티플렉서의 기능, T·D 플립플롭의 입력에 따른 다음 출력 상태를 구별하는 문제이다.

핵심 개념

반가산기: 두 개의 1비트 입력을 더하여 합과 자리올림을 출력하는 조합논리회로이다.
멀티플렉서: 선택 신호에 따라 여러 입력 중 하나를 골라 하나의 출력으로 전달하는 조합논리회로이다.
T 플립플롭: 유효한 클록 시점에 T=0이면 현재 상태를 유지하고, T=1이면 현재 상태를 반전한다.
D 플립플롭: 유효한 클록 시점에 입력 D의 값을 다음 출력 상태로 저장한다.

선지별 해설

① O — 반가산기는 두 입력 비트를 더해 합(sum)과 자리올림(carry)이라는 두 출력을 얻는 회로이다.
② O — 멀티플렉서는 선택 신호가 지정한 여러 입력 중 하나만 출력으로 전달한다.
③ X — 틀린 표현은 '입력 신호가 0이면 출력 상태가 반전된다'이다. T 플립플롭은 T=0이면 현재 출력 상태를 유지하고, T=1이면 출력 상태가 반전된다.
④ O — D 플립플롭은 유효한 클록 시점에 D 입력을 저장하므로 D=0이면 다음 출력 상태는 0이 된다.

오답 포인트

T 플립플롭에서 T는 반전 여부를 정하는 입력이다. ③은 T=0과 T=1의 동작을 바꾸어 제시한 함정이다. 또한 플립플롭의 상태 변화는 유효한 클록 시점을 기준으로 판단한다.

NoteBase

다시 나오면 이렇게 판단

조합논리회로는 입력에 따른 기능을 확인하고, 플립플롭은 유효한 클록 시점의 다음 상태를 확인한다. T 플립플롭은 0이면 유지·1이면 반전, D 플립플롭은 입력값을 그대로 저장하는 것으로 판단한다.

암기 포인트

T: 0 유지, 1 반전 / D: 입력값 저장 / MUX: 여러 입력 중 하나 선택

NoteBase

11. 다음 중 캐시 메모리로 사용되는 것으로 가장 적절한 것은?
① SRAM(Static RAM)
② DRAM(Dynamic RAM)
③ PRAM(Phase-change RAM)
④ MRAM(Magnetic RAM)

정답 1번

출제 포인트

캐시 메모리에 요구되는 빠른 접근 속도와 리프레시 불필요성에 가장 잘 맞는 기억장치를 구별하는 문제이다.

핵심 개념

SRAM은 플립플롭 방식으로 비트를 저장하여 전원이 공급되는 동안 리프레시 없이 데이터를 유지하는 휘발성 메모리이며, 속도가 빨라 캐시 메모리에 주로 사용된다.
DRAM은 축전기에 전하를 저장하므로 주기적인 리프레시가 필요하고, 집적도가 높고 단가가 낮아 주기억장치에 주로 사용된다.
PRAM은 물질의 상변화에 따른 저항 차이를 이용하는 비휘발성 메모리이고, MRAM은 자기 상태를 이용하는 비휘발성 메모리이다.

선지별 해설

① O — SRAM은 리프레시가 필요 없고 접근 속도가 빨라 CPU와 주기억장치 사이의 속도 차이를 줄이는 캐시 메모리로 가장 적절하다.
② X — 틀린 부분은 DRAM을 캐시 메모리에 가장 적절한 장치로 본 점이다. DRAM은 주기적인 리프레시가 필요하고 SRAM보다 느리며, 올바르게는 대용량 주기억장치에 주로 사용된다.
③ X — 틀린 부분은 PRAM을 일반적인 캐시 메모리로 본 점이다. PRAM은 상변화 물질의 저항 차이를 이용하는 비휘발성 메모리이며, 올바르게는 비휘발성 저장장치 계열의 메모리로 분류한다.
④ X — 틀린 부분은 MRAM을 이 문항에서 묻는 대표적인 캐시 메모리로 본 점이다. MRAM은 자기 상태를 이용하는 비휘발성 메모리이며, 올바르게는 일반적인 캐시 메모리에는 SRAM이 사용된다고 판단한다.

오답 포인트

PRAM과 MRAM도 빠른 차세대 메모리로 언급될 수 있어 혼동하기 쉽다. 그러나 이 문항의 함정은 새로운 메모리 기술의 가능성이 아니라 교과서적인 메모리 용도를 묻는 데 있으며, 캐시는 SRAM, 주기억장치는 DRAM이라는 대응이 핵심이다.

NoteBase

다시 나오면 이렇게 판단

캐시 메모리를 물으면 먼저 속도가 빠르고 리프레시가 필요 없는지를 확인한다. 이 조건에 맞는 대표 장치는 SRAM이며, DRAM은 주기억장치, PRAM과 MRAM은 비휘발성 메모리로 구분한다.

암기 포인트

캐시는 SRAM, 주기억장치는 DRAM이다. S는 Speed와 연결해 기억한다.

NoteBase

12. 다음 python 프로그램의 실행 결과로 옳은 것은?

```
str = "89점"
try :
    score = int(str)
except ValueError :
    print("값 에러")
except TypeError :
    print("타입 에러")
else :
    print(score)
finally :
    print("마칩니다.")
```

① 값 에러
마칩니다.
② 타입 에러
마칩니다.
③ 89
④ 89
마칩니다.

정답 1번

출제 포인트

문자열을 정수로 변환할 때 발생하는 예외의 종류와 try-except-else-finally문의 실행 흐름을 묻는 문제이다.

핵심 개념

ValueError: 자료형은 변환 대상이 될 수 있지만 값의 형식이 올바르지 않을 때 발생하는 예외이다.
int("89점")은 문자열 안의 '점' 때문에 정수로 변환할 수 없어 ValueError가 발생한다.
TypeError: 함수가 처리할 수 없는 자료형의 인수를 받았을 때 발생하는 예외이다.
else 블록은 try 블록에서 예외가 발생하지 않았을 때만 실행된다. finally 블록은 예외 발생 여부와 관계없이 실행된다.

NoteBase

선지별 해설

① O — int("89점")에서 ValueError가 발생하므로 첫 번째 except 블록이 '값 에러'를 출력하고, 이어서 finally 블록이 '마칩니다.'를 출력한다.
② X — 틀린 부분은 '타입 에러'가 출력된다는 것이다. 인수의 자료형이 문자열이라는 사실 자체가 문제가 아니라 문자열에 숫자가 아닌 '점'이 포함된 것이 문제이므로 TypeError가 아닌 ValueError가 발생한다. 올바르게는 '값 에러'와 '마칩니다.'가 차례로 출력된다.
③ X — 틀린 부분은 '89'만 출력된다는 것이다. 정수 변환이 실패하므로 else 블록의 print(score)는 실행되지 않으며, 올바르게는 '값 에러'와 '마칩니다.'가 차례로 출력된다.
④ X — 틀린 부분은 '89'가 출력된다는 것이다. '89점' 전체를 int로 변환할 수 없으므로 score에 89가 저장되지 않는다. 올바르게는 '값 에러'와 '마칩니다.'가 차례로 출력된다.

오답 포인트

문자열을 int에 전달했다는 이유만으로 TypeError라고 판단하거나, 문자열 앞부분의 숫자 89만 자동으로 변환된다고 판단하기 쉽다. int는 숫자 형식의 문자열은 처리하지만 '89점'처럼 숫자가 아닌 문자가 섞인 문자열에는 ValueError를 발생시킨다. 또한 finally는 예외가 처리된 뒤에도 반드시 실행된다.

다시 나오면 이렇게 판단

먼저 try 블록에서 예외가 발생하는지와 그 종류를 판단한다. 예외가 발생하면 일치하는 except만 실행하고 else는 건너뛴다. 마지막으로 finally는 예외 발생 여부와 관계없이 실행된다고 판단한다.

암기 포인트

값의 형식이 잘못되면 ValueError, 자료형 자체가 맞지 않으면 TypeError, 예외가 없을 때만 else, 항상 실행되는 블록은 finally이다.

NoteBase

13. 프로세스의 반환시간은 프로세스의 도착부터 종료까지 걸린 시간이다. 다음과 같은 프로세스들을 선점식 SJF로 스케줄링할 때, 프로세스의 평균 반환시간으로 옳은 것은?

프로세스	도착시간	실행시간(ms)
1	0.0	7
2	1.0	3
3	5.0	2
4	7.0	4

① 6.25 ② 6.75
③ 6.5 ④ 6.35

정답 1번

출제 포인트

선점식 SJF에서 새 프로세스가 도착할 때마다 남은 실행시간을 비교하여 종료시간과 평균 반환시간을 구하는 문제이다.

조건 정리

프로세스별 도착시간과 실행시간은 P1=(0, 7), P2=(1, 3), P3=(5, 2), P4=(7, 4)이다.
선점식 SJF는 현재 준비 큐에서 남은 실행시간이 가장 짧은 프로세스를 실행하며, 더 짧은 프로세스가 도착하면 실행 중인 프로세스를 선점한다.
구해야 할 값은 네 프로세스의 반환시간 평균이다.

사용할 공식

반환시간은 프로세스가 도착한 때부터 종료될 때까지의 시간이다.
반환시간=종료시간-도착시간
평균 반환시간=각 프로세스의 반환시간 합프로세스 수
도착 이후 대기시간과 실제 실행시간을 모두 포함해야 하므로 이 공식을 사용한다.

NoteBase

계산 과정

1. 0~1에는 P1이 실행된다. 1에 P2가 도착하면 P1의 남은 시간은 $6ms$, P2의 실행시간은 $3ms$이므로 P2가 P1을 선점하여 1~4에 실행되고 종료된다.
2. 4~5에는 P1이 다시 실행된다. 5에 P3가 도착하면 P1의 남은 시간은 $5ms$, P3의 실행시간은 $2ms$이므로 P3가 5~7에 실행되고 종료된다.
3. 7에 P4가 도착하면 P1의 남은 시간은 $5ms$, P4의 실행시간은 $4ms$이므로 P4가 7~11에 실행되고 종료된다. 이어서 P1이 11~16에 실행되고 종료된다.
4. 종료시간은 P1=16, P2=4, P3=7, P4=11이다. 따라서 반환시간은 각각 $16 - 0 = 16ms$, $4 - 1 = 3ms$, $7 - 5 = 2ms$, $11 - 7 = 4ms$이다.
5. 평균 반환시간은 다음과 같다.

$$\frac{16 + 3 + 2 + 4}{4} = \frac{25}{4} = 6.25ms$$

선지 판단

① O — 계산한 평균 반환시간이 $6.25ms$이므로 옳다.
② X — $6.75ms$는 계산 결과와 다르다. 올바른 평균 반환시간은 $6.25ms$이다.
③ X — $6.5ms$는 계산 결과와 다르다. 올바른 평균 반환시간은 $6.25ms$이다.
④ X — $6.35ms$는 계산 결과와 다르다. 올바른 평균 반환시간은 $6.25ms$이다.

오답 포인트

선점식 SJF를 비선점식 SJF처럼 처리하거나, 새 프로세스가 도착했을 때 원래 실행시간과 비교하면 실행 순서가 달라진다. 반드시 그 시점의 남은 실행시간을 비교해야 한다. 또한 반환시간은 종료시간 자체가 아니라 종료시간에서 도착시간을 뺀 값이다.

다시 나오면 이렇게 판단

각 도착 시점을 경계로 실행 구간을 나누고, 그때 준비된 프로세스들의 남은 실행시간을 비교한다. 모든 종료시간을 구한 뒤 각 종료시간에서 도착시간을 빼고 그 결과를 평균한다.

암기 포인트

선점식 SJF는 최단 잔여 시간 우선이며, 반환시간은 종료시간에서 도착시간을 뺀 값이다.

NoteBase

14. 파일의 디스크 공간 할당 방법 중 다음 설명에 해당되는 것으로 가장 적절한 것은?

- 순차 접근과 직접 접근이 모두 가능하다.
- 외부 단편화 문제가 없다.

① 연속 할당 ② 연결 할당
③ 고정 할당 ④ 색인(index) 할당

정답 4번

출제 포인트

파일의 디스크 공간 할당 방법을 접근 방식과 외부 단편화 여부로 구별하는 문제이다. 순차 접근과 직접 접근을 모두 지원하면서 외부 단편화가 없는 방식은 색인 할당이다.

핵심 개념

연속 할당: 파일을 디스크의 연속된 블록에 저장하는 방식이다. 순차 접근과 직접 접근이 쉽지만, 연속 공간을 확보해야 하므로 외부 단편화가 생길 수 있다.
연결 할당: 파일 블록을 포인터로 연결하여 비연속적으로 저장하는 방식이다. 외부 단편화는 없지만 원하는 블록까지 연결을 따라가야 하므로 직접 접근에 불리하다.
색인 할당: 파일의 모든 데이터 블록 주소를 별도의 색인 블록에 모아 두는 방식이다. 색인 항목으로 원하는 블록을 찾을 수 있어 순차 접근과 직접 접근이 모두 가능하고, 데이터 블록을 연속 배치할 필요가 없어 외부 단편화가 없다.

선지별 해설

① X — 연속 할당은 순차 접근과 직접 접근이 모두 가능하지만, 파일마다 연속된 빈 공간이 필요하므로 외부 단편화가 발생할 수 있다. 틀린 표현은 '외부 단편화 문제가 없다'이며, 올바르게는 '외부 단편화가 발생할 수 있다'이다.
② X — 연결 할당은 블록을 비연속적으로 배치하므로 외부 단편화는 없고 순차 접근에는 적합하지만, 특정 블록에 가려면 앞선 블록의 포인터를 차례로 따라가야 하므로 직접 접근에 불리하다. 틀린 표현은 '직접 접근이 가능하다'이며, 올바르게는 '직접 접근이 어렵고 순차 접근에 적합하다'이다.
③ X — 고정 할당은 제시된 두 조건을 만족하는 파일의 대표적인 디스크 공간 할당 방법이 아니다. 틀린 표현은 '고정 할당'이며, 올바르게는 '색인 할당'이다.
④ O — 색인 할당은 색인 블록에 각 데이터 블록의 주소를 저장하므로 순서대로 읽는 순차 접근과 특정 블록을 찾는 직접 접근이 모두 가능하다. 데이터 블록을 비연속적으로 배치할 수 있어 외부 단편화도 없다.

NoteBase

오답 포인트

①은 두 접근 방식이 모두 가능하다는 점만 보고 고르기 쉽지만 '외부 단편화가 없다'는 조건에서 탈락한다. ②는 외부 단편화가 없다는 점은 맞지만 포인터를 차례로 따라가야 하므로 직접 접근 조건을 충족하지 못한다.

다시 나오면 이렇게 판단

'직접 접근'이 보이면 블록 위치를 즉시 계산하거나 주소표에서 찾을 수 있는지를 확인한다. 여기에 '외부 단편화 없음'까지 제시되면, 비연속 배치가 가능하면서 색인표로 원하는 블록을 찾는 색인 할당으로 판단한다.

암기 포인트

연속 할당은 빠른 직접 접근·외부 단편화, 연결 할당은 순차 접근·외부 단편화 없음, 색인 할당은 직접 접근·외부 단편화 없음이다.

NoteBase

15. 페이징 기법을 사용하는 64비트 컴퓨터에서 페이지 크기가 4,096바이트일 때, 논리주소를 구성하는 페이지 번호의 비트 수와 페이지 변위(offset)의 비트 수로 올바른 것은?

① 40, 24 ② 12, 52

③ 42, 12 ④ 52, 12

정답 4번

출제 포인트

페이지 크기로 페이지 변위(offset)의 비트 수를 구하고, 전체 논리주소 비트 수에서 이를 빼 페이지 번호의 비트 수를 구하는 문제이다.

조건 정리

주어진 값: 논리주소는 64비트이고 페이지 크기는 4,096바이트이다.
구해야 할 값: 페이지 번호의 비트 수와 페이지 변위의 비트 수이다.
주의할 조건: 바이트 단위 주소에서 한 페이지의 크기가 2^n바이트이면 페이지 내부의 각 바이트를 구분하는 변위는 n비트이다.

사용할 공식

페이지 변위 비트 수=_2(페이지 크기)
페이지 번호 비트 수=전체 주소 비트 수-페이지 변위 비트 수
페이지 변위는 한 페이지 안의 바이트 위치를 나타내므로 페이지 크기만큼의 서로 다른 위치를 표현해야 한다.

계산 과정

1. $4,096 = 2^{12}$이므로 페이지 변위는 12비트이다.
2. 논리주소 64비트 중 변위 12비트를 제외하면 페이지 번호는 $64 - 12 = 52$비트이다.
3. 따라서 '페이지 번호 52비트, 페이지 변위 12비트'가 된다.

NoteBase

선지 판단

① X — ‘40, 24’는 페이지 변위를 24비트로 잡은 것이 잘못이다. 페이지 크기 4,096바이트는 2^{12}바이트이므로 올바르게는 ‘52, 12’이다.
② X — ‘12, 52’는 페이지 번호와 페이지 변위의 비트 수를 서로 뒤바꾼 값이다. 올바르게는 ‘52, 12’이다.
③ X — ‘42, 12’에서 페이지 변위 12비트는 맞지만 페이지 번호 42비트가 잘못이다. 전체 64비트에서 12비트를 빼야 하므로 올바르게는 ‘52, 12’이다.
④ O — 페이지 번호 52비트와 페이지 변위 12비트는 각각 $64 - 12$와 $\log_2 4{,}096$의 계산 결과와 일치한다.

오답 포인트

페이지 크기 4,096의 숫자 자체를 비트 수로 보거나, 페이지 번호와 페이지 변위의 순서를 뒤바꾸기 쉽다. 특히 ②는 두 값을 정확히 역순으로 제시한 함정이다.

다시 나오면 이렇게 판단

먼저 페이지 크기를 2^n으로 바꾸어 변위 n비트를 구한 뒤, 전체 주소 비트 수에서 n을 빼 페이지 번호 비트 수를 구한다. 마지막으로 두 비트 수의 합이 전체 주소 비트 수와 같은지 검산한다.

암기 포인트

페이지 크기 $= 2^n$바이트이면 변위는 n비트이고, 페이지 번호는 ‘전체 주소 비트 수 $-n$’비트이다.

NoteBase

16. 데이터 링크 계층의 프로토콜인 HDLC(High-level Data Link Control)에 대한 설명으로 가장 적절하지 않은 것은?
① 바이트 단위로 전송하는 동기 방식이다.
② 전이중(full duplex) 방식을 사용한다.
③ 비트 스터핑(bit stuffing) 기능으로 투명성을 제공한다.
④ HDLC 프레임의 플래그는 프레임 시작과 끝을 나타내며 동기화에 사용한다.

정답 1번

출제 포인트

HDLC의 전송 단위와 통신 방식, 비트 스터핑, 플래그의 역할을 구분하는 문제이다.

핵심 개념

HDLC는 데이터 링크 계층에서 사용하는 비트 지향 동기식 프로토콜이다. 프레임의 시작과 끝은 플래그로 표시하며, 데이터 안의 비트열이 플래그로 잘못 인식되지 않도록 비트 스터핑을 사용한다.

선지별 해설

① X — 틀린 표현은 '바이트 단위로 전송'이다. HDLC는 바이트 지향이 아니라 비트 단위로 프레임을 처리하는 비트 지향 동기식 프로토콜이다.
② O — HDLC는 양방향으로 동시에 데이터를 전송할 수 있는 전이중 방식을 지원한다.
③ O — 비트 스터핑은 데이터 부분에 플래그와 같은 비트 패턴이 우연히 나타나는 것을 방지하여 데이터와 프레임 경계를 구별하게 하므로 투명성을 제공한다.
④ O — HDLC 프레임의 플래그는 프레임의 시작과 끝을 표시하며, 수신 측의 프레임 동기화에 사용된다.

오답 포인트

①의 '동기 방식'은 맞지만 '바이트 단위'가 틀린 표현이다. HDLC를 문자·바이트 지향 프로토콜과 혼동하면 정답을 놓치기 쉽다.

다시 나오면 이렇게 판단

HDLC의 특징은 '비트 지향·동기식·전이중·플래그로 경계 표시·비트 스터핑으로 투명성 확보'의 묶음으로 판단한다. 이와 어긋나는 '바이트 지향'이라는 표현은 틀린 것으로 본다.

NoteBase

암기 포인트

HDLC는 비트 지향, 플래그는 경계와 동기화, 비트 스터핑은 투명성이다.

NoteBase

17. 다음 중 정보 전송의 다중화(multiplexing)에 대한 설명으로 가장 적절하지 않은 것은?

① 주파수 분할 다중화(FDM)는 정보를 같은 시간에 전송하기 위해 별도의 주파수 채널을 설정해야 하고, 채널 간의 상호 간섭을 막기 위해 보호 대역이 필요하다.

② 동기식 시분할 다중화(STDM)는 전송로 대역폭 하나를 시간 슬롯으로 나눈 채널에 할당하여 채널 여러 개가 전송로의 시간을 분할하여 사용한다.

③ 비동기식 시분할 다중화(ATDM)는 STDM과 유사한 방법이지만, 전송 요구가 없을 때는 시간 슬롯 낭비가 발생한다.

④ 코드분할 다중화(CDMA)는 대역확산 기법을 사용한다.

정답 3번

출제 포인트

이 문제는 FDM, STDM, ATDM, CDMA가 전송로를 나누어 쓰는 기준과 유휴 자원을 처리하는 방식의 차이를 묻는다.

핵심 개념

FDM은 전송 대역폭을 여러 주파수 대역으로 나누어 동시에 사용하는 방식이며, 인접 채널의 간섭을 줄이기 위한 보호 대역이 필요하다.
STDM은 전송 시간을 고정된 시간 슬롯으로 나누고 각 채널에 주기적으로 슬롯을 배정하는 방식이므로, 해당 채널에 보낼 데이터가 없어도 배정된 슬롯이 비어 낭비될 수 있다.
ATDM은 전송 요구가 있는 채널에 시간 슬롯을 동적으로 배정하는 방식이므로 STDM의 빈 슬롯 낭비를 줄인다.
CDMA는 사용자마다 서로 다른 코드를 부여하고 대역확산을 이용하여 같은 주파수 대역을 함께 사용하는 방식이다.

NoteBase

선지별 해설

① O — FDM은 여러 신호에 서로 다른 주파수 채널을 배정하여 같은 시간에 전송한다. 인접 주파수 채널 사이의 상호 간섭을 줄이기 위해 보호 대역을 둔다는 설명도 옳다.
② O — STDM은 하나의 전송로 사용 시간을 고정된 시간 슬롯으로 나누어 여러 채널에 차례로 배정하는 방식이다. 즉 여러 채널이 전송로의 시간을 분할하여 사용한다.
③ X — 틀린 표현은 '전송 요구가 없을 때는 시간 슬롯 낭비가 발생한다'이다. ATDM은 전송할 데이터가 있는 채널에 슬롯을 동적으로 배정하므로, 전송 요구가 없는 채널에 고정 슬롯을 배정해서 생기는 낭비를 줄이는 방식이다. 시간 슬롯 낭비가 발생할 수 있는 것은 고정적으로 슬롯을 배정하는 STDM의 특징이다.
④ O — CDMA는 각 사용자의 신호를 고유 코드로 구분하고 넓은 주파수 대역으로 확산하는 대역확산 기법을 사용한다.

오답 포인트

③의 함정은 STDM과 ATDM이 모두 시간 슬롯을 사용한다는 공통점만 보고, 고정 배정과 동적 배정의 차이를 놓치게 하는 데 있다. 빈 슬롯 낭비는 STDM에서 나타나며, ATDM은 이를 줄이기 위해 전송 요구가 있는 채널에 슬롯을 배정한다.

다시 나오면 이렇게 판단

다중화 방식은 무엇을 나누거나 구분하는지부터 확인한다. 주파수 대역을 나누면 FDM, 시간을 고정 슬롯으로 나누면 STDM, 전송 요구에 따라 시간 슬롯을 동적으로 배정하면 ATDM, 코드로 사용자를 구분하면 CDMA이다.

암기 포인트

FDM은 주파수와 보호 대역, STDM은 고정 시간 슬롯과 빈 슬롯, ATDM은 요구 기반 동적 슬롯, CDMA는 코드와 대역확산이다.

NoteBase

18. 다음 표의 수식을 후위 표기법으로 변환했을 때 가장 적절한 것은?

100 - 20 * 5 + 2

① 100 20 5 2 * - +
② 100 20 5 * 2 - +
③ 100 20 5 * - 2 +
④ 100 20 5 * - + 2

정답 3번

출제 포인트

중위 표기식에서 연산자의 우선순위와 결합 순서를 반영하여 후위 표기식으로 변환하는 절차를 묻는 문제이다.

올바른 순서

1. 피연산자 100, 20, 5는 원래 순서대로 출력한다.
2. 곱셈은 뺄셈보다 우선하므로 20과 5 뒤에 *를 출력하여 20 5 *로 만든다.
3. 곱셈 결과를 100에서 빼는 연산을 반영하여 그 뒤에 –를 출력한다. 이때까지의 결과는 100 20 5 * –이다.
4. 마지막 피연산자 2를 출력한 뒤 덧셈 연산자 +를 출력한다. 최종 후위 표기식은 100 20 5 * – 2 +이다.

왜 이 순서인가: 후위 표기법에서는 연산자가 해당 연산의 두 피연산자 뒤에 놓여야 하므로, 먼저 계산되는 20 * 5의 *, 그 결과를 사용하는 –, 마지막 덧셈의 + 순서로 배치된다.

선지별 해설

① X — 100 20 5 2 * – +는 피연산자 2를 곱셈 연산자보다 먼저 출력하여 원래 식의 20 * 5를 보존하지 못한다. 틀린 배열은 '5 2 *' 부분이며, 올바르게는 100 20 5 * – 2 +이다.

② X — 100 20 5 * 2 – +는 2를 뺄셈보다 먼저 놓아 원래 식의 100 – (20 * 5)가 아니라 곱셈 결과와 2를 먼저 빼는 배열이 된다. 올바르게는 뺄셈 연산자를 2보다 먼저 출력한 100 20 5 * – 2 +이다.

③ O — 100 20 5 * – 2 +는 먼저 20 * 5, 다음으로 100에서 그 결과를 빼는 연산, 마지막으로 2를 더하는 연산을 정확히 나타낸다.

④ X — 100 20 5 * – + 2는 덧셈의 오른쪽 피연산자 2보다 +를 먼저 놓았으므로 올바른 후위 표기식이 아니다. 틀린 배열은 '+ 2'이며, 올바르게는 '2 +'로 고친 100 20 5 * – 2 +이다.

NoteBase

오답 포인트

피연산자를 보이는 대로 모두 먼저 적는 것이 아니라, 연산자 우선순위에 따라 하나의 연산이 완성될 때 그 연산자를 출력해야 한다. 특히 마지막 피연산자 2를 너무 일찍 출력하거나 +를 2보다 먼저 출력하는 배열이 함정이다.

다시 나오면 이렇게 판단

피연산자의 상대적 순서는 유지하고, 각 연산자를 그 연산에 필요한 두 피연산자 뒤로 옮긴다. 괄호가 없다면 우선순위가 높은 연산을 먼저 묶고, 우선순위가 같으면 왼쪽에서 오른쪽으로 처리한다.

암기 포인트

후위 표기법은 '피연산자 두 개 뒤에 연산자'이며, 이 식의 연산 순서는 * → − → +이다.

NoteBase

19. 다음 중 UML(Unified Modeling Language) 다이어그램에 대한 설명 중 가장 적절하지 않은 것은?

① Class 다이어그램은 클래스 내부의 내용과 클래스 사이의 연관관계를 이용하여 시스템의 구조를 정의한다.

② State 다이어그램은 객체의 동적 행위를 모형화하기 위해 객체 간의 메시지 처리를 시간적 흐름으로 표현한다.

③ Use Case 다이어그램은 사용자(actor) 관점에서 시스템의 기능과 상호작용을 표현한다.

④ Component 다이어그램은 시스템을 구성하는 컴포넌트와 상호작용을 표현한다.

정답 2번

출제 포인트

UML의 대표 다이어그램이 각각 시스템의 정적 구조, 상태 변화, 사용자 관점의 기능, 컴포넌트 구조 중 무엇을 표현하는지 구분하는 문제이다.

핵심 개념

UML은 소프트웨어 시스템의 구조와 동작을 표준화된 다이어그램으로 모델링하는 언어이다.
Class 다이어그램은 클래스의 속성·연산과 클래스 사이의 관계를 나타내는 정적 구조 다이어그램이다.
State 다이어그램은 하나의 객체나 시스템이 사건에 따라 어떤 상태로 전이하는지를 나타내는 동적 행위 다이어그램이다.
Sequence 다이어그램은 여러 객체가 주고받는 메시지를 시간의 흐름에 따라 나타내는 상호작용 다이어그램이다.
Use Case 다이어그램은 사용자와 시스템 기능의 관계를 나타내며, Component 다이어그램은 시스템을 구성하는 컴포넌트와 그 의존·상호작용 관계를 나타낸다.

NoteBase

선지별 해설

① O — Class 다이어그램은 클래스 내부의 속성·연산과 클래스 사이의 연관, 상속 등의 관계를 이용하여 시스템의 정적 구조를 정의한다.
② X — 틀린 표현은 '객체 간의 메시지 처리를 시간적 흐름으로 표현한다'이다. 이는 Sequence 다이어그램의 설명이다. State 다이어그램은 객체나 시스템이 사건에 반응하여 상태를 바꾸는 과정과 상태 전이를 표현한다.
③ O — Use Case 다이어그램은 사용자(actor)의 관점에서 시스템이 제공하는 기능과 사용자·기능 사이의 상호작용을 표현한다.
④ O — Component 다이어그램은 시스템을 이루는 소프트웨어 컴포넌트와 컴포넌트 사이의 의존 및 상호작용 관계를 표현한다.

오답 포인트

②는 State 다이어그램과 Sequence 다이어그램이 모두 동적 측면을 표현한다는 점을 이용한 함정이다. '상태와 전이'는 State 다이어그램이고, '객체 간 메시지와 시간 순서'는 Sequence 다이어그램이다.

다시 나오면 이렇게 판단

선지의 핵심 명사를 먼저 찾는다. 클래스·관계·정적 구조이면 Class, 상태·사건·전이면 State, 객체 간 메시지·시간 순서이면 Sequence, 사용자·기능이면 Use Case, 컴포넌트·의존 관계이면 Component 다이어그램으로 판단한다.

암기 포인트

Class는 정적 구조, State는 상태 전이, Sequence는 시간순 메시지, Use Case는 사용자 기능, Component는 구성 요소와 의존 관계이다.

NoteBase

20. 다음 중 이진 트리(binary tree)에 대한 설명으로 가장 적절하지 않은 것은? (단, 트리의 최상위 레벨이 깊이 1이다.)

① n개의 노드를 가진 이진 트리는 n-1개의 간선을 갖는다.

② 레벨 a에서의 최대 노드 수는 2^{a-1}개이다.

③ n개의 노드를 가진 이진 트리의 깊이는 최대 n-1이다.

④ 깊이가 d인 이진 트리의 노드 수는 최대 $2^{d}-1$개이다.

정답 3번

출제 포인트

이 문제는 루트의 깊이를 1로 정할 때 이진 트리의 간선 수, 레벨별 최대 노드 수, 최대 깊이, 전체 최대 노드 수의 관계를 묻는 문제이다.

핵심 개념

이진 트리는 각 노드가 최대 2개의 자식 노드를 갖는 트리이다. 노드가 n개인 트리의 간선 수는 항상 $n-1$개이다. 레벨 a의 최대 노드 수는 2^{a-1}개이고, 깊이가 d인 이진 트리의 전체 최대 노드 수는 $2^{d}-1$개이다. 한쪽 자식으로만 이어지는 편향 이진 트리에서는 노드가 n개일 때 최대 깊이가 n이 된다.

선지별 해설

① O — 트리는 루트를 제외한 모든 노드가 부모와 연결되는 간선 하나씩을 가지므로, 노드가 n개이면 간선은 $n-1$개이다.

② O — 루트가 레벨 1이고 각 노드가 다음 레벨에서 최대 2개의 자식을 가질 수 있으므로 레벨 a의 최대 노드 수는 2^{a-1}개이다.

③ X — 틀린 표현은 '깊이는 최대 $n-1$'이다. 루트의 깊이가 1이므로 노드가 한쪽으로만 이어지는 경우 각 노드의 깊이는 1부터 n까지가 되어, 올바르게는 '깊이는 최대 n'이다.

④ O — 깊이 1부터 d까지 각 레벨을 최대 노드 수로 채우면 전체 노드 수는 $1+2+\cdots+2^{d-1}=2^{d}-1$개이다.

오답 포인트

③은 간선 수의 최댓값 또는 루트에서 가장 깊은 노드까지의 간선 수 $n-1$과, 문제에서 레벨 1부터 세는 트리의 깊이 n을 혼동하게 만든다.

NoteBase

다시 나오면 이렇게 판단

먼저 루트의 깊이가 0인지 1인지 확인한다. 루트의 깊이가 1이면 레벨 a의 최대 노드 수는 2^{a-1}, 깊이 d의 전체 최대 노드 수는 $2^d - 1$, 노드 n개의 최대 깊이는 n으로 판단한다.

암기 포인트

루트 깊이 1 기준으로 '레벨 최대 2^{a-1}, 전체 최대 $2^d - 1$, 편향 트리 최대 깊이 n'이다.

NoteBase

21. 빅오(big O) 표기법은 알고리즘의 시간 복잡도를 입력의 크기에 대한 함수로 나타내는 방법이다. 다음 중 빅오(big O) 표기법의 실행시간 크기를 비교한 것으로 가장 적절한 것은?

① $O(1) < O(\log n) < O(n) < O(n^2) < O(2^n)$

② $O(\log n) < O(1) < O(n) < O(n^2) < O(2^n)$

③ $O(1) < O(\log n) < O(n) < O(2^n) < O(n^2)$

④ $O(\log n) < O(1) < O(n) < O(2^n) < O(n^2)$

정답 1번

출제 포인트

빅오 표기법으로 나타낸 알고리즘의 실행시간 증가율을 입력 크기 n이 충분히 커질 때의 크기 순서로 비교하는 문제이다.

핵심 개념

빅오 표기법은 입력 크기가 커질 때 알고리즘의 실행시간 또는 연산량이 증가하는 정도의 상한을 나타낸다.
$O(1)$: 입력 크기와 무관한 상수 시간이다.
$O(\log n)$: 입력이 커져도 증가가 매우 완만한 로그 시간이다.
$O(n)$: 입력 크기에 비례하는 선형 시간이다.
$O(n^2)$: 입력 크기의 제곱에 비례하는 이차 시간이다.
$O(2^n)$: 입력이 하나 늘 때마다 급격히 커지는 지수 시간이다.
증가율은 $O(1) < O(\log n) < O(n) < O(n^2) < O(2^n)$의 순서이다.

선지별 해설

① O — $O(1) < O(\log n) < O(n) < O(n^2) < O(2^n)$은 입력 크기가 충분히 커질 때 증가율이 작은 것부터 큰 것으로 올바르게 배열한 순서이다.
② X — 틀린 부분은 $O(\log n) < O(1)$이다. 상수 시간은 입력 크기에 따라 증가하지 않으므로 로그 시간보다 증가율이 작다. 올바르게는 $O(1) < O(\log n) < O(n) < O(n^2) < O(2^n)$이다.
③ X — 틀린 부분은 $O(2^n) < O(n^2)$이다. 지수 함수는 입력 크기가 충분히 커지면 이차함수보다 빠르게 증가한다. 올바르게는 $O(n^2) < O(2^n)$, 전체 순서는 $O(1) < O(\log n) < O(n) < O(n^2) < O(2^n)$이다.
④ X — $O(\log n) < O(1)$과 $O(2^n) < O(n^2)$이 모두 틀렸다. 올바르게는 $O(1) < O(\log n)$이고 $O(n^2) < O(2^n)$이므로, 전체 순서는 $O(1) < O(\log n) < O(n) < O(n^2) < O(2^n)$이다.

NoteBase

오답 포인트

②와 ④는 상수 시간과 로그 시간의 위치를 바꾸고, ③과 ④는 이차 시간과 지수 시간의 위치를 바꾸어 혼동을 유도한다. 특히 $O(1)$은 입력이 커져도 실행시간의 증가가 없고, $O(2^n)$은 다항식인 $O(n^2)$보다 결국 훨씬 빠르게 증가한다.

다시 나오면 이렇게 판단

복잡도는 상수형, 로그형, 선형, 다항형, 지수형의 순서로 증가한다고 판단한다. 같은 계열 안에서는 지수가 큰 다항식이 더 빠르게 증가하지만, 모든 고정 차수의 다항식은 지수함수보다 증가율이 작다.

암기 포인트

상수 < 로그 < 선형 < 제곱 < 지수, 즉 $1 < \log n < n < n^2 < 2^n$이다.

NoteBase

22. 다음 중 SQL에 대한 설명으로 가장 적절하지 않은 것은?

① SQL은 비절차적 언어이며 다른 언어에 삽입되어 내장 SQL로도 사용 가능하다.
② SQL 뷰(view)는 하나 이상의 테이블로부터 만들어진 가상 테이블이다.
③ SQL ALTER 문을 사용하여 테이블을 변경할 수 있다.
④ SQL 뷰가 정의된 기본 테이블이 변경되어도 뷰는 변경되지 않는다.

정답 4번

출제 포인트

SQL의 언어적 특징, 뷰의 성격, ALTER 문의 기능을 구분하고 뷰와 기본 테이블의 의존 관계를 판단하는 문제이다.

핵심 개념

SQL은 사용자가 원하는 결과를 선언하면 데이터베이스 관리 시스템이 처리 방법을 결정하는 비절차적 언어이며, 다른 프로그래밍 언어 안에 삽입된 내장 SQL로도 사용할 수 있다.
뷰(view)는 하나 이상의 테이블이나 다른 뷰를 조회하는 질의로 정의된 가상 테이블이다. 일반적으로 실제 데이터를 별도로 저장하지 않고 기본 테이블의 데이터와 구조에 의존하므로, 기본 테이블의 데이터가 바뀌면 뷰의 조회 결과도 달라질 수 있고 구조 변경은 뷰의 사용에 영향을 줄 수 있다.
ALTER 문은 테이블 등의 데이터베이스 객체 구조를 변경하는 데이터 정의어이다.

선지별 해설

① O — SQL은 처리 절차보다 원하는 결과를 기술하는 비절차적 언어이며, 호스트 언어의 프로그램에 삽입하여 내장 SQL로 사용할 수 있다.
② O — 뷰는 하나 이상의 테이블을 바탕으로 질의를 통해 정의되는 가상 테이블이다.
③ O — ALTER TABLE 문을 사용하면 열의 추가·변경·삭제나 제약 조건 변경 등 테이블 구조를 변경할 수 있다.
④ X — 틀린 표현은 '기본 테이블이 변경되어도 뷰는 변경되지 않는다'이다. 올바르게는 뷰가 기본 테이블에 의존하므로 기본 테이블의 데이터 변경은 뷰의 조회 결과에 반영되고, 기본 테이블의 구조 변경도 뷰의 유효성이나 사용에 영향을 줄 수 있다고 해야 한다.

NoteBase

오답 포인트

뷰가 실제 데이터를 독립적으로 저장하지 않는 가상 테이블이라는 점을 '기본 테이블과 무관하다'는 뜻으로 오해하기 쉽다. 뷰의 정의문 자체와 뷰를 통해 보이는 조회 결과를 구분해야 한다.

다시 나오면 이렇게 판단

SQL 관련 선지는 언어의 성격, 명령문의 기능, 객체 간 의존 관계로 나누어 판단한다. 특히 뷰는 독립된 데이터 사본이 아니라 기본 테이블을 참조하는 질의라는 점을 기준으로 삼는다.

암기 포인트

SQL은 비절차적 언어, ALTER는 구조 변경, 뷰는 기본 테이블에 의존하는 가상 테이블이다.

NoteBase

23. 다음 중 2진수 10101001.11000011를 16진수로 표현한 것으로 옳은 것은?

① A9.B3 ② A9.C3

③ A7.B3 ④ A7.C3

정답 2번

출제 포인트

이 문제는 2진수의 각 4비트를 16진수 한 자리로 대응시켜 정수부와 소수부를 변환하는 문제이다.

조건 정리

주어진 값: 2진수 10101001.11000011

구해야 할 값: 이에 대응하는 16진수

주의할 조건: 소수점을 기준으로 정수부는 오른쪽부터, 소수부는 왼쪽부터 4비트씩 묶는다.

사용할 공식

2진수 4비트는 16진수 한 자리와 대응한다.

$1010_2 = A_{16}, \quad 1001_2 = 9_{16}, \quad 1100_2 = C_{16}, \quad 0011_2 = 3_{16}$

16진수 한 자리가 나타낼 수 있는 0부터 15까지의 값을 2진수 4비트가 그대로 나타내므로 이 대응 관계를 사용한다.

계산 과정

1. 소수점을 기준으로 4비트씩 묶으면 다음과 같다.

1010 1001.1100 0011

2. 정수부를 변환하면 $1010_2 = A_{16}$, $1001_2 = 9_{16}$이므로 A9이다.

3. 소수부를 변환하면 $1100_2 = C_{16}$, $0011_2 = 3_{16}$이므로 C3이다.

4. 따라서 전체 변환 결과는 A9.C3이다.

선지 판단

① X — 정수부 A9는 맞지만 소수부 1100은 B가 아니라 C이므로 A9.B3는 틀리다. 올바르게는 A9.C3이다.

② O — 정수부 1010 1001은 A9이고 소수부 1100 0011은 C3이므로 A9.C3가 맞다.

③ X — A7은 2진수 10100111에 대응하여 주어진 정수부 10101001과 다르고, 소수부의 B도 1100에 대응하지 않는다. 올바르게는 A9.C3이다.

④ X — 소수부 C3는 맞지만 정수부 1001은 7이 아니라 9이므로 A7.C3는 틀리다. 올바르게는 A9.C3이다.

NoteBase

오답 포인트

핵심 함정은 소수점을 없애고 전체를 연속해서 묶거나 4비트와 16진수의 대응을 혼동하는 것이다. 특히 1100은 B가 아니라 C이고, 1001은 7이 아니라 9이다.

다시 나오면 이렇게 판단

소수점을 경계로 나눈 뒤 정수부는 오른쪽부터, 소수부는 왼쪽부터 4비트씩 묶고 각 묶음을 16진수 한 자리로 바꾼다. 부족한 비트가 있으면 정수부의 왼쪽 또는 소수부의 오른쪽에 0을 채운다.

암기 포인트

2진수 4비트는 16진수 1자리이며, 1010=A, 1011=B, 1100=C이다.

NoteBase

24. 다음 중 소프트웨어 모듈 설계의 평가에 대한 설명으로 가장 적절하지 않은 것은?

① 모듈의 결합도(coupling)는 모듈들이 서로 관련되거나 연결된 정도를 의미하며 낮을수록 좋다.

② 모듈의 응집도(cohesion)는 모듈 간에 기능적인 연관 정도를 의미하며 높을수록 좋다.

③ 어떤 모듈이 다른 모듈을 호출하며 데이터를 넘겨주는 경우 이 모듈 간에 제어 결합도를 가진다고 한다.

④ 모듈 내 모든 요소가 한 가지 기능을 수행하기 위해 구성되는 경우 이들 요소는 기능적 응집도로 묶여있다고 한다.

정답 2번

출제 포인트

모듈 사이의 의존 정도인 결합도와 모듈 내부 요소의 관련 정도인 응집도를 구별하는 문제이다. 좋은 모듈 설계는 결합도가 낮고 응집도가 높다.

핵심 개념

결합도(coupling): 서로 다른 모듈이 연결되거나 의존하는 정도이다. 낮을수록 한 모듈의 변경이 다른 모듈에 미치는 영향이 작아진다.
응집도(cohesion): 한 모듈 내부의 구성 요소들이 하나의 목적을 위해 서로 관련된 정도이다. 높을수록 모듈의 기능이 명확해진다.
제어 결합도: 한 모듈이 다른 모듈의 처리 흐름을 지시하는 제어 정보나 플래그를 전달하는 결합 형태이다.
기능적 응집도: 모듈 내부의 모든 요소가 하나의 명확한 기능을 수행하기 위해 함께 작동하는 가장 강한 응집 형태이다.

NoteBase

선지별 해설

① O — 결합도는 모듈 사이의 관련·의존 정도를 뜻하며, 낮을수록 모듈의 독립성과 변경 용이성이 높아지므로 좋은 설계이다.
② X — 틀린 표현은 '모듈 간에 기능적인 연관 정도'이다. 응집도는 모듈 간이 아니라 하나의 모듈 내부 구성 요소 사이의 기능적 관련 정도이며, 높을수록 좋다. 올바르게는 '모듈 내부 요소들의 기능적 연관 정도'이다.
③ O — 호출하는 모듈이 호출되는 모듈의 처리 방향을 정하는 제어용 데이터를 넘기면 두 모듈은 제어 결합 관계가 된다. 다만 처리할 일반 데이터만 매개변수로 넘기는 경우는 데이터 결합에 해당한다.
④ O — 모듈 내부의 모든 요소가 하나의 기능을 수행하기 위해 구성되어 있으면 기능적 응집도에 해당한다.

오답 포인트

②는 '높을수록 좋다'는 뒤 문장이 맞아 보여 앞의 '모듈 간'이라는 잘못된 범위를 놓치기 쉽다. 응집도는 모듈 내부, 결합도는 모듈 사이를 평가한다. 또한 ③에서 일반 데이터 전달은 데이터 결합이고, 처리 흐름을 지시하는 제어용 데이터 전달이어야 제어 결합이라는 점을 구분한다.

다시 나오면 이렇게 판단

평가 범위를 먼저 본다. 모듈 사이는 결합도, 모듈 내부는 응집도이다. 그다음 좋은 설계의 방향이 '낮은 결합도·높은 응집도'인지 확인한다.

암기 포인트

밖으로는 느슨하게 결합하고, 안으로는 단단하게 응집한다.

NoteBase

25. 다음은 선택 정렬(selection sort)을 구현한 C 프로그램이다. 배열 arr의 값을 오름차순으로 출력하려고 할 때, /* 공란 */ 으로 표시된 곳에 채워져야 할 코드로 가장 적절한 것은?

```
#include <stdio.h>
int main() {
    int arr[5] = {10, 5, 8, 3, 7};
    int temp, min;

    for (int i = 0; i < 5; i++) {
        min = i;
        /* 공란 */
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }
}
```

① for (int j = 1; j < 5; j++)
    if (arr[j] > arr[min]) min = j;

② for (int j = 1; j < 5; j++)
    if (arr[j] < arr[min]) min = j;

③ for (int j = i; j < 5; j++)
    if (arr[j] > arr[min]) min = j;

④ for (int j = i; j < 5; j++)
    if (arr[j] < arr[min]) min = j;

정답 4번

출제 포인트

선택 정렬로 배열을 오름차순 정렬할 때, 아직 정렬되지 않은 구간에서 최솟값의 위치를 찾는 반복문의 시작 위치와 비교 연산자를 판단하는 문제이다.

NoteBase

핵심 개념

선택 정렬은 각 단계에서 미정렬 구간의 최솟값을 찾아 그 구간의 첫 원소와 교환하는 정렬 방식이다. 바깥 반복문의 인덱스가 i이면 탐색 범위는 i부터 배열의 끝까지이고, 오름차순 정렬에서는 arr[j] < arr[min]일 때 min을 j로 갱신한다. j를 i부터 시작하면 arr[i]를 자기 자신과 한 번 비교하지만 결과에는 영향을 주지 않는다.

선지별 해설

① X

비교식 arr[j] > arr[min]은 최댓값의 위치를 찾으므로 오름차순이 아니라 내림차순 방향의 선택 정렬이 된다. 또한 j를 항상 1부터 시작하면 i보다 앞선 정렬 완료 구간까지 다시 탐색한다. 올바르게는 j를 i 또는 i + 1부터 시작하고 arr[j] < arr[min]일 때 min = j로 갱신한다.

② X

비교 연산자 <는 최솟값을 찾는 데 맞지만, 시작값 j = 1이 잘못되었다. i가 2 이상일 때도 이미 정렬된 앞부분을 다시 탐색하여 확정된 원소가 뒤로 이동할 수 있다. 올바르게는 for (int j = i; j < 5; j++)로 미정렬 구간만 검사한다.

③ X

탐색 범위 j = i는 맞지만, 비교식 arr[j] > arr[min]이 최댓값을 선택하므로 내림차순으로 정렬된다. 올바르게는 arr[j] < arr[min]일 때 min = j로 갱신한다.

④ O

j를 i부터 시작하여 현재 미정렬 구간만 검사하고, arr[j] < arr[min]일 때 더 작은 원소의 위치를 min에 저장한다. 반복문이 끝난 뒤 arr[i]와 arr[min]을 교환하므로 배열이 오름차순으로 정렬된다.

오답 포인트

j의 시작값과 비교 연산자를 따로 확인해야 한다. ②는 < 때문에 맞아 보이지만 j = 1이 정렬 완료 구간을 다시 포함하고, ③은 j = i 때문에 맞아 보이지만 >가 최댓값을 찾아 정렬 방향을 거꾸로 만든다.

다시 나오면 이렇게 판단

오름차순 선택 정렬에서는 매 단계마다 '미정렬 구간 i부터 끝까지'와 '더 작으면 최솟값 위치 갱신'의 두 조건을 함께 확인한다. 즉 시작 위치는 i 또는 i + 1이고 비교식은 arr[j] < arr[min]이다.

암기 포인트

오름차순 선택 정렬은 미정렬 구간에서 최솟값을 선택한다: j는 i부터, 비교는 <이다.

NoteBase