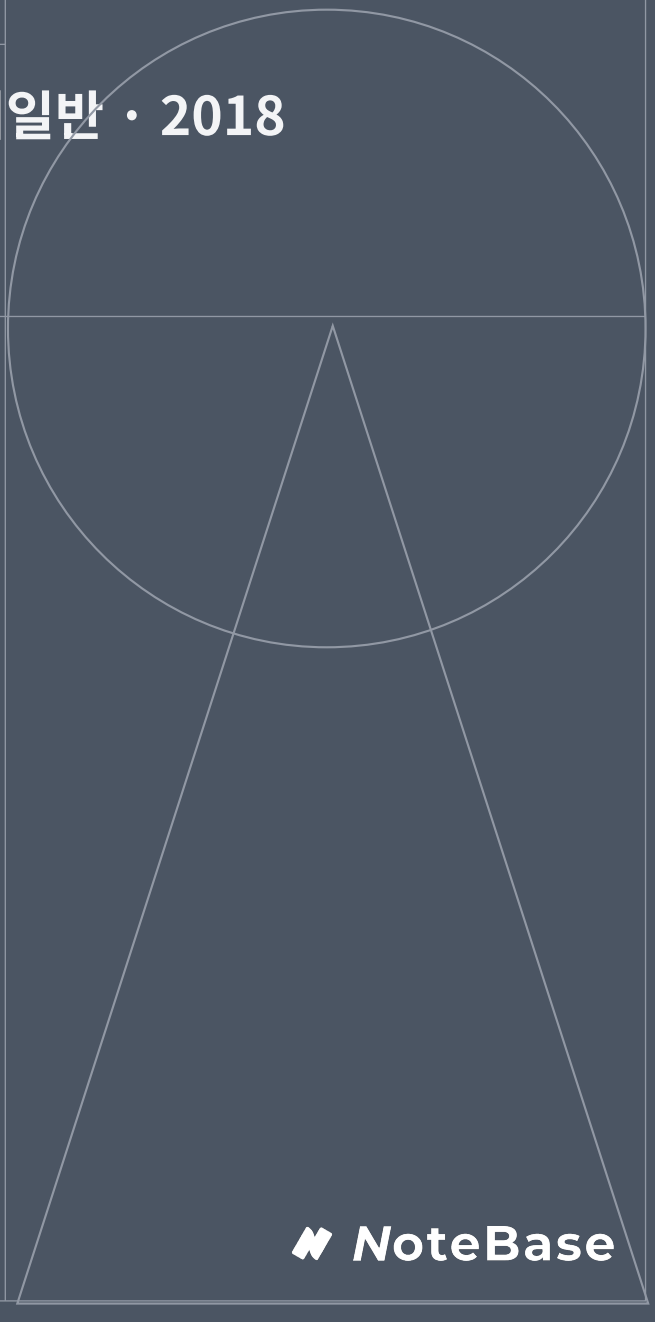
기출OX 해설집

공무원/지방직/9급 · 컴퓨터일반 · 2018



NoteBase

기출OX 해설집

공무원/지방직/9급 · 컴퓨터일반 · 2018

공무원/지방직/9급 · 컴퓨터일반 · 2018

문 1. 선형 자료구조에 해당하지 않는 것은?

① 큐

② 스택

③ 이진 트리

④ 단순 연결 리스트

정답 3번

간단한 해설

이진 트리는 부모와 자식의 관계로 자료가 연결되는 **비선형 자료구조**이다. 자료가 한 줄로 이어지는 구조와 달리, 한 부모에서 두 자식으로 갈라질 수 있다.

공통 해설

선형 자료구조는 자료 사이의 논리적 관계가 **하나의 순서로 이어지는** 구조이다. 예를 들어 A → B → C처럼 각 자료가 앞뒤로 이어진다. 큐, 스택, 단순 연결 리스트는 모두 자료가 이러한 순서를 이루므로 선형 자료구조에 속한다.

이진 트리는 각 노드가 최대 두 개의 자식을 갖는 **부모·자식의 계층 구조**이다. 예를 들어 A의 자식으로 B와 C가 연결되면 A에서 두 갈래로 나뉜다. 이처럼 자료 사이의 관계가 가지를 이루므로 이진 트리는 선형 자료구조에 해당하지 않는다.

NoteBase

문 2. 비트열(bit string) A를 2의 보수로 표현된 부호 있는(signed) 2진 정수로 해석한 값은 -5이다. 비트열 A를 1의 보수로 표현된 부호 있는 2진 정수로 해석한 값은?

① -4

② -5

③ -6

④ -7

정답 1번

간단한 해설

음수의 절댓값을 구할 때 2의 보수는 비트 반전 후 1을 더하지만, 1의 보수는 **비트 반전만** 한다. 따라서 같은 비트열을 1의 보수로 해석한 값은 -5 + 1 = -4이다.

계산 과정

2의 보수 해석: A의 값이 -5이므로 최상위 부호 비트는 1이다. 음수의 2의 보수 표현에서는 모든 비트를 반전한 뒤 1을 더하면 절댓값 5를 얻는다. 따라서 **A의 모든 비트를 반전한 값**을 부호 없는 정수로 읽으면 5 - 1 = 4이다.

1의 보수 해석: 같은 A는 부호 비트가 1이므로 1의 보수에서도 음수이다. 1의 보수 표현에서는 모든 비트를 반전한 값이 곧 절댓값이며, **1을 더하지 않는다**. 앞에서 반전한 값이 4였으므로 A가 나타내는 정수는 -4이다.

따라서 1의 보수로 해석한 값 -4는 ①에 해당한다.

NoteBase

문 3. 직원 테이블 emp의 모든 레코드를 근무연수 wyear에 대해서는 내림차순으로, 동일 근무연수에 대해서는 나이 age의 오름차순으로 정렬한 결과를 얻기 위한 SQL 질의문은?

① SELECT * FROM emp ORDER BY age, wyear DESC;

② SELECT * FROM emp ORDER BY age ASC, wyear;

③ SELECT * FROM emp ORDER BY wyear DESC, age;

④ SELECT * FROM emp ORDER BY wyear, age ASC;

정답 3번

간단한 해설

ORDER BY는 **왼쪽 열을 우선 기준**으로 정렬하고, 그 값이 같을 때 다음 열을 적용한다. 근무연수에는 DESC를 붙이고, 나이는 **방향을 생략하면 오름차순**이므로 wyear DESC, age로 쓴다.

공통 해설

ORDER BY에 여러 열을 나열하면 왼쪽 열부터 우선순위를 갖는다. 첫 번째 열의 값이 같은 레코드끼리만 두 번째 열로 순서를 정한다. 정렬 방향은 **각 열에 따로 적용**하며, ASC는 오름차순, DESC는 내림차순이다. 방향을 생략한 열은 기본적으로 ASC가 적용된다.

이 문제에서는 근무연수 wyear가 우선 기준이므로 wyear를 먼저 쓰고 DESC를 붙인다. 근무연수가 다르면 나이와 관계없이 근무연수가 큰 직원이 먼저 오고, **근무연수가 같을 때만 나이가 적은 직원**이 먼저 온다. 따라서 두 번째 기준은 age ASC이며, ASC를 생략한 age도 같은 뜻이다. 요구한 질의문은 SELECT * FROM emp ORDER BY wyear DESC, age;이다.

NoteBase

문 4. 다음에서 설명하는 디스크 스케줄링은?

디스크 헤드가 한쪽 방향으로 트랙의 끝까지 이동하면서 만나는 요청을 모두 처리한다. 트랙의 끝에 도달하면 반대 방향으로 이동하면서 만나는 요청을 모두 처리한다. 이러한 방식으로 헤드가 디스크 양쪽을 계속 왕복하면서 남은 요청을 처리한다.

① 선입 선처리(FCFS) 스케줄링

② 최소 탐색 시간 우선(SSTF) 스케줄링

③ 스캔(SCAN) 스케줄링

④ 라운드 로빈(RR) 스케줄링

정답 3번

간단한 해설

헤드가 한 방향의 요청을 처리한 뒤 **디스크의 끝에서 방향을 바꾸어 왕복**하는 것은 스캔(SCAN) 스케줄링의 특징이다.

공통 해설

스캔(SCAN)은 디스크 헤드가 한쪽 방향으로 이동하면서 경로에 있는 요청을 트랙 위치 순서대로 처리하고, 디스크의 끝에 도달하면 방향을 바꾸어 반대쪽 요청을 처리하는 방식이다. 요청 하나를 처리할 때마다 이동 방향을 새로 정하지 않고 **끝에 도달할 때까지 진행 방향을 유지**한다. 따라서 현재 진행 방향의 뒤쪽에 있는 요청은 헤드가 되돌아올 때 처리된다. 지문의 '트랙의 끝에서 방향을 바꾸며 디스크 양쪽을 계속 왕복한다'는 동작은 이 방식에 해당한다.

NoteBase

문 5. 정보량의 크기가 작은 것에서 큰 순서대로 바르게 나열한 것은? (단, PB, TB, ZB, EB는 각각 petabyte, terabyte, zettabyte, exabyte이다)

① 1 PB, 1 TB, 1 ZB, 1 EB

② 1 PB, 1 TB, 1 EB, 1 ZB

③ 1 TB, 1 PB, 1 ZB, 1 EB

④ 1 TB, 1 PB, 1 EB, 1 ZB

정답 4번

간단한 해설

테라 이후 접두어가 한 단계 커질 때마다 정보량이 **1,000배씩 증가**한다. 엑사바이트는 제타바이트보다 작다.

올바른 순서

1 TB → 1 PB → 1 EB → 1 ZB

1 TB(테라바이트): 테라(T)는 10^{12}배를 뜻하므로 $1\text{TB} = 10^{12}\text{B}$이다. 제시된 네 단위 중 바이트 수가 가장 적다.

1 PB(페타바이트): 페타(P)는 10^{15}배를 뜻하므로 $1\text{PB} = 10^{15}\text{B}$이다. **1 PB는 1 TB의 1,000배**이므로 테라바이트 다음에 놓인다.

1 EB(엑사바이트): 엑사(E)는 10^{18}배를 뜻하므로 $1\text{EB} = 10^{18}\text{B}$이다. 1 EB는 1 PB의 1,000배이므로 페타바이트보다 크다.

1 ZB(제타바이트): 제타(Z)는 10^{21}배를 뜻하므로 $1\text{ZB} = 10^{21}\text{B}$이다. **1 ZB는 1 EB의 1,000배**이므로 네 단위 중 가장 크다.

NoteBase

문 6. 다음에서 설명하는 RAID 레벨은?

◯ 블록 단위 스트라이핑(striping)을 통해 데이터를 여러 디스크에 분산 저장한다.
◯ 패리티를 패리티 전용 디스크에 저장한다.

① RAID 레벨 1
② RAID 레벨 2
③ RAID 레벨 4
④ RAID 레벨 5

정답 3번

간단한 해설

RAID 레벨 4는 데이터를 **블록 단위로 분산 저장**하고, 복구에 필요한 **패리티를 전용 디스크에 저장**한다. 두 조건을 함께 확인하면 RAID 레벨을 구분할 수 있다.

공통 해설

RAID 레벨 4는 **블록 단위 스트라이핑과 전용 패리티 디스크**를 사용하는 방식이다. 스트라이핑은 데이터를 일정한 크기의 블록으로 나누어 여러 데이터 디스크에 분산 저장하는 것이고, 패리티는 데이터로부터 계산한 복구용 정보이다. 데이터 디스크 하나에 장애가 발생하면 남아 있는 데이터와 패리티를 이용해 손실된 데이터를 복원할 수 있다.

예를 들어 데이터 블록 A와 B를 서로 다른 데이터 디스크에 저장하고, 이들로부터 계산한 패리티 P를 별도의 전용 디스크에 저장한다. 다음 블록 묶음에서도 패리티는 같은 전용 디스크에 저장한다. 지문의 두 설명은 이처럼 데이터를 블록으로 나누어 배치하면서 패리티를 한 전용 디스크에 모으는 RAID 레벨 4의 저장 구조를 나타낸다.

NoteBase

문 7. 소프트웨어 개발을 위한 애자일 기법에 대한 설명으로 옳은 것은?

① 소프트웨어를 점증적으로 개발한다.

② 작동하는 소프트웨어보다 포괄적인 문서에 더 가치를 둔다.

③ 계획에 따라 단계적으로 개발하므로 변화에 대응하기 어렵다.

④ 고객과의 협업보다 계약 협상을 더 중요시한다.

정답 1번

간단한 해설

애자일은 짧은 개발 주기마다 작동하는 기능을 완성하고, 고객의 피드백을 반영하여 소프트웨어를 **점증적으로 확장**한다.

선지별 해설

① O: 애자일은 개발을 짧은 주기로 반복하면서 **작동하는 기능을 조금씩 추가**한다. 예를 들어 먼저 상품 조회 기능을 완성하고, 다음 주기에 장바구니 기능을 추가하는 방식으로 소프트웨어를 점증적으로 개발한다.

② X: 애자일은 포괄적인 문서보다 **작동하는 소프트웨어에 더 높은 가치**를 둔다. 문서 작성 자체를 부정하는 것은 아니며, 문서의 양보다 실제로 사용할 수 있는 결과물을 중시한다.

③ X: 애자일은 고정된 계획을 그대로 따르기보다 **변화에 대응하는 것**을 중시한다. 개발 중 요구사항이 달라지면 고객의 피드백을 반영하여 계획과 개발 우선순위를 조정한다.

④ X: 애자일은 계약 협상보다 **고객과의 협업**을 중시한다. 계약 조건을 정하는 데 그치지 않고 개발 과정에서 고객과 지속적으로 소통하여 필요한 기능과 개선 사항을 확인한다.

NoteBase

문 8. 2-way 집합 연관 사상(set-associative mapping) 방식을 사용하는 캐시 기억장치를 가진 컴퓨터가 있다. 캐시 기억장치 접근(access)을 위해 주기억장치 주소가 다음 세 필드(field)로 구분된다면, 캐시 기억장치의 총 라인(line) 개수는?

태그(tag) 필드	세트(set) 필드	오프셋(offset) 필드
8비트	9비트	7비트

① 128개

② 256개

③ 512개

④ 1,024개

정답 4번

간단한 해설

세트 필드가 9비트이므로 세트는 $2^9 = 512$개이다. 2-way는 **세트마다 라인이 2개**라는 뜻이므로, 총 라인 수는 $512 \times 2 = 1{,}024$개이다.

계산 과정

2-way의 의미: 집합 연관 사상은 캐시 라인을 여러 세트로 나누는 방식이다. 2-way에서는 **한 세트가 라인 2개**로 구성되므로, 총 라인 수는 세트 수에 2를 곱하여 구한다.

세트 수: 그림의 세트 필드는 9비트이다. 9비트로 구분할 수 있는 세트는 $2^9 = 512$개이며, 이 값은 **라인 수가 아니라 세트 수**이다. 태그 필드 8비트는 해당 세트에 저장된 주기억장치 블록을 식별하고, 오프셋 필드 7비트는 블록 내부의 위치를 지정하므로 세트 수 계산에는 사용하지 않는다.

총 라인 수: 세트가 512개이고 각 세트에 라인이 2개 있으므로 $512 \times 2 = 1{,}024$개이다. 이는 ④의 1,024개에 해당한다.

NoteBase

문 9. 다음 C 프로그램의 출력 결과는?

```
#include <stdio.h>
#define SIZE 3
void func(int *m, int *a, int b);

int main(void){
        int num[SIZE] = { 1, 3, 6 };
        int a=10, b=30;

        func(num, &a, b);
        printf("a = %d, b = %d\n", a, b);

        return 0;
}
void func(int *m, int *x, int y){
        int i = 0, n = 0;
        y = *x;
        n = *(m + 1) + (*m + 2);
        *x = ++n;
}
```

① $a = 7,\ b = 10$

② $a = 7,\ b = 30$

③ $a = 10,\ b = 10$

④ $a = 10,\ b = 30$

정답 2번

간단한 해설

(*m + 2)는 세 번째 원소가 아니라 **첫 번째 원소의 값에 2를 더한 값**이므로 n은 6이다. ++n으로 증가한 7이 포인터를 통해 a에 저장되며, **값을 복사해 전달한 b는 30을 유지**한다.

NoteBase

계산 과정

함수 호출: main에서 num은 {1, 3, 6}, a는 10, b는 30으로 초기화된다. func(num, &a, b)를 호출하면 m은 num의 첫 번째 원소를 가리키고, x는 main의 a를 가리킨다. y에는 b의 값 30이 복사된다. 따라서 ***x에 대입하면 main의 a가 바뀌지만**, y에 대입하면 func의 지역 변수 y만 바뀐다.

y = *x: x가 가리키는 a의 현재 값은 10이므로 y가 30에서 10으로 바뀐다. y와 main의 b는 별개의 변수이므로 **main의 b는 여전히 30**이다.

n 계산: *(m + 1)은 m에서 int 원소 하나만큼 이동한 위치의 값이므로 num[1], 즉 3이다. 반면 (*m + 2)는 먼저 *m으로 num[0]의 값 1을 읽고 2를 더하므로 3이다. 즉, **(*m + 2)는 *(m + 2)와 다르다**. 세 번째 원소 num[2]의 값 6을 읽는 연산이 아니므로 n은 3 + (1 + 2) = 6이 된다.

*x = ++n과 출력: **전위 증가 연산자 ++n은 n을 먼저 1 증가시킨 값**을 사용하므로 n은 6에서 7이 되고, 그 값이 *x에 대입된다. x는 main의 a를 가리키므로 a는 7로 바뀐다. 함수가 끝난 뒤 printf가 출력하는 값은 a = 7, b = 30이며, ②의 값과 일치한다.

NoteBase

문 10. TCP 프로토콜에 대한 설명으로 옳지 않은 것은?

① 전이중(full duplex) 연결 서비스를 제공한다.

② 3-way 핸드셰이크(handshake)를 사용하여 연결을 설정한다.

③ 흐름제어(flow control)와 혼잡제어(congestion control)를 제공한다.

④ TCP 세그먼트(segment)에서 검사합(checksum)의 포함은 선택 사항이다.

정답 4번

간단한 해설

TCP 검사합은 전송 중 오류를 검출하기 위한 **필수 항목**이다. 송신 측은 검사합을 계산해 넣고 수신 측은 이를 검사하므로, ④의 '선택 사항'이라는 설명은 잘못되었다.

선지별 해설

① O: 전이중은 **양쪽이 동시에 데이터를 송수신**할 수 있는 통신 방식이다. TCP 연결에서는 A가 B에게 데이터를 보내는 동안 B도 A에게 데이터를 보낼 수 있다.

② O: TCP는 일반적인 연결 설정에서 **SYN → SYN+ACK → ACK**의 3-way 핸드셰이크를 사용한다. 연결을 요청하는 측이 SYN을 보내면 상대가 SYN+ACK로 응답하고, 요청 측이 ACK를 보내 양쪽의 초기 순서번호를 서로 확인한다.

③ O: 흐름제어는 **수신 측의 처리 능력**을 고려해 송신량을 조절하여 수신 버퍼가 넘치지 않도록 한다. 혼잡제어는 **네트워크의 혼잡 상태**를 고려해 송신량을 조절한다. TCP는 두 기능을 모두 제공한다.

④ X: TCP 헤더의 검사합은 **선택 사항이 아니라 필수 항목**이다. 송신 측은 검사합을 계산하여 세그먼트에 포함하고, 수신 측은 이를 검사해 전송 중 오류가 발생했는지 확인한다. 따라서 '검사합의 포함은 선택 사항이다'를 '검사합을 반드시 포함한다'로 고쳐야 한다.

NoteBase

문 11. 사용자가 인터넷 등을 통해 하드웨어, 소프트웨어 등의 컴퓨팅 자원을 원격으로 필요한 만큼 빌려서 사용하는 방식의 서비스 기술은?

① 클라우드 컴퓨팅

② 유비쿼터스 센서 네트워크

③ 웨어러블 컴퓨터

④ 소셜 네트워크

정답 1번

간단한 해설

하드웨어·소프트웨어 등의 자원을 직접 갖추는 대신, 인터넷을 통해 **필요한 만큼 원격으로 제공받아 사용하는** 방식이 클라우드 컴퓨팅이다.

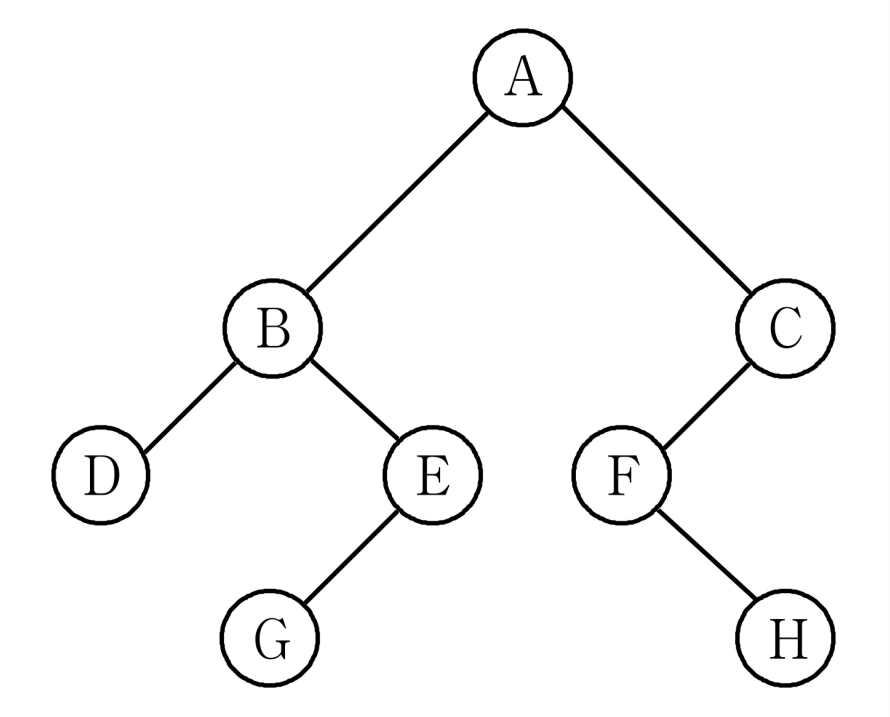
공통 해설

클라우드 컴퓨팅은 서버, 저장 공간, 소프트웨어 등의 컴퓨팅 자원을 네트워크를 통해 서비스로 제공하는 방식이다. 사용자는 서비스 제공자가 관리하는 자원에 원격으로 접근하며, **필요에 따라 자원을 할당받거나 사용 규모를 조절**할 수 있다.

예를 들어 웹사이트 운영자가 서버를 직접 구매하는 대신 인터넷으로 서버 자원을 빌리고, 접속자가 늘면 이용 자원을 늘리는 방식이다. 문제의 '컴퓨팅 자원을 원격으로 필요한 만큼 빌려서 사용'한다는 조건은 이러한 클라우드 컴퓨팅의 자원 제공 방식을 가리킨다.

NoteBase

문 12. 문 12. 다음 이진 트리의 노드를 전위 순회(preorder traversal)할 경우의 방문 순서는?



① A – B – C – D – E – F – G – H

② A – B – D – E – G – C – F – H

③ D – B – G – E – A – F – H – C

④ D – G – E – B – H – F – C – A

정답 2번

간단한 해설

전위 순회는 **루트 → 왼쪽 부분 트리 → 오른쪽 부분 트리** 순서로 방문한다. 따라서 B 아래의 D·E·G를 모두 방문한 뒤에 A의 오른쪽 자식 C로 이동한다.

NoteBase

올바른 순서

A → B → D → E → G → C → F → H

A: 전위 순회는 **루트를 먼저 방문**하고 왼쪽 부분 트리, 오른쪽 부분 트리 순서로 진행한다. 각 부분 트리에도 같은 규칙을 적용하므로 전체 트리의 루트 A부터 방문한다.

B: A를 방문한 뒤 왼쪽 부분 트리로 들어간다. 이 부분 트리의 루트인 B를 자식 D와 E보다 먼저 방문한다.

D: B의 왼쪽 자식 D를 방문한다. D는 자식이 없는 잎 노드이므로 여기서 더 내려가지 않고 B 쪽으로 돌아간다.

E: B의 왼쪽 부분 트리 방문을 마쳤으므로 오른쪽 자식 E를 방문한다. E 역시 자신의 자식 G보다 먼저 방문한다.

G: E의 왼쪽 자식 G를 방문한다. G는 잎 노드이고 E에는 오른쪽 자식이 없으므로, 이로써 **A의 왼쪽 부분 트리 방문을 모두 마친다**.

C: A의 왼쪽 부분 트리 방문을 마친 뒤 오른쪽 부분 트리의 루트 C를 방문한다. 같은 깊이에 있는 노드를 차례로 방문하는 방식이 아니므로 C는 B 직후가 아니라 G 다음에 방문한다.

F: C의 왼쪽 자식 F를 방문한다. F는 자신의 오른쪽 자식 H보다 먼저 방문하며, 왼쪽 자식은 없으므로 다음에는 오른쪽으로 이동한다.

H: F의 오른쪽 자식 H를 방문한다. H는 잎 노드이고 C에도 오른쪽 자식이 없으므로 전체 순회를 마친다.

NoteBase

문 13. 전자메일의 송신 또는 수신을 목적으로 하는 응용 계층 프로토콜에 해당하지 않는 것은?

① IMAP

② POP3

③ SMTP

④ SNMP

정답 4번

간단한 해설

SNMP는 라우터 등 네트워크 장비의 상태와 관리 정보를 주고받는 **네트워크 관리용 프로토콜**이다. 전자메일을 보내거나 받아 읽기 위한 프로토콜에는 해당하지 않는다.

공통 해설

전자메일용 프로토콜은 **메일 메시지의 전달 또는 메일함 접근**을 담당한다. SMTP는 사용자가 메일을 보낼 때와 메일 서버 사이에서 메시지를 전달할 때 사용한다. POP3와 IMAP은 사용자가 메일 서버에 도착한 메시지를 받아 읽는 데 사용하므로, 세 프로토콜 모두 전자메일의 송신 또는 수신과 관련된다.

SNMP는 라우터 등 네트워크 장비의 상태를 조회하고 관리 정보를 교환하는 프로토콜이다. SNMP도 응용 계층에 속하지만, 다루는 대상이 메일 메시지가 아니라 **네트워크 장비의 관리 정보**이므로 전자메일의 송수신을 목적으로 하는 프로토콜에서 제외된다.

NoteBase

문 14. 모바일 기기에 특화된 운영체제에 해당하지 않는 것은?

① iOS
② Android
③ Symbian
④ Solaris

정답 4번

간단한 해설

Solaris는 **서버·워크스테이션용 유닉스 계열 운영체제**로, 모바일 기기에 특화된 운영체제에 해당하지 않는다.

공통 해설

모바일 운영체제는 스마트폰 등 휴대용 기기의 하드웨어 자원과 응용 프로그램을 관리하도록 설계된 운영체제이다. 이 문항의 분류 기준은 **모바일 기기를 주된 사용 대상으로 설계했는가**이다. iOS와 Android는 스마트폰용 운영체제이며, Symbian도 스마트폰에 사용된 모바일 운영체제이다.

Solaris는 **서버와 워크스테이션용**으로 개발된 유닉스 계열 운영체제이다. 따라서 운영체제에는 해당하지만, 모바일 기기에 특화된 운영체제의 범주에는 해당하지 않는다.

NoteBase

문 15. 다음 표는 단일 중앙처리장치에 진입한 프로세스의 도착 시간과 그 프로세스를 처리하는 데 필요한 실행 시간을 나타낸 것이다. 비선점 SJF(Shortest Job First) 스케줄링 알고리즘을 사용한 경우, P1, P2, P3, P4 프로세스 4개의 평균 대기 시간은? (단, 프로세스 간 문맥 교환에 따른 오버헤드는 무시하며, 주어진 4개의 프로세스 외에 처리할 다른 프로세스는 없다고 가정한다)

프로세스	도착 시간(ms)	실행 시간(ms)
P1	0	5
P2	3	6
P3	4	3
P4	6	4

① 3 ms

② 3.5 ms

③ 4 ms

④ 4.5 ms

정답 1번

간단한 해설

비선점 SJF는 **현재 도착해 있는 프로세스 중 실행 시간이 가장 짧은 것**을 선택하고, 실행을 시작하면 끝날 때까지 처리한다. 각 프로세스의 실행 시작 시각에서 도착 시각을 뺀 대기 시간을 평균하면 **3 ms**이다.

NoteBase

계산 과정

비선점 SJF에서는 CPU가 비었을 때 도착해 있는 프로세스들의 실행 시간을 비교한다. **아직 도착하지 않은 프로세스는 선택 대상이 아니며**, 실행 중인 프로세스는 더 짧은 프로세스가 도착해도 중단하지 않는다. 표의 도착 시간과 실행 시간은 모두 ms 단위이고, 문맥 교환 오버헤드는 없으므로 실행 구간 사이에 추가 시간을 넣지 않는다.

P1: 0 ms에 도착한 프로세스는 P1뿐이므로 바로 실행한다. 실행 시간이 5 ms이므로 0~5 ms에 처리한다. P2는 3 ms, P3는 4 ms에 도착하지만 **P1의 실행을 중단시키지 못한다**. P1의 대기 시간은 0 − 0 = 0 ms이다.

P3: 5 ms에는 P2와 P3가 대기하고 있다. 실행 시간이 각각 6 ms, 3 ms이므로 P3를 선택하여 5~8 ms에 처리한다. P4는 6 ms에 도착하므로 5 ms의 선택 대상에 포함되지 않는다. P3의 대기 시간은 5 − 4 = 1 ms이다.

P4: 8 ms에는 P2와 P4가 대기하고 있다. 실행 시간이 각각 6 ms, 4 ms이므로 P4를 선택하여 8~12 ms에 처리한다. P4의 대기 시간은 8 − 6 = 2 ms이다.

P2: 12 ms에는 남은 P2를 선택하여 12~18 ms에 처리한다. P2는 3 ms에 도착하여 12 ms에 실행을 시작하므로 대기 시간은 12 − 3 = 9 ms이다.

평균 대기 시간: 이 문제에서는 각 프로세스가 한 번에 끝까지 실행되므로, 대기 시간은 **실행 시작 시각 − 도착 시각**이다. P1, P2, P3, P4의 대기 시간 합은 0 + 9 + 1 + 2 = 12 ms이다. 이를 프로세스 수 4로 나누면 $\frac{12}{4} = 3$ ms이며, ①의 3 ms에 해당한다.

NoteBase

문 16. IPv4와 IPv6에 대한 설명으로 옳지 않은 것은?

① IPv4는 비연결형 프로토콜이다.

② IPv6 주소의 비트 수는 IPv4 주소 비트 수의 2배이다.

③ IPv6는 애니캐스트(anycast) 주소를 지원한다.

④ IPv6는 IPv4 네트워크와의 호환성을 위한 방법을 제공한다.

정답 2번

간단한 해설

IPv4 주소는 32비트이고 IPv6 주소는 128비트이므로, IPv6 주소의 비트 수는 IPv4의 **4배**이다. ②의 '2배'는 '4배'로 고쳐야 한다.

선지별 해설

① O: IPv4는 데이터를 보내기 전에 **연결을 설정하지 않는** 비연결형 프로토콜이다. 각 패킷을 독립적으로 처리하며, IPv4 자체가 패킷의 도착이나 도착 순서를 보장하지는 않는다.

② X: IPv4 주소는 32비트, IPv6 주소는 128비트이다. 32 × 4 = 128이므로 주소의 비트 수는 2배가 아니라 **4배**이다.

③ O: IPv6는 애니캐스트 주소를 지원한다. 여러 인터페이스에 같은 애니캐스트 주소를 할당하면, 그 주소로 보낸 패킷은 **라우팅 기준으로 가장 가까운 하나**에 전달된다. 여기서 가까움은 지리적 거리가 아니라 라우팅 프로토콜이 정하는 기준에 따른다.

④ O: IPv4와 IPv6의 공존과 전환을 위해 **듀얼 스택과 터널링** 등의 방법을 제공한다. 듀얼 스택은 한 장치에서 IPv4와 IPv6를 함께 지원하는 방식이고, 터널링은 IPv6 패킷을 IPv4 패킷 안에 넣어 IPv4 네트워크를 통과시키는 방식이다. 두 프로토콜의 패킷 형식이 그대로 호환되는 것은 아니므로 이러한 방법이 필요하다.

NoteBase

문 17. 순차논리회로(sequential logic circuit)에 해당하는 것은?

① 3-to-8 디코더(decoder)

② 전가산기(full adder)

③ 동기식 카운터(synchronous counter)

④ 4-to-1 멀티플렉서(multiplexer)

정답 3번

간단한 해설

동기식 카운터는 플립플롭에 **현재 계수 상태를 저장**하고, 클록이 들어오면 그 상태를 바탕으로 다음 상태로 바뀐다. 이처럼 기억된 상태가 동작에 관여하므로 순차논리회로에 해당한다.

공통 해설

순차논리회로는 **현재 입력뿐 아니라 저장된 상태**도 동작에 관여하는 회로이다. 반면 조합논리회로는 상태를 기억하지 않고 현재 입력의 조합만으로 출력이 결정된다.

동기식 카운터는 플립플롭에 현재 계수값을 저장하며, 각 플립플롭이 같은 클록에 맞추어 다음 상태로 전환된다. 예를 들어 증가하도록 동작하는 카운터는 현재 값이 2이면 다음 계수 클록에서 3이 된다. 같은 계수 클록이 들어와도 이전에 저장된 값에 따라 다음 값이 달라지므로 순차논리회로이다.

3-to-8 디코더, 전가산기, 4-to-1 멀티플렉서는 모두 **현재 입력만으로 출력이 정해지는 조합논리회로**이다. 전가산기에 들어오는 자리올림도 현재 입력의 일부이며, 전가산기 자체가 이전 계산 결과를 기억한다는 뜻은 아니다.

NoteBase

문 18. 클록(clock) 주파수가 2 GHz인 중앙처리장치를 사용하는 컴퓨터 A에서 프로그램 P를 실행하는 데 10초가 소요된다. 클록 주파수가 더 높은 중앙처리장치를 사용하는 컴퓨터 B에서 프로그램 P를 실행하면, 소요되는 클록 사이클 수는 컴퓨터 A에 대비하여 1.5배로 증가하나 실행 시간은 6초로 감소한다. 컴퓨터 B에 사용된 중앙처리장치의 클록 주파수는? (단, 실행 시간은 중앙처리장치의 실행 시간만을 고려한 것이며 프로그램 P만 실행하여 측정된다)

① 3 GHz

② 4 GHz

③ 5 GHz

④ 6 GHz

정답 3번

간단한 해설

클록 주파수는 **전체 클록 사이클 수를 실행 시간으로 나눈 값**이다. B는 A보다 사이클 수가 1.5배이므로, 주파수는 $2 \cdot 1.5 \cdot \frac{10}{6} = 5$ GHz이다.

계산 과정

클록 주파수는 1초 동안 발생하는 클록 사이클 수이다. 전체 사이클 수를 N, 주파수를 f, 중앙처리장치의 실행 시간을 t라 하면 $N = f \cdot t$이므로 $f = \frac{N}{t}$로 구한다.

A의 전체 사이클 수: 1 GHz는 1초당 10억 사이클을 뜻한다. A는 2 GHz이므로 1초당 20억 사이클이 발생하며, 10초 동안 프로그램 P를 실행하는 데 사용한 전체 사이클 수는 20억 × 10 = 200억 회이다.

B의 전체 사이클 수: **A에서 사용한 사이클 수의 1.5배**가 필요하므로, 200억 × 1.5 = 300억 회이다. 같은 프로그램을 실행하더라도 이 문제에서는 두 컴퓨터의 전체 사이클 수가 다르다는 조건을 반영해야 한다.

B의 클록 주파수: 300억 사이클을 6초 동안 처리하므로 $\frac{300}{6} = 50$억 사이클이 1초마다 발생한다. 1초당 50억 사이클은 50억 Hz, 즉 5 GHz이다.

계산한 B의 클록 주파수 5 GHz는 ③의 값과 일치한다.

NoteBase

문 19. 다음 Java 프로그램의 출력 결과는?

```
public class Foo {
    public static void main(String[] args) {
        int i, j, k;
        for (i = 1, j = 1, k = 0; i < 5; i++ ) {
            if ((i % 2) == 0)
                continue;
            k += i * j++;
        }
        System.out.println(k);
    }
}
```

① 5
② 7
③ 11
④ 15

정답 2번

간단한 해설

짝수 i에서는 **continue로 누적 연산과 j++를 모두 건너뛴다**. 따라서 i = 1, 3일 때 각각 1 × 1과 3 × 2가 더해져 k는 **7**이 된다.

NoteBase

계산 과정

초기값과 연산: i = 1, j = 1, k = 0으로 시작하며, i < 5인 동안 반복한다. k += i * j++는 i와 j를 곱한 값을 k에 더하는 연산이다. 이때 **j++는 증가 전 값을 곱셈에 사용**하고 j를 1 증가시킨다. i % 2는 i를 2로 나눈 나머지이므로, 그 값이 0이면 i는 짝수이다.

i = 1: 홀수이므로 누적 연산을 실행한다. k = 0 + 1 × 1 = 1이 되고, j는 1에서 2로 증가한다. 반복문의 증감식 i++에 따라 i는 2가 된다.

i = 2: 짝수이므로 continue를 실행한다. **이번 반복의 나머지 본문을 건너뛰고 i++로 이동**하므로 k는 1, j는 2로 유지되고 i는 3이 된다.

i = 3: 홀수이므로 현재 j 값인 2를 사용하여 k = 1 + 3 × 2 = 7을 계산한다. 이어 j는 3이 되고, i++에 따라 i는 4가 된다.

i = 4: 짝수이므로 누적 연산을 건너뛴다. k는 7, j는 3으로 유지되고 i++에 따라 i는 5가 된다.

반복 종료와 출력: i = 5이면 i < 5가 거짓이 되어 반복문이 끝난다. System.out.println(k)는 최종 k 값인 7을 출력하며, 이는 ②의 값과 일치한다.

NoteBase

문 20. 다음 카르노 맵(Karnaugh map)으로 표현된 부울 함수 F(A, B, C, D)를 곱의 합(sum of products) 형태로 최소화(minimization)한 결과는? (단, X는 무관(don’t care) 조건을 나타낸다)

CD \ AB	00	01	11	10
00	0	1	X	1
01	0	X	0	0
11	X	1	0	0
10	0	1	X	1

① $F(A, B, C, D) = AD' + BC'D' + A'BC$

② $F(A, B, C, D) = AB'D' + BC'D' + A'BC$

③ $F(A, B, C, D) = A'B + AD'$

④ $F(A, B, C, D) = A'C + AD'$

정답 3번

간단한 해설

무관 조건 X를 활용하면 **AB = 01인 열 전체**와 **위·아래 경계를 연결한 오른쪽 네 칸**을 각각 묶을 수 있다. 각 묶음에서 변하지 않는 변수만 남기면 $A'B$와 AD'가 되어 최소 곱의 합은 $F = A'B + AD'$이다.

NoteBase

계산 과정

묶음의 기준: 가로축은 AB, 세로축은 CD이며, 두 축 모두 00 → 01 → 11 → 10 순서이다. 곱의 합을 구할 때는 1인 칸을 모두 덮도록 인접한 칸을 2^n개씩 직사각형으로 묶는다. **0인 칸은 포함할 수 없고**, X인 칸은 묶음을 크게 만드는 데 필요할 때 포함한다. 맵의 위·아래 끝과 왼쪽·오른쪽 끝도 서로 인접한다. 묶음 안에서 값이 바뀌는 변수는 제거하고, 항상 1인 변수는 그대로, 항상 0인 변수는 보수로 남긴다.

AB = 01인 열: 위에서부터 1, X, 1, 1이므로 열 전체의 네 칸을 묶는다. 이 안에서 CD는 00, 01, 11, 10으로 모두 변하지만 **A = 0, B = 1은 일정**하다. 따라서 C와 D를 제거하면 곱항은 $A'B$이다. 여기서 작은따옴표는 보수, 변수의 곱은 AND를 뜻한다.

AB = 11 · 10인 열과 CD = 00 · 10인 행: 맨 위와 맨 아래 행은 경계를 사이에 두고 인접하므로, 해당 네 칸 X, 1, X, 1을 하나로 묶는다. B와 C는 변하지만 **A = 1, D = 0은 일정**하므로 곱항은 AD'이다.

최소화 결과: 두 묶음이 1인 다섯 칸을 모두 덮으며, 0인 칸은 포함하지 않는다. 0 없이 만들 수 있는 8칸 묶음은 없으므로 한 곱항을 변수 하나로 줄일 수 없고, 하나의 4칸 묶음으로는 1인 다섯 칸을 모두 덮을 수 없다. 따라서 두 변수로 된 곱항 두 개를 OR로 연결한 $F(A, B, C, D) = A'B + AD'$가 최소 곱의 합이며, ③의 식과 일치한다.

NoteBase