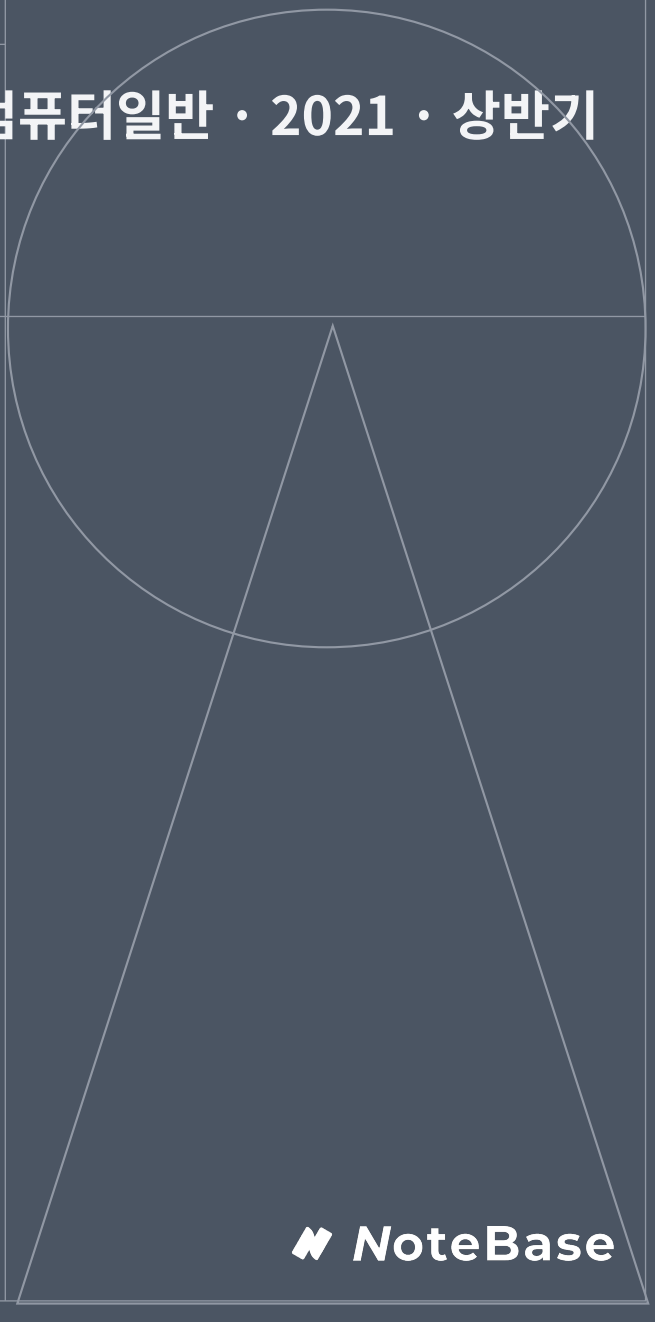
기출OX 해설집

공무원/해경/일반직 9급 · 컴퓨터일반 · 2021 · 상반기



NoteBase

기출OX 해설집

공무원/해경/일반직 9급 · 컴퓨터일반 · 2021 · 상반기

공무원/해경/일반직 9급 · 컴퓨터일반 · 2021 · 상반기

문 1. 다음 중 해싱(Hashing)에 대한 설명으로 가장 옳지 않은 것은?

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

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

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

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

정답 3번

출제 포인트

해싱의 충돌 개념, 해시 함수의 탐색 효율, 충돌 해결 방법인 선형 조사법과 체이닝의 차이를 구별하는 문제이다.

핵심 개념

해싱은 탐색키를 해시 함수에 넣어 얻은 해시 주소로 데이터의 저장 위치에 접근하는 방법이다.
충돌은 서로 다른 탐색키가 같은 해시 주소를 얻는 현상이다.
선형 조사법은 충돌이 발생하면 해시 테이블 안의 다음 버킷을 차례로 조사하는 개방 주소법이고, 체이닝은 같은 해시 주소에 속한 원소들을 연결리스트 등으로 연결하는 방법이다.

선지별 해설

① O — 해시 함수는 여러 탐색키를 제한된 수의 해시 주소에 대응시키므로, 서로 다른 탐색키가 같은 해시 주소로 사상되는 충돌이 발생할 수 있다.
② O — 충돌이 없다면 해시 함수로 주소를 계산한 뒤 해당 위치에 직접 접근하므로 탐색시간 복잡도는 $O(1)$ 이다.
③ X — 틀린 표현은 '선형 조사법은 연결리스트를 사용한다'이다. 선형 조사법은 충돌한 위치에서 다음 버킷들을 순차적으로 조사하는 개방 주소법이며, 연결리스트를 사용하는 대표적인 방법은 체이닝이다.
④ O — 폴딩 함수는 탐색키를 여러 부분으로 나눈 뒤 각 부분을 더하거나 배타적 논리합하여 해시 주소를 만든다.

NoteBase

오답 포인트

③은 충돌 해결 방법이라는 공통점 때문에 선형 조사법과 체이닝을 바꾸어 놓았다. '테이블 내부의 다음 빈 버킷 조사'는 선형 조사법이고, '같은 주소의 원소를 연결리스트로 연결'은 체이닝이다.

다시 나오면 이렇게 판단

충돌 해결 방식에서 빈 버킷을 직접 찾아 이동하면 개방 주소법, 같은 주소에 여러 원소를 연결하면 체이닝으로 판단한다. 선형 조사법은 현재 위치에서 일정 간격으로 다음 버킷을 확인하므로 개방 주소법에 속한다.

암기 포인트

선형 조사는 다음 칸, 체이닝은 연결리스트이다.

NoteBase

문 2. 후위표기법(Postfix Notation)으로 된 다음 식의 전위표기법(Prefix Notation)으로 옳은 것은?

$ABC+D/-AE+BF*/+$

① $+-A/+BCD/+AE*BF$

② $-+A/BC+D/+AE*BF$

③ $+-A/+BCD/+\ *AEBF$

④ $+A-/+BCD/+AE*BF$

정답 1번

출제 포인트

후위표기식을 스택으로 읽어 각 연산자의 두 피연산자를 확정한 뒤, 연산자를 피연산자 앞에 놓는 전위표기식으로 변환하는 문제이다.

조건 정리

주어진 후위표기식은 $ABC+D/-AE+BF*/+$이다. 피연산자는 스택에 넣고, 연산자를 만나면 먼저 꺼낸 항을 오른쪽 피연산자, 다음에 꺼낸 항을 왼쪽 피연산자로 삼는다.

사용할 원리

후위표기식의 부분식 $XYop$는 전위표기식 $opXY$로 바뀐다. 다만 복합식에서는 각 연산자가 결합하는 왼쪽 항과 오른쪽 항을 스택으로 먼저 확정해야 한다.

변환 과정

1. $BC+$를 $+BC$로 바꾸고, 이어지는 $D/$를 결합하면 $/+BCD$가 된다.
2. 앞의 A와 뺄셈을 결합하면 $-A/+BCD$가 된다.
3. $AE+$는 $+AE$, $BF*$는 $*BF$가 되고, 두 부분식을 나눗셈으로 결합하면 $/+AE*BF$가 된다.
4. 마지막 덧셈으로 두 큰 부분식을 결합하면 $+-A/+BCD/+AE*BF$가 된다.

NoteBase

선지 판단

① O — + − *A*/ + *BCD*/ + *AE* * *BF*는 두 큰 부분식 −*A*/ + *BCD*와 / + *AE* * *BF*를 마지막 +로 결합한 정확한 전위표기식이다.
② X — − + *A*/*BC* + *D*/ + *AE* * *BF*는 전체의 최상위 연산자를 뺄셈으로 두고 왼쪽을 +*A*/*BC*로 묶으므로 원식의 연산 구조와 다르다. 올바르게는 최상위 연산자를 덧셈으로 두어 + − *A*/ + *BCD*/ + *AE* * *BF*로 고쳐야 한다.
③ X — + − *A*/ + *BCD*/ + * *AEBF*는 두 번째 큰 부분식을 (*A* × *E* + *B*)/*F*의 구조로 만든다. 원식에서는 *A* + *E*를 *B* × *F*로 나누므로 / + * *AEBF*가 아니라 / + *AE* * *BF*로 고쳐야 한다.
④ X — +*A* − / + *BCD*/ + *AE* * *BF*는 *A*를 최상위 덧셈의 첫 피연산자로 두고 뒤의 두 분수 부분을 뺄셈으로 묶는다. 올바르게는 먼저 *A* − (*B* + *C*)/*D*를 만든 뒤 나머지 부분식과 더하여 + − *A*/ + *BCD*/ + *AE* * *BF*로 고쳐야 한다.

오답 포인트

함정은 연산자만 앞으로 옮기면서 피연산자의 결합 순서를 바꾸는 것이다. 특히 ②는 마지막 +를 최상위 연산자로 보지 못한 경우이고, ③은 *AE* +와 *BF* *라는 두 부분식의 경계를 잘못 잡은 경우이며, ④는 첫 번째 뺄셈 부분식의 경계를 놓친 경우이다.

다시 나오면 이렇게 판단

후위표기식을 왼쪽부터 읽으며 피연산자는 스택에 넣고, 연산자를 만나면 오른쪽 항과 왼쪽 항의 순서로 두 항을 꺼내 연산자+왼쪽 항+오른쪽 항 형태로 다시 넣는다. 마지막에 스택에 남은 하나가 전위표기식이다.

암기 포인트

후위식에서 연산자를 만나면 오른쪽 항을 먼저 꺼내고 왼쪽 항을 나중에 꺼낸다. 전위식에는 연산자를 가장 앞에 붙인다.

NoteBase

문 3. 다음 <보기>의 고객계좌 테이블에서 잔고가 1,000,000원에서 3,000,000원 사이인 고객들의 등급을 '우대고객'으로 변경하고자 <보기>와 같은 SQL문을 작성하였다. ㉠과 ㉡에 순서대로 들어갈 내용으로 가장 옳은 것은?

<보기>
UPDATE 고객계좌
(㉠) 등급 = '우대고객'
WHERE 잔고 (㉡) 1000000 AND 3000000

① SET, IN

② SET, BETWEEN

③ VALUES, IN

④ VALUES, BETWEEN

정답 2번

출제 포인트

UPDATE문에서 열의 값을 변경하는 SET 절과 범위를 지정하는 BETWEEN 연산자의 쓰임을 묻는 문제이다.

핵심 개념

SET: UPDATE문에서 변경할 열과 새 값을 지정하는 절이다. 기본 형태는 UPDATE 테이블명 SET 열 = 값 WHERE 조건이다.
BETWEEN: 하한과 상한 사이의 범위를 지정하는 조건 연산자이다. '잔고 BETWEEN 1000000 AND 3000000'은 두 경곗값을 포함한다.
IN: 괄호 안에 나열한 여러 값 중 하나와 일치하는지를 판단하는 연산자이다.
VALUES: 주로 INSERT문에서 삽입할 값을 지정할 때 사용한다.

NoteBase

선지별 해설

① X
㉠의 SET은 옳지만 ㉡의 IN이 틀렸다. IN은 'IN (값1, 값2, …)'처럼 값의 목록을 검사하며 '1000000 AND 3000000'으로 연속 범위를 나타내지 않는다. 올바르게는 SET, BETWEEN이다.

② O
UPDATE 고객계좌 SET 등급 = '우대고객' WHERE 잔고 BETWEEN 1000000 AND 3000000이 되어, 해당 범위의 고객 등급을 변경하는 올바른 문장이 된다.

③ X
㉠의 VALUES와 ㉡의 IN이 모두 틀렸다. UPDATE문에서 변경값은 SET으로 지정하고, 두 경곗값 사이의 범위는 BETWEEN ... AND ...로 지정한다. 올바르게는 SET, BETWEEN이다.

④ X
㉡의 BETWEEN은 옳지만 ㉠의 VALUES가 틀렸다. VALUES는 주로 INSERT문에서 삽입값을 지정하며 UPDATE문에서 열의 변경값은 SET으로 지정한다. 올바르게는 SET, BETWEEN이다.

오답 포인트

IN과 BETWEEN을 모두 WHERE 절의 조건 연산자로 본 뒤 용도를 구분하지 않거나, INSERT문의 VALUES를 UPDATE문의 값 지정 절로 혼동하기 쉽다. 이 문장에는 값 목록이 아니라 연속 범위가 제시되었으므로 BETWEEN이 필요하다.

다시 나오면 이렇게 판단

먼저 SQL 명령이 UPDATE인지 확인해 변경 열 앞에 SET을 둔다. 이어 조건이 나열된 값 중 하나이면 IN, 하한과 상한 사이이면 BETWEEN ... AND ...를 선택한다.

암기 포인트

UPDATE는 SET, 범위는 BETWEEN A AND B, 값 목록은 IN (...), 삽입값은 VALUES이다.

NoteBase

문 4. 다음 중 RISC와 CISC에 대한 설명 중 가장 옳지 않은 것은?

① RISC는 실행 빈도가 적은 하드웨어를 제거하여 자원 이용률을 높이는 장점이 있다.

② CISC는 고급 언어를 이용하여 알고리즘을 쉽게 표현 할 수 있는 장점이 있다.

③ RISC는 프로그램의 길이가 길어지므로 CISC보다 수행 속도가 느린 단점이 있다.

④ CISC는 복잡한 명령어군을 제공하므로 컴퓨터 설계 및 구현 시 많은 시간을 필요로 하는 단점이 있다.

정답 3번

출제 포인트

이 문제는 RISC와 CISC의 명령어 구성, 프로그램 길이, 실행 속도 및 하드웨어 설계 특성을 비교하는 문제이다.

핵심 개념

RISC는 자주 사용하는 단순한 명령어 중심으로 명령어 집합을 구성하여 명령어 실행과 파이프라이닝을 효율화하는 구조이다. CISC는 하나의 명령어가 여러 동작을 수행할 수 있도록 다양하고 복잡한 명령어를 제공하는 구조이다. 일반적으로 RISC는 명령어 수가 늘어 프로그램 길이가 길어질 수 있지만 명령어 자체가 단순하여 빠르게 처리할 수 있고, CISC는 프로그램을 짧게 표현하기 쉽지만 명령어 해석과 하드웨어 제어가 복잡해진다.

선지별 해설

① O — RISC는 사용 빈도가 낮은 복잡한 명령어를 하드웨어 명령어 집합에서 줄이고 자주 쓰는 단순 명령어에 자원을 집중하므로 자원 이용 효율을 높일 수 있다.
② O — CISC는 고급 언어의 복잡한 연산에 대응하는 다양한 명령어를 제공하므로 알고리즘을 비교적 적은 수의 기계 명령어로 표현하기 쉽다.
③ X — 틀린 표현은 '프로그램의 길이가 길어지므로 CISC보다 수행 속도가 느리다'이다. RISC는 프로그램 길이가 길어질 수 있으나, 명령어가 단순하고 형식이 규칙적이어서 빠른 실행과 효율적인 파이프라이닝에 유리하다. 따라서 프로그램 길이가 길다는 사실만으로 CISC보다 수행 속도가 느리다고 할 수 없다.
④ O — CISC는 복잡하고 다양한 명령어를 하드웨어로 해석·제어해야 하므로 프로세서의 설계와 구현이 복잡해지고 더 많은 개발 시간이 필요할 수 있다.

오답 포인트

③은 '프로그램 길이'와 '수행 속도'를 같은 방향으로 판단하게 만드는 함정이다. RISC는 한 작업에 필요한 명령어 수가 많아질 수 있지만 각 명령어의 구조와 실행이 단순하므로 전체 수행 속도가 반드시 느려지는 것은 아니다.

NoteBase

다시 나오면 이렇게 판단

명령어 개수와 명령어당 처리 복잡도를 나누어 판단한다. RISC는 단순한 명령어·긴 프로그램·빠른 명령어 처리·파이프라이닝 용이, CISC는 복잡한 명령어·짧은 프로그램·복잡한 제어·설계 난도 증가로 구분한다.

암기 포인트

RISC는 명령어는 단순하고 코드는 길어질 수 있으며, CISC는 명령어는 복잡하고 코드는 짧아질 수 있다. 코드가 길다고 실행까지 반드시 느린 것은 아니다.

NoteBase

문 5. IPv4 주소체계 기반의 어떤 네트워크 상에서 두 컴퓨터 A, B가 각각 192.168.0.1과 192.168.0.65의 주소를 사용할 때, 이 두 컴퓨터가 서로 다른 서브넷(Subnet) 상에 존재하기 위해 사용해야 하는 서브넷 마스크(Subnet Mask)로 가장 옳은 것은?

① 0.0.0.0

② 255.255.255.0

③ 255.255.255.192

④ 255.255.255.128

정답 3번

출제 포인트

두 IPv4 주소에 서브넷 마스크를 비트 AND 연산하여 얻은 네트워크 주소가 서로 달라지는 마스크를 찾는 문제이다.

조건 정리

A의 주소는 192.168.0.1이고, B의 주소는 192.168.0.65이다. 두 주소는 마지막 옥텟에서만 다르므로 마지막 옥텟을 비교한다.
A: 1 = 00000001
B: 65 = 01000001
두 컴퓨터가 서로 다른 서브넷에 있으려면 두 주소의 네트워크 주소가 달라야 한다.

사용할 공식

네트워크 주소=IP 주소\&서브넷 마스크

서브넷은 네트워크 주소로 구분되므로 각 IP 주소와 후보 마스크를 비트 AND 연산하여 결과를 비교한다.

계산 과정

255.255.255.192는 /26 마스크이며 마지막 옥텟은 11000000이다.
A: 00000001 AND 11000000 = 00000000이므로 네트워크 주소는 192.168.0.0이다.
B: 01000001 AND 11000000 = 01000000이므로 네트워크 주소는 192.168.0.64이다.
두 네트워크 주소가 다르므로 A와 B는 서로 다른 서브넷에 속한다.

NoteBase

선지 판단

① X — 0.0.0.0은 /0 마스크이므로 모든 IPv4 주소의 네트워크 주소가 0.0.0.0이 된다. 따라서 A와 B를 서로 다른 서브넷으로 나누지 못한다. 올바르게는 두 주소를 구분하는 255.255.255.192를 사용해야 한다.
② X — 255.255.255.0은 /24 마스크이므로 A와 B의 네트워크 주소가 모두 192.168.0.0이다. 따라서 같은 서브넷에 속한다. 올바르게는 /24가 아니라 255.255.255.192인 /26 마스크를 사용해야 한다.
③ O — 255.255.255.192는 /26 마스크이다. A의 네트워크 주소는 192.168.0.0, B의 네트워크 주소는 192.168.0.64가 되어 서로 다른 서브넷에 속한다.
④ X — 255.255.255.128은 /25 마스크로 마지막 옥텟의 범위를 0~127과 128~255로 나눈다. 1과 65는 모두 0~127 범위이므로 같은 192.168.0.0 서브넷에 속한다. 올바르게는 경계를 64 단위로 만드는 255.255.255.192인 /26 마스크를 사용해야 한다.

오답 포인트

65가 64보다 크다는 이유만으로 /25에서도 다른 서브넷이라고 판단하기 쉽다. /25의 실제 경계는 128이고, /26의 경계가 64이다. 따라서 1과 65를 나누는 것은 /26이다.

다시 나오면 이렇게 판단

각 후보 마스크에 대해 서브넷 블록 크기 또는 비트 AND 결과를 구한다. 두 IP 주소가 서로 다른 블록에 들어가는지 확인하고, 모든 선지의 네트워크 주소를 비교한다.

암기 포인트

255.255.255.192는 /26이며 마지막 옥텟을 64개씩 0~63, 64~127, 128~191, 192~255로 나눈다.

NoteBase

문 6. 다음 <보기> 중 잘 알려진 포트번호(Well-Known Port)와 TCP 프로토콜이 바르게 연결된 것을 모두 고른 것은?

<보기>
㉠ 21번 포트 : FTP
㉡ 67번 포트 : DHCP
㉢ 23번 포트 : SMTP
㉣ 80번 포트 : HTTP

① ㉠, ㉡
② ㉠, ㉣
③ ㉡, ㉢
④ ㉡, ㉣

정답 2번

출제 포인트

잘 알려진 포트 번호에 대응하는 응용 계층 프로토콜과 그 프로토콜이 사용하는 전송 계층 프로토콜을 함께 구별하는 문제이다.

핵심 개념

FTP의 제어 연결은 TCP 21번, SMTP는 TCP 25번, Telnet은 TCP 23번, HTTP는 TCP 80번 포트를 사용한다. DHCP는 서버가 UDP 67번, 클라이언트가 UDP 68번 포트를 사용한다. 따라서 TCP를 사용하는 올바른 연결은 ㉠과 ㉣이다.

선지별 해설

① X — ㉠의 '21번 포트 : FTP'는 옳지만, ㉡의 '67번 포트 : DHCP'는 TCP 연결이 아니다. DHCP는 UDP 67번·68번 포트를 사용하므로, TCP 프로토콜의 올바른 연결 조합은 ㉠, ㉣이다.
② O — FTP의 제어 연결은 TCP 21번 포트를 사용하고 HTTP는 TCP 80번 포트를 사용하므로 ㉠과 ㉣이 모두 옳다.
③ X — ㉡은 DHCP가 UDP 67번 포트를 사용한다는 점에서 TCP 조건에 맞지 않고, ㉢의 '23번 포트 : SMTP'도 잘못된 연결이다. 올바르게는 TCP 23번은 Telnet, SMTP는 TCP 25번이다. 따라서 올바른 조합은 ㉠, ㉣이다.
④ X — ㉣의 '80번 포트 : HTTP'는 옳지만, ㉡의 DHCP는 TCP가 아니라 UDP 67번 포트를 사용한다. 따라서 TCP 프로토콜의 올바른 연결 조합은 ㉠, ㉣이다.

NoteBase

오답 포인트

㉡은 'DHCP와 67번' 자체는 맞아 보이지만 문제의 TCP 조건 때문에 틀린 보기이다. 또한 ㉢은 TCP 23번이라는 점만 보고 맞다고 판단하기 쉽지만, 23번의 서비스는 SMTP가 아니라 Telnet이며 SMTP의 기본 포트는 25번이다.

다시 나오면 이렇게 판단

각 보기에서 '서비스명－포트 번호'가 맞는지 먼저 확인하고, 이어서 TCP와 UDP 중 어느 전송 계층 프로토콜을 사용하는지 확인한다. 포트 번호만 맞아도 전송 계층 조건이 다르면 오답이다.

암기 포인트

FTP 21 · SMTP 25 · Telnet 23 · HTTP 80은 TCP, DHCP 67 · 68은 UDP이다.

NoteBase

문 7. 다음 중 RAID 기술에 대한 설명으로 가장 옳지 않은 것은?

① RAID 1레벨은 미러링(Mirroring)을 지원한다.

② RAID 3레벨은 데이터를 블록 단위로 분산 저장하여 대용량의 읽기 중심 서버용으로 사용한다.

③ RAID 5레벨은 고정적인 패리티 디스크 대신 패리티가 모든 디스크에 분산되어 저장되므로 병목현상을 줄여준다.

④ RAID 6레벨은 두 개의 패리티 디스크를 사용하므로 두 개의 디스크 장애 시에도 데이터의 복구가 가능하다.

정답 2번

출제 포인트

RAID 각 레벨의 데이터 분산 단위, 패리티 배치 방식, 장애 허용 능력을 구분하는 문제이다.

핵심 개념

RAID 1은 같은 데이터를 복제해 저장하는 미러링 방식이다. RAID 3은 데이터를 바이트 단위로 스트라이핑하고 전용 패리티 디스크를 사용하여 대용량 순차 입출력에 적합하다. RAID 5는 블록 단위 스트라이핑과 분산 패리티를 사용해 패리티 디스크의 병목을 줄인다. RAID 6은 서로 독립적인 이중 패리티를 분산 저장하여 디스크 2개가 동시에 고장 나도 데이터를 복구할 수 있다.

선지별 해설

① O — RAID 1은 동일한 데이터를 둘 이상의 디스크에 복제하는 미러링을 지원하므로 옳다.
② X — 틀린 표현은 '블록 단위로 분산 저장하여 읽기 중심 서버용으로 사용한다'이다. RAID 3은 데이터를 바이트 단위로 분산하고 전용 패리티 디스크를 사용하며, 여러 디스크가 함께 동작하는 대용량 순차 입출력에 적합하다. 블록 단위 스트라이핑과 독립적인 읽기 요청 처리에 유리한 특성은 RAID 5와 구별해야 한다.
③ O — RAID 5는 패리티 정보를 모든 디스크에 분산하므로 하나의 고정 패리티 디스크에 입출력이 집중되는 병목을 줄인다.
④ O — RAID 6은 두 종류의 독립적인 패리티 정보를 사용하므로 디스크 2개가 동시에 고장 나도 데이터를 복구할 수 있다. 여기서 '두 개의 패리티 디스크'는 고정된 전용 디스크 2개라기보다 패리티에 디스크 2개 분량의 용량을 사용하는 것으로 이해한다.

오답 포인트

②는 RAID 3의 '전용 패리티'라는 특징과 다른 RAID 레벨의 '블록 단위 스트라이핑'을 섞은 선지이다. RAID 3은 바이트 단위, RAID 5는 블록 단위라는 차이가 핵심 함정이다.

NoteBase

다시 나오면 이렇게 판단

RAID 선지는 데이터 분산 단위와 패리티 위치를 따로 확인한다. RAID 1은 미러링, RAID 3은 바이트 단위·전용 패리티, RAID 5는 블록 단위·분산 패리티, RAID 6은 이중 분산 패리티와 2개 디스크 장애 허용으로 판단한다.

암기 포인트

1은 복제, 3은 바이트·전용 패리티, 5는 블록·분산 패리티, 6은 이중 패리티이다.

NoteBase

문 8. 다음 중 인터럽트 우선순위를 결정하는 Polling방식에 대한 설명으로 가장 옳지 않은 것은?

① 많은 인터럽트 발생 시 처리시간 및 반응시간이 매우 빠르다.

② S/W적으로 CPU가 각 장치 하나하나를 차례로 조사하는 방식이다.

③ 조사 순위가 우선순위가 된다.

④ 모든 인터럽트를 위한 공통의 서비스 루틴을 갖고 있다.

정답 1번

출제 포인트

인터럽트 우선순위를 소프트웨어로 판별하는 폴링 방식의 동작 원리와 속도 특성을 묻는 문제이다.

핵심 개념

폴링 방식은 공통 인터럽트 처리 루틴에서 CPU가 각 장치의 인터럽트 요청 여부를 정해진 순서대로 검사하여 원인을 찾는 소프트웨어 우선순위 방식이다. 먼저 조사하는 장치일수록 우선순위가 높다. 장치를 하나씩 확인하므로 하드웨어 우선순위 방식보다 판별이 느리고, 인터럽트가 많을수록 처리 부담과 반응 지연이 커질 수 있다.

선지별 해설

① X — 틀린 표현은 '많은 인터럽트 발생 시 처리시간 및 반응시간이 매우 빠르다'이다. 폴링은 CPU가 장치들을 차례로 조사하므로 인터럽트 요청이 많거나 조사 대상 장치가 많으면 우선순위 판별에 시간이 걸린다.
올바르게는 '많은 인터럽트가 발생하면 처리시간과 반응시간이 느려질 수 있다'이다.
② O — 폴링은 별도의 우선순위 판별 하드웨어에 의존하기보다 소프트웨어 루틴으로 CPU가 각 장치의 상태 또는 인터럽트 플래그를 순서대로 조사하는 방식이다.
③ O — 장치를 조사하는 순서가 곧 우선순위가 된다. 검사 순서가 앞선 장치의 요청이 먼저 발견되어 처리된다.
④ O — 여러 인터럽트가 공통 인터럽트 처리 루틴으로 진입한 뒤, 그 루틴이 각 장치를 폴링하여 인터럽트 원인을 식별한다. 원인을 찾은 뒤에는 해당 장치의 처리 부분으로 분기할 수 있다.

오답 포인트

①의 '매우 빠르다'가 함정이다. 폴링은 구성이 간단하고 우선순위를 소프트웨어로 바꾸기 쉽지만, 순차 검사 때문에 많은 인터럽트에 대한 빠른 반응이 장점인 방식은 아니다.

다시 나오면 이렇게 판단

'CPU가 순서대로 조사한다'가 보이면 소프트웨어 폴링으로 판단한다. 조사 순서는 우선순위를 뜻하며, 순차 검사라는 말에서 속도 저하 가능성을 함께 도출한다.

NoteBase

암기 포인트

폴링은 소프트웨어 순차 조사, 조사 순서가 우선순위, 인터럽트가 많으면 반응은 느려진다.

NoteBase

문 9. 운영체제는 일괄처리(Batch), 대화식(Interactive), 실시간(Real-Time) 시스템 그리고 일괄처리와 대화식이 결합된 혼합 시스템(Hybrid System) 등으로 분류될 수 있다. 이와 같은 분류 근거로 가장 옳은 것은?

① 응답 시간과 데이터 입력 방식

② 데이터 보호의 필요성 여부

③ 버퍼링(Buffering) 기능 수행 여부

④ 고급 프로그램 언어의 사용 여부

정답 1번

출제 포인트

운영체제의 처리 시스템을 일괄처리, 대화식, 실시간, 혼합 시스템으로 나누는 기준이 무엇인지 묻는 문제이다.

핵심 개념

일괄처리 시스템은 작업과 데이터를 모아 한꺼번에 처리하는 방식이고, 대화식 시스템은 사용자의 입력과 시스템의 응답이 서로 오가며 처리되는 방식이다. 실시간 시스템은 정해진 시간 안에 결과를 내야 하는 방식이며, 혼합 시스템은 일괄처리와 대화식 처리의 특성을 결합한 방식이다. 따라서 이 분류는 사용자가 데이터를 입력하는 방식과 시스템이 응답해야 하는 시간적 특성을 기준으로 한다.

선지별 해설

① O — 일괄처리는 작업을 모아 처리하고, 대화식 처리는 입력에 응답하며, 실시간 처리는 정해진 시간 안에 응답한다. 따라서 응답 시간과 데이터 입력 방식이 이들을 구분하는 기준이다.
② X — 틀린 표현은 '데이터 보호의 필요성 여부'이다. 데이터 보호 필요성은 보안·보호 정책과 관련된 기준이며, 올바르게는 '응답 시간과 데이터 입력 방식'으로 분류한다.
③ X — 틀린 표현은 '버퍼링 기능 수행 여부'이다. 버퍼링은 입출력 장치와 처리 장치 사이의 속도 차이를 완화하는 기능으로 여러 처리 시스템에서 사용할 수 있다. 올바르게는 '응답 시간과 데이터 입력 방식'을 기준으로 구분한다.
④ X — 틀린 표현은 '고급 프로그램 언어의 사용 여부'이다. 프로그래밍 언어의 수준은 프로그램 작성 도구에 관한 구분일 뿐 처리 시스템의 운영 방식과 직접 관련되지 않는다. 올바르게는 '응답 시간과 데이터 입력 방식'을 기준으로 구분한다.

오답 포인트

버퍼링이나 고급 프로그램 언어도 운영체제와 관련된 용어이므로 분류 기준처럼 보일 수 있다. 그러나 선지에 제시된 네 시스템의 이름은 작업을 언제 입력하고 시스템이 언제 응답하는지를 직접 나타낸다.

NoteBase

다시 나오면 이렇게 판단

처리 시스템의 분류 문제에서는 각 명칭이 나타내는 작업 투입 방식과 응답 시점을 확인한다. ‘모아서 처리’, ‘사용자와 상호작용’, ‘제한 시간 내 처리’가 대비되면 입력 방식과 응답 시간이 분류 기준이다.

암기 포인트

일괄은 모아서, 대화식은 주고받으며, 실시간은 제시간 안에 처리한다.

NoteBase

문 10. 다음 <보기>의 JAVA 프로그램 실행 결과로 가장 옳은 것은?

<보기>

```
class Test {
public static void main(String[] args) {
int a = 101;
System.out.println((a>>3) << 2);
}
}
```

① 0

② 48

③ 404

④ 600

정답 2번

출제 포인트

이 문제는 Java의 오른쪽 시프트와 왼쪽 시프트를 괄호에 따라 차례로 적용하여 출력값을 구하는 문제이다.

조건 정리

주어진 값: 정수 $a = 101$
구해야 할 값: (a >> 3) << 2의 결과
주의할 조건: 괄호 안의 오른쪽 시프트를 먼저 계산한 뒤 왼쪽 시프트를 적용한다. a는 양수이므로 a >> 3은 a를 2^3으로 나눈 몫과 같다.

사용할 공식

양의 정수에서 x >> n은 x를 2^n으로 나눈 몫이고, x << n은 x에 2^n을 곱한 값이다.
이 원리를 쓰는 이유는 시프트 연산자가 이진수의 비트를 지정한 수만큼 이동시키기 때문이다.

계산 과정

1. 101 >> 3 = 101 ÷ 2^3의 정수 몫이므로 12이다.
2. 이어서 12 << 2 = 12 × 2^2 = 48이다.
3. 따라서 System.out.println이 출력하는 값은 48이다.

NoteBase

선지 판단

① X — 0은 계산 결과가 아니다. 오른쪽 시프트 후에도 값은 12이므로, 올바른 출력값은 48이다.
② O — 101 >> 3=12이고 12 << 2=48이므로 실행 결과와 일치한다.
③ X — 404는 101에 왼쪽 시프트만 적용한 101 << 2의 값이다. 괄호 안의 오른쪽 시프트를 빠뜨린 결과이며, 올바른 출력값은 48이다.
④ X — 600은 두 시프트 연산을 차례로 적용한 결과가 아니다. 올바른 계산은 (101 >> 3) << 2=48이다.

오답 포인트

오른쪽으로 3비트 이동한 뒤 왼쪽으로 2비트 이동한다고 해서 원래 값으로 돌아오지 않는다. 첫 연산에서 버려진 하위 3비트는 복구되지 않으며, ③의 404는 첫 번째 오른쪽 시프트를 누락할 때 나오는 값이다.

다시 나오면 이렇게 판단

괄호와 연산 순서에 따라 각 시프트를 한 단계씩 계산한다. 양의 정수에서는 오른쪽 시프트를 2^n으로 나눈 정수 몫, 왼쪽 시프트를 2^n을 곱한 값으로 바꾸어 검산한다.

암기 포인트

양의 정수 기준으로 >> n은 2^n으로 나눈 몫, << n은 2^n을 곱한 값이다. 오른쪽 시프트에서 버린 비트는 다시 왼쪽으로 이동해도 돌아오지 않는다.

NoteBase

문 11. 다음 중 Flynn의 병렬컴퓨터 분류방식에 대한 설명으로 가장 옳지 않은 것은?

① SISD : 명령어와 데이터를 순서대로 처리하는 단일프로세서 시스템이다.

② SIMD : 단일 명령어 스트림을 처리하며 배열 프로세서라고도 한다.

③ MISD : 여러 개의 프로세서를 갖는 구조로 밀결합 시스템(tightly-coupled system)과 소결합 시스템(loosely-coupled system)으로 분류한다.

④ MIMD : 여러 개의 프로세서들이 서로 다른 명령어와 데이터를 처리하는 진정한 의미의 병렬 프로세서이다.

정답 3번

출제 포인트

Flynn 분류에서 명령어 스트림과 데이터 스트림의 수에 따라 SISD, SIMD, MISD, MIMD를 구별하는 문제이다.

핵심 개념

SISD는 단일 명령어 스트림으로 단일 데이터 스트림을 처리하는 구조이다. SIMD는 하나의 명령어를 여러 데이터에 동시에 적용하는 구조이다. MISD는 여러 명령어 스트림이 하나의 데이터 스트림을 처리하는 구조이다. MIMD는 여러 명령어 스트림이 여러 데이터 스트림을 각각 처리하는 구조이다.

선지별 해설

① O — SISD는 한 명령어 스트림과 한 데이터 스트림을 순차적으로 처리하는 전형적인 단일 프로세서 구조이므로 옳다.
② O — SIMD는 하나의 명령어 스트림으로 여러 데이터 요소를 병렬 처리하며, 배열 프로세서가 대표적인 구현이므로 옳다.
③ X — 틀린 부분은 MISD를 밀결합 시스템과 소결합 시스템으로 분류한 것이다. MISD는 여러 명령어 스트림이 하나의 데이터 스트림을 처리하는 구조이며, 밀결합·소결합 구분은 여러 프로세서가 각기 다른 명령어와 데이터를 처리하는 MIMD 시스템에 적용된다.
④ O — MIMD는 여러 프로세서가 서로 다른 명령어 스트림으로 서로 다른 데이터 스트림을 동시에 처리하는 병렬 처리 구조이므로 옳다.

오답 포인트

③은 '여러 개의 프로세서'라는 공통 표현을 이용해 MIMD의 결합 방식 분류를 MISD의 특징처럼 바꾼 선지이다. Flynn 분류의 핵심은 프로세서 수 자체가 아니라 명령어 스트림과 데이터 스트림이 각각 단일인지 다중인지이다.

다시 나오면 이렇게 판단

각 약어의 앞 글자는 명령어 스트림, 뒤 글자는 데이터 스트림을 나타낸다고 판단한다. 따라서 MISD는 다중 명령어·단일 데이터이고, 다중 명령어·다중 데이터 및 밀결합·소결합 분류는 MIMD와 연결된다.

NoteBase

암기 포인트

앞은 명령어, 뒤는 데이터이다. SISD는 1·1, SIMD는 1·다, MISD는 다·1, MIMD는 다·다이다.

NoteBase

문 12. 다음 표는 각 프로세스의 제출시간과 CPU 수행시간을 나타낸 것이다. HRN(Highest Response-ratio Next) 스케줄링 기법을 사용하였을 때, 우선순위가 가장 높은 프로세스는 무엇인가?

프로세스	제출시간	CPU 수행시간
P1	15	5
P2	8	4
P3	12	3
P4	10	2

① P1

② P2

③ P3

④ P4

정답 4번

출제 포인트

이 문제는 HRN 스케줄링의 응답률을 계산하여 가장 큰 프로세스를 고르는 문제이다.

조건 정리

가장 늦게 제출된 P1의 제출시간인 15를 스케줄링 시점으로 둔다.
대기시간은 현재 시각에서 제출시간을 뺀 값이다.
P1, P2, P3, P4의 대기시간은 각각 0, 7, 3, 5이다.

사용할 공식

HRN의 응답률은 다음과 같다.
응답률=대기시간+CPU 수행시간CPU 수행시간=1+대기시간CPU 수행시간
HRN은 준비 큐에 있는 프로세스 중 이 응답률이 가장 큰 프로세스를 다음 실행 대상으로 선택한다.

NoteBase

계산 과정

P1: $\frac{0+5}{5} = 1$
P2: $\frac{7+4}{4} = \frac{11}{4} = 2.75$
P3: $\frac{3+3}{3} = 2$
P4: $\frac{5+2}{2} = \frac{7}{2} = 3.5$
응답률이 가장 큰 프로세스는 P4이다.

선지 판단

① P1 — X. 응답률이 1이므로 가장 높지 않다. ‘P1’은 틀린 선택이며 올바른 프로세스는 P4이다.
② P2 — X. 응답률은 2.75로 P4의 3.5보다 작다. ‘P2’는 틀린 선택이며 올바른 프로세스는 P4이다.
③ P3 — X. 응답률은 2로 P4의 3.5보다 작다. ‘P3’은 틀린 선택이며 올바른 프로세스는 P4이다.
④ P4 — O. 응답률이 3.5로 네 프로세스 중 가장 크므로 우선순위가 가장 높다.

오답 포인트

제출시간이 빠른 프로세스를 무조건 먼저 고르거나 CPU 수행시간이 짧은 프로세스를 무조건 먼저 고르면 틀린다. HRN은 대기시간과 CPU 수행시간을 함께 반영한 응답률을 비교한다.

다시 나오면 이렇게 판단

스케줄링 시점을 정한 뒤 각 프로세스의 대기시간을 계산하고, HRN 응답률 공식에 대입하여 가장 큰 값을 선택한다.

암기 포인트

HRN은 1+대기시간CPU 수행시간이 가장 큰 프로세스를 선택한다.

NoteBase

문 13. 다음 정수 리스트를 퀵 정렬 알고리즘으로 오름차순 정렬할 때, 리스트를 처음 분할한 직후 분할된 두 리스트의 상태로 가장 옳은 것은? (단, 제어키는 5로 한다.)

(5 2 6 4 7 3 8 1)

① (1 2 3 4), (6 7 8)
② (2 4 3 1), (6 7 8)
③ (3 1 2 4), (7 6 8)
④ (3 2 1 4), (7 8 6)

정답 4번

출제 포인트

이 문제는 제어키 5를 기준으로 작은 값은 왼쪽, 큰 값은 오른쪽에 놓이도록 퀵 정렬의 첫 분할을 수행한 결과를 묻는다.

올바른 순서

1. 왼쪽에서 5보다 큰 6을 찾고, 오른쪽에서 5보다 작은 1을 찾아 서로 교환한다. 리스트는 (5 2 1 4 7 3 8 6)이 된다.
2. 탐색을 이어서 왼쪽의 7과 오른쪽의 3을 찾아 서로 교환한다. 리스트는 (5 2 1 4 3 7 8 6)이 된다.
3. 두 탐색 위치가 교차하면 제어키 5와 3을 교환하여 제어키의 위치를 확정한다. 리스트는 (3 2 1 4 5 7 8 6)이 된다.
4. 제어키 5를 제외하면 처음 분할된 두 리스트는 (3 2 1 4)와 (7 8 6)이다.

왜 이 순서인가: 첫 분할은 각 부분 리스트를 완전히 정렬하는 과정이 아니라, 제어키보다 작은 값과 큰 값을 제어키의 양쪽으로 나누는 과정이기 때문이다.

선지별 해설

① X — (1 2 3 4)와 (6 7 8)은 양쪽 부분 리스트까지 오름차순으로 정렬된 상태이므로 첫 분할 직후의 상태가 아니다. 틀린 부분은 각 부분 리스트가 이미 정렬되어 있다는 점이며, 올바르게는 (3 2 1 4)와 (7 8 6)이다.
② X — 왼쪽의 6과 1, 이어서 7과 3을 교환한 뒤 제어키를 옮기면 (2 4 3 1)이 만들어지지 않는다. 틀린 부분은 두 부분 리스트의 내부 순서이며, 올바르게는 (3 2 1 4)와 (7 8 6)이다.
③ X — 첫 분할 과정의 교환 결과는 왼쪽이 (3 2 1 4), 오른쪽이 (7 8 6)이므로 (3 1 2 4)와 (7 6 8)은 실제 분할 결과와 다르다. 틀린 부분은 왼쪽의 1·2와 오른쪽의 6·8의 위치이며, 올바르게는 (3 2 1 4)와 (7 8 6)이다.
④ O — 첫 분할이 끝나면 제어키 5보다 작은 3, 2, 1, 4가 왼쪽에, 큰 7, 8, 6이 오른쪽에 놓이므로 (3 2 1 4)와 (7 8 6)이 된다.

NoteBase

오답 포인트

①처럼 분할 직후를 전체 정렬이 끝난 상태로 착각하기 쉽다. 첫 분할에서는 제어키 5의 최종 위치만 확정되며, 왼쪽과 오른쪽 부분 리스트의 내부 순서는 아직 정렬되지 않는다.

다시 나오면 이렇게 판단

제어키를 기준으로 왼쪽에서는 제어키보다 큰 값을, 오른쪽에서는 제어키보다 작은 값을 찾아 교환한다. 탐색 위치가 교차하면 제어키를 경계 위치로 옮기고, 제어키를 제외한 양쪽 리스트를 확인한다.

암기 포인트

퀵 정렬의 한 번의 분할은 완전 정렬이 아니라 제어키의 자리 확정이다.

NoteBase

문 14. 다음 중 소프트웨어 프로토타이핑(Prototyping)에 대한 설명으로 가장 옳지 않은 것은?

① 개발자가 구축할 소프트웨어의 모델을 사전에 만드는 공정으로서 요구사항을 효과적으로 유도, 수집한다.

② 프로토타입은 기능적으로 제품의 하위 기능을 담당하는 작동 가능한 모형이다.

③ 프로토타이핑에 의해 만들어진 프로토타입은 폐기될 수 있고, 재사용될 수도 있다.

④ 적용사례가 많고, 가장 오래됐으며 널리 사용되는 방법으로 결과물이 명확하므로 가시성이 매우 좋다.

정답 4번

출제 포인트

프로토타이핑의 목적과 프로토타입의 성격을 폭포수 모형의 특징과 구별하는 문제이다.

핵심 개념

프로토타이핑은 실제 시스템을 완성하기 전에 작동 가능한 시제품을 만들어 사용자 요구를 확인하고 구체화하는 개발 방법이다. 프로토타입은 요구사항 파악 후 폐기하는 폐기형일 수도 있고, 계속 보완하여 최종 시스템으로 발전시키는 진화형일 수도 있다. 반면 폭포수 모형은 개발 단계를 순차적으로 진행하며 단계별 산출물과 문서가 명확한 전통적 모형이다.

선지별 해설

① O — 프로토타이핑은 개발할 소프트웨어의 모형을 먼저 만들어 사용자의 반응을 확인하므로 불명확한 요구사항을 효과적으로 유도하고 수집할 수 있다.
② O — 프로토타입은 전체 제품을 완성한 것이 아니라 핵심 기능이나 일부 하위 기능을 실제로 작동하도록 구현한 모형이다.
③ O — 폐기형 프로토타이핑에서는 요구사항을 확인한 뒤 프로토타입을 버리고, 진화형 프로토타이핑에서는 이를 반복적으로 개선하여 최종 시스템의 일부 또는 기반으로 재사용할 수 있다.
④ X — 틀린 부분은 프로토타이핑을 '가장 오래됐으며 널리 사용되고, 결과물이 명확하여 가시성이 매우 좋은 방법'이라고 한 표현이다. 이는 단계별 산출물과 문서가 명확한 전통적 폭포수 모형의 특징에 가깝다. 올바르게는 프로토타이핑은 사용자 피드백을 통해 요구사항을 구체화하고 변경에 유연하게 대응하는 방법이라고 해야 한다.

오답 포인트

④는 '사전에 모형을 만든다'는 프로토타이핑의 설명이 아니라 '오래된 전통적 방법', '명확한 단계별 결과물', '높은 가시성'이라는 폭포수 모형의 특징을 섞은 선지이다.

다시 나오면 이렇게 판단

사용자 피드백·요구사항 구체화·시제품의 폐기 또는 진화가 나오면 프로토타이핑으로, 순차적 단계·명확한 문서와 산출물·전통적 개발 모형이 나오면 폭포수 모형으로 판단한다.

NoteBase

암기 포인트

프로토타이핑은 ‘먼저 만들어 요구를 확인’, 폭포수 모형은 ‘단계대로 진행하고 산출물을 명확히’이다.

NoteBase

문 15. 다음 중 OSI(Open System Interconnection) 7계층에 대한 설명으로 가장 옳지 않은 것은?

① 물리계층에서는 기계적·전기적·기능적·절차적 특징을 다루어 물리적 매체를 전송한다.

② 수신 측에서 패킷을 수신하게 되면, 상위 계층에서 하위 계층 순으로 처리된다.

③ 전송계층에서는 종점 간의 에러복구와 흐름제어를 담당한다.

④ 세션계층에서는 서로 협력하는 응용프로그램(Applications)들에 대하여 연결을 설정, 유지, 종료한다.

정답 2번

출제 포인트

OSI 7계층의 계층별 기능과 송신·수신 과정의 처리 방향을 구별하는 문제이다.

핵심 개념

OSI 7계층은 네트워크 통신 기능을 응용·표현·세션·전송·네트워크·데이터링크·물리 계층으로 나눈 참조 모델이다. 송신 측에서는 상위 계층에서 하위 계층으로 캡슐화하며 처리하고, 수신 측에서는 하위 계층에서 상위 계층으로 역캡슐화하며 처리한다. 물리 계층은 비트 전송, 전송 계층은 종점 간 신뢰성 및 흐름 제어, 세션 계층은 응용 프로그램 사이의 세션 관리를 담당한다.

선지별 해설

① O — 물리 계층은 전송 매체와 인터페이스의 기계적·전기적·기능적·절차적 특성을 규정하고, 물리 매체를 통해 비트열을 전송한다.
② X — 틀린 부분은 수신 측의 처리 순서를 '상위 계층에서 하위 계층 순'이라고 한 것이다. 실제로 수신 측은 물리 계층에서 응용 계층으로, 즉 하위 계층에서 상위 계층 순으로 처리한다.
③ O — 전송 계층은 송신 종점과 수신 종점 사이에서 오류 제어와 흐름 제어를 수행하여 종점 간 전송의 신뢰성을 제공한다.
④ O — 세션 계층은 서로 통신하는 응용 프로그램 사이의 세션을 설정하고 유지하며 종료하는 기능을 담당한다.

오답 포인트

②는 송신 측의 처리 방향을 수신 측에도 그대로 적용하게 만드는 선지이다. 송신은 상위 계층에서 하위 계층으로 진행하지만, 수신은 그 반대인 하위 계층에서 상위 계층으로 진행한다.

다시 나오면 이렇게 판단

계층별 기능 문제에서는 먼저 송신인지 수신인지 확인한 뒤 처리 방향을 판단한다. 이어서 물리 계층은 비트 전송, 전송 계층은 종점 간 제어, 세션 계층은 연결의 설정·유지·종료라는 핵심어로 대응한다.

NoteBase

암기 포인트

송신은 위에서 아래로, 수신은 아래에서 위로 처리한다. 물리는 비트, 전송은 종점 간 제어, 세션은 연결 관리이다.

NoteBase

문 16. 다음 중 HTML5의 특징에 대한 설명으로 가장 옳지 않은 것은?

① 쌍방향 통신을 제공하여 실시간 채팅이나 온라인 게임을 만들 수 있다.

② 스마트폰의 일반 응용프로그램도 HTML5를 사용해 개발할 수 있다.

③ 디바이스에 접근할 수 없어서 개인정보 보호 및 보안을 철저히 유지할 수 있다.

④ 플러그인의 도움 없이 음악과 동영상 재생이 가능하다.

정답 3번

출제 포인트

HTML5가 제공하는 실시간 통신, 모바일 응용프로그램 개발, 디바이스 연동, 멀티미디어 재생 기능의 특징을 구별하는 문제이다.

핵심 개념

HTML5는 웹 문서 표현을 넘어 통신, 멀티미디어, 저장소와 각종 웹 API를 활용해 응용프로그램 수준의 기능을 구현할 수 있도록 확장된 웹 표준이다. 다만 디바이스 기능에 접근할 수 있다는 사실과 개인정보 보호·보안이 자동으로 보장된다는 주장은 구분해야 한다.

선지별 해설

① O — WebSocket 등의 기술을 이용하면 클라이언트와 서버 사이의 양방향 통신을 구현할 수 있으므로 실시간 채팅이나 온라인 게임에 활용할 수 있다.
② O — HTML5, CSS, JavaScript 기반의 웹 기술을 이용하면 모바일 웹 앱이나 하이브리드 앱 등 스마트폰에서 실행되는 응용프로그램을 개발할 수 있다.
③ X — 틀린 표현은 '디바이스에 접근할 수 없어서'와 그로 인해 '개인정보 보호 및 보안을 철저히 유지할 수 있다'는 부분이다. HTML5 관련 API는 위치 정보 등 디바이스 기능을 활용할 수 있으며, 접근 권한 통제와 보안 정책을 별도로 적용해야 한다.
④ O — audio와 video 요소를 지원하므로 별도의 플러그인 없이 음악과 동영상을 재생할 수 있다.

오답 포인트

③은 디바이스 접근을 보안 위험과 연결한 뒤 '접근할 수 없다'고 단정한 것이 함정이다. HTML5는 디바이스 기능을 활용할 수 있지만, 사용자의 허가와 브라우저의 보안 정책 아래에서 접근한다.

다시 나오면 이렇게 판단

HTML5의 대표 특징을 '양방향 통신, 모바일 활용, 디바이스 API, 플러그인 없는 멀티미디어'로 나누어 판단한다. '디바이스에 접근할 수 없다'처럼 기능 자체를 부정하는 표현은 틀린 설명으로 본다.

NoteBase

암기 포인트

HTML5는 실시간 통신·모바일·디바이스 연동·플러그인 없는 멀티미디어를 지원한다.

NoteBase

문 17. 다음 <보기>의 UML 다이어그램 중 시스템의 구조(structure)보다는 주로 동작(behavior)을 가장 잘 묘사하는 다이어그램들만 고른 것은?

<보기>
㉠ 클래스 다이어그램(Class Diagram)
㉡ 상태 다이어그램(State Diagram)
㉢ 시퀀스 다이어그램(Sequence Diagram)
㉣ 패키지 다이어그램(Package Diagram)
㉤ 배치 다이어그램(Deployment Diagram)

① ㉠, ㉣
② ㉢, ㉣
③ ㉡, ㉤
④ ㉡, ㉢

정답 4번

출제 포인트

이 문제는 UML 다이어그램을 시스템의 정적인 구조를 나타내는 구조 다이어그램과 시간에 따른 변화·상호작용을 나타내는 동작 다이어그램으로 구분하는 문제이다.

핵심 개념

구조 다이어그램: 시스템을 이루는 요소와 그 관계를 정적으로 표현한다. 클래스·패키지·배치 다이어그램이 이에 해당한다.
동작 다이어그램: 시스템이 시간의 흐름에 따라 어떻게 상태를 바꾸거나 객체끼리 상호작용하는지를 표현한다. 상태·시퀀스 다이어그램이 이에 해당한다.
상태 다이어그램은 사건에 따른 객체의 상태 변화와 전이를, 시퀀스 다이어그램은 객체 사이 메시지의 시간 순서를 나타낸다.

NoteBase

선지별 해설

① X — ㉠ 클래스 다이어그램과 ㉣ 패키지 다이어그램은 모두 시스템 요소와 관계를 정적으로 나타내는 구조 다이어그램이다. 틀린 조합은 '㉠, ㉣'이며, 동작 다이어그램만 고르면 '㉡, ㉢'이다.
② X — ㉢ 시퀀스 다이어그램은 동작 다이어그램이지만, ㉣ 패키지 다이어그램은 패키지의 구성과 의존 관계를 나타내는 구조 다이어그램이다. 틀린 항목은 ㉣이며, 이를 ㉡으로 바꾸면 '㉡, ㉢'이 된다.
③ X — ㉡ 상태 다이어그램은 동작 다이어그램이지만, ㉤ 배치 다이어그램은 하드웨어 노드와 소프트웨어 산출물의 물리적 배치를 나타내는 구조 다이어그램이다. 틀린 항목은 ㉤이며, 이를 ㉢으로 바꾸면 '㉡, ㉢'이 된다.
④ O — ㉡ 상태 다이어그램은 상태 변화와 전이를 나타내고, ㉢ 시퀀스 다이어그램은 객체 간 메시지의 시간 순서를 나타내므로 둘 다 시스템의 동작을 묘사한다.

오답 포인트

배치 다이어그램의 '배치'를 실행 동작으로 오해하기 쉽다. 그러나 배치 다이어그램은 실행 환경에서 노드와 산출물이 어디에 놓이는지를 나타내므로 구조 다이어그램이다. 또한 시퀀스 다이어그램은 객체가 등장하더라도 핵심이 객체의 정적 관계가 아니라 메시지의 시간 순서이므로 동작 다이어그램이다.

다시 나오면 이렇게 판단

시간의 흐름, 상태 전이, 메시지 순서가 핵심이면 동작 다이어그램으로 판단한다. 구성 요소, 포함 관계, 물리적 배치가 핵심이면 구조 다이어그램으로 판단한다.

암기 포인트

상태·시퀀스는 동작, 클래스·패키지·배치는 구조이다.

NoteBase

문 18. 다음 중 파일 할당(File Allocation)에 대한 설명으로 가장 옳지 않은 것은?

① 파일에 대한 임의 접근(Random Access)은 연결 할당이 연속할당보다 성능이 우수하다.

② 연결 할당(Linked Allocation)은 파일에 할당된 모든 블록에 다음 블록의 위치를 나타내는 포인터가 포함된다.

③ 블록 단위의 색인 할당(Indexed Allocation)은 파일 내용을 저장하는 블록 외에 별도의 색인 블록(Index Block)이 필요하다.

④ 연속 할당(Contiguous Allocation)은 파일 생성 시 연속된 블록(Block)에 파일을 할당한다.

정답 1번

출제 포인트

연속 할당, 연결 할당, 색인 할당의 저장 구조와 임의 접근 성능 차이를 구별하는 문제이다.

핵심 개념

연속 할당은 파일을 연속된 디스크 블록에 저장하므로 시작 블록과 상대 위치를 이용한 임의 접근이 빠르다. 연결 할당은 각 데이터 블록의 포인터로 다음 블록을 연결하므로 순차 접근에는 적합하지만 임의 접근에는 불리하다. 색인 할당은 별도의 색인 블록에 파일 데이터 블록들의 주소를 모아 두는 방식이다.

선지별 해설

① X — 연결 할당은 원하는 블록에 도달할 때까지 앞 블록의 포인터를 차례로 따라가야 하므로 임의 접근 성능이 연속 할당보다 우수하지 않다. 틀린 표현은 '연결 할당이 연속 할당보다 성능이 우수하다'이며, 올바르게는 '임의 접근에서는 연속 할당이 연결 할당보다 유리하다'이다.
② O — 연결 할당에서는 파일의 각 데이터 블록에 다음 블록의 위치를 가리키는 포인터가 포함되어 블록들이 연결된다.
③ O — 색인 할당에서는 파일 내용을 담는 데이터 블록과 별도로, 그 데이터 블록들의 위치를 기록하는 색인 블록이 필요하다.
④ O — 연속 할당은 파일을 디스크의 연속된 블록들에 배치하는 방식이다.

오답 포인트

①의 함정은 연결 할당이 빈 블록을 유연하게 사용할 수 있다는 장점을 임의 접근 성능의 장점으로 바꾸어 제시한 데 있다. 공간 배치의 유연성과 접근 속도는 서로 다른 판단 기준이다.

다시 나오면 이렇게 판단

원하는 블록으로 바로 이동할 수 있는지를 본다. 연속 할당은 시작 위치에서 상대 위치를 계산할 수 있고, 연결 할당은 포인터 사슬을 순서대로 따라가야 하며, 색인 할당은 색인 블록에서 데이터 블록 주소를 찾는다.

NoteBase

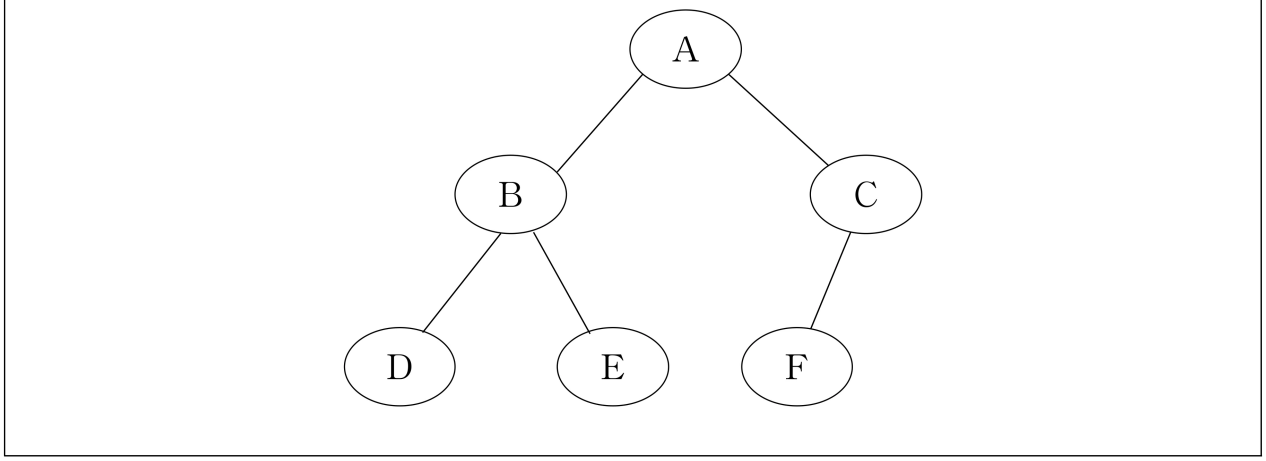
암기 포인트

연속 할당은 빠른 임의 접근, 연결 할당은 포인터 추적, 색인 할당은 별도 색인 블록이다.

NoteBase

문 19. 다음 <보기>의 이진 트리의 내부 경로 길이(Length)와 외부 경로 길이(Length)로 가장 옳은 것은?

<보기>



① 5, 20

② 5, 22

③ 8, 20

④ 8, 22

정답 3번

출제 포인트

이 문제는 루트의 깊이를 0으로 두고 실제 노드의 깊이 합인 내부 경로 길이와 널 링크가 가리키는 외부 노드의 깊이 합인 외부 경로 길이를 구하는 문제이다.

조건 정리

실제 노드는 A, B, C, D, E, F의 6개이다. 깊이는 A가 0, B와 C가 1, D·E·F가 2이다. 외부 노드는 자식이 없는 각 널 링크의 끝에 있다고 본다.

사용할 공식

내부 경로 길이를 I, 외부 경로 길이를 E, 실제 노드 수를 n이라 하면 확장 이진 트리에서 $E = I + 2n$이다. 내부 경로 길이는 실제 노드의 깊이를 모두 더해 직접 구할 수 있고, 이 관계식으로 외부 경로 길이를 계산할 수 있다.

NoteBase

계산 과정

1. 내부 경로 길이는 $I = 0 + 1 + 1 + 2 + 2 + 2 = 8$이다.
2. 실제 노드 수는 $n = 6$이다.
3. 외부 경로 길이는 $E = I + 2n = 8 + 2 \times 6 = 20$이다.

따라서 내부 경로 길이와 외부 경로 길이는 각각 8과 20이다.

선지 판단

① X — 내부 경로 길이를 5로 제시한 부분이 틀렸다. 실제 노드의 깊이 합은 8이므로 올바른 값의 쌍은 8, 20이다.
② X — 내부 경로 길이 5와 외부 경로 길이 22가 모두 틀렸다. 올바르게는 내부 경로 길이 8, 외부 경로 길이 20이다.
③ O — 계산한 내부 경로 길이 8과 외부 경로 길이 20에 모두 일치한다.
④ X — 외부 경로 길이를 22로 제시한 부분이 틀렸다. 내부 경로 길이 8은 맞지만 외부 경로 길이는 $8 + 2 \times 6 = 20$이므로 올바르게는 8, 20이다.

오답 포인트

내부 경로 길이를 간선 수나 루트 이외의 노드 수로 착각하면 5를 고르기 쉽다. 또한 외부 경로 길이는 그림에 표시된 실제 잎 노드 D·E·F의 깊이 합이 아니라, 모든 널 링크 끝에 추가한 외부 노드의 깊이 합이다.

다시 나오면 이렇게 판단

루트 깊이를 0으로 정하고 실제 노드의 깊이를 모두 더해 I를 구한 뒤, 실제 노드 수 n을 세어 $E = I + 2n$을 적용한다.

암기 포인트

확장 이진 트리에서는 외부 경로 길이 = 내부 경로 길이 + 실제 노드 수의 2배이다.

NoteBase

문 20. 다음 중 소프트웨어 테스트에 대한 설명으로 가장 옳지 않은 것은?

① 스트레스 테스트(Stress Test)는 비정상적으로 과도한 분량 또는 빈도로 자원을 요청할 때의 영향을 감사한다.

② 시스템 테스트(System Test)는 모듈들이 통합된 후 넓이 우선 방식 또는 깊이 우선 방식을 사용하여 테스트한다.

③ 단위 테스트(Unit Test)는 개별적인 모듈에 대한 테스트이며 테스트 드라이버(Test Driver)와 테스트 스텁(Test Stub)을 사용할 수 있다.

④ 인수 테스트(Acceptance Test)는 인수 전에 사용자의 요구사항이 만족되었는지 테스트한다.

정답 2번

출제 포인트

소프트웨어 테스트 단계와 기법의 목적을 구분하는 문제이다. 특히 넓이 우선 방식과 깊이 우선 방식이 시스템 테스트가 아니라 통합 테스트의 통합 순서라는 점이 핵심이다.

핵심 개념

단위 테스트는 개별 모듈을 독립적으로 검사하는 테스트이며, 아직 준비되지 않은 호출자나 하위 모듈을 대신하기 위해 테스트 드라이버와 테스트 스텁을 사용할 수 있다.
통합 테스트는 단위 테스트를 마친 모듈을 결합하면서 모듈 사이의 인터페이스와 상호작용을 검사하며, 넓이 우선 방식이나 깊이 우선 방식으로 진행할 수 있다.
시스템 테스트는 통합된 전체 시스템이 명세된 기능 및 비기능 요구사항을 충족하는지 검사한다.
인수 테스트는 최종 사용자의 요구사항을 충족하여 시스템을 인수할 수 있는지 확인한다. 스트레스 테스트는 비정상적으로 큰 부하나 높은 요청 빈도에서 시스템이 어떻게 동작하는지 검사한다.

NoteBase

선지별 해설

① O — 스트레스 테스트는 정상 범위를 넘는 과도한 작업량이나 빈도로 자원을 요청하여 시스템의 반응과 한계를 검사하는 테스트이다. 따라서 비정상적으로 과도한 자원 요청의 영향을 감사한다는 설명은 스트레스 테스트의 취지에 부합한다.
② X — 틀린 부분은 '시스템 테스트는 … 넓이 우선 방식 또는 깊이 우선 방식을 사용하여 테스트한다'이다. 넓이 우선 방식과 깊이 우선 방식은 모듈을 결합하는 통합 테스트의 방식이다. 올바르게는 '통합 테스트는 모듈들을 통합하면서 넓이 우선 방식 또는 깊이 우선 방식을 사용할 수 있다'이다. 시스템 테스트는 통합이 끝난 전체 시스템이 요구사항을 충족하는지 검사한다.
③ O — 단위 테스트는 개별 모듈을 대상으로 한다. 이때 상위 호출 모듈을 대신하는 테스트 드라이버와 하위 피호출 모듈을 대신하는 테스트 스텁을 사용할 수 있다.
④ O — 인수 테스트는 시스템을 인수하기 전에 사용자의 요구사항이 충족되었는지를 확인하는 테스트이므로 옳다.

오답 포인트

②는 '모듈들이 통합된 후'라는 표현 때문에 시스템 테스트처럼 보이지만, 넓이 우선·깊이 우선은 여러 모듈을 어떤 순서로 결합할지를 나타내는 통합 테스트의 용어이다. 테스트 대상이 전체 시스템인지, 모듈 간 결합 과정인지로 구분해야 한다.

다시 나오면 이렇게 판단

개별 모듈이면 단위 테스트, 모듈 간 연결과 결합 순서이면 통합 테스트, 완성된 전체 시스템의 요구사항 검증이면 시스템 테스트, 사용자의 최종 승인 여부이면 인수 테스트로 판단한다. 과도한 부하와 한계 상황은 스트레스 테스트로 판단한다.

암기 포인트

단위는 모듈 하나, 통합은 모듈 사이, 시스템은 전체 요구사항, 인수는 사용자 승인이다. 넓이 우선·깊이 우선은 통합 테스트이다.

NoteBase