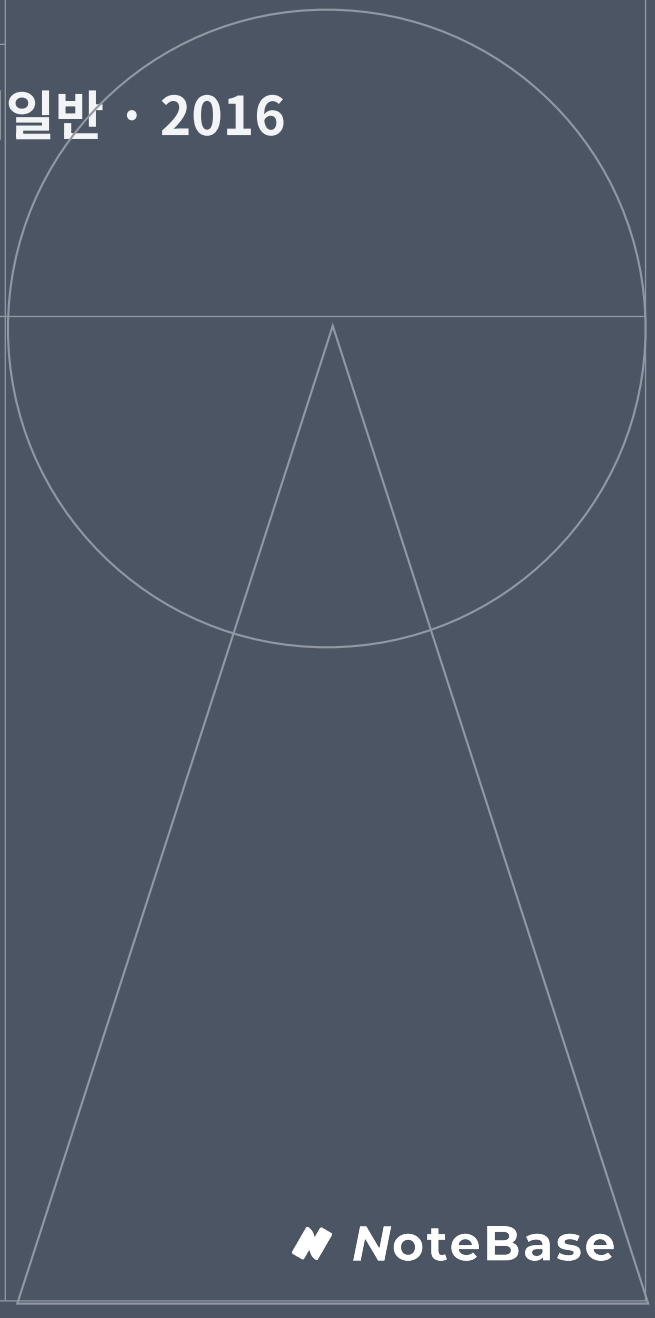
기출OX 해설집

공무원/지방직/9급 · 컴퓨터일반 · 2016



NoteBase

기출OX 해설집

공무원/지방직/9급 · 컴퓨터일반 · 2016

공무원/지방직/9급 · 컴퓨터일반 · 2016

문 1. 접근 속도가 가장 빠른 기억장치는?

① 주기억장치

② 보조기억장치

③ 레지스터

④ 캐시

정답 3번

간단한 해설

레지스터는 CPU 내부에서 연산에 필요한 데이터와 중간 결과를 저장하며, **CPU가 직접 빠르게 접근**할 수 있다. 일반적인 기억장치 계층에서 캐시보다 접근 속도가 빠르다.

선지별 해설

① 주기억장치: 실행 중인 프로그램과 데이터를 저장하는 RAM이 대표적이다. 보조기억장치보다 빠르지만, **레지스터와 캐시보다 접근 속도가 느리다**.

② 보조기억장치: SSD나 HDD처럼 프로그램과 데이터를 장기간 저장하는 장치이다. CPU가 연산에 사용할 데이터는 일반적으로 주기억장치로 옮겨 사용하며, 제시된 기억장치 중 **접근 속도가 가장 느리다**.

③ 레지스터: CPU 내부에서 연산할 데이터나 연산 결과 등을 일시적으로 저장한다. 연산장치가 사용할 값을 직접 제공하므로 **제시된 기억장치 중 접근 속도가 가장 빠르다**.

④ 캐시: 자주 사용하는 데이터나 명령어를 저장하여 CPU가 주기억장치에 접근하며 기다리는 시간을 줄인다. **주기억장치보다 빠르지만 레지스터보다 느리다**.

일반적인 접근 속도는 레지스터 → 캐시 → 주기억장치 → 보조기억장치 순으로 느려진다. 따라서 가장 빠른 기억장치는 ③ 레지스터이다.

NoteBase

문 2. 10진수 −20을 2의 보수 형식의 8비트 2진수로 나타낸 것은?

① 10010100

② 11101011

③ 11101100

④ 11110100

정답 3번

간단한 해설

음수를 2의 보수로 나타내려면 양수의 8비트 표현에서 **모든 비트를 반전한 뒤 1을 더한다**. +20의 표현 00010100에 이 과정을 적용하면 −20은 11101100이 된다.

계산 과정

1. +20을 8비트로 표현한다. 20 = 16 + 4이므로 2진수로는 10100이며, 앞을 0으로 채우면 00010100이다.

2. 모든 비트를 반전한다. 8비트 전체에서 0은 1로, 1은 0으로 바꾸면 11101011이 된다. 이 값은 아직 **1의 보수**이므로 여기서 끝내면 안 된다.

3. 반전한 값에 1을 더한다. 2진수 덧셈으로 11101011 + 00000001 = 11101100을 얻는다. 이처럼 **비트 반전 후 1을 더한 값**이 −20의 8비트 2의 보수 표현이다. 최상위 비트 1은 음수를 나타내며, 부호 비트를 따로 덧붙이지 않는다.

구한 11101100은 ③의 값과 일치한다.

NoteBase

문 3. 어떤 프로세스가 일정 크기의 CPU 시간 할당량(time quantum)을 한 번 받은 후에는 강제로 대기 큐의 다른 프로세스에게 CPU를 넘겨주는 방식의 스케줄링 기법은?

① FCFS(First-Come-First-Served)

② RR(Round-Robin)

③ SPN(Shortest Process Next)

④ HRRN(Highest Response Ratio Next)

정답 2번

간단한 해설

RR(Round-Robin)은 각 프로세스에 일정한 CPU 시간 할당량을 주고, 그 시간을 다 쓰면 **실행 중인 프로세스의 CPU를 회수**하여 다음 프로세스에게 넘기는 선점형 스케줄링이다.

공통 해설

RR(Round-Robin)은 준비 큐의 프로세스들에게 CPU를 차례로 배분한다. 시간 할당량(time quantum)은 한 번의 차례에 연속으로 실행할 수 있는 최대 CPU 시간이다. 프로세스가 할당량을 모두 사용했는데도 실행을 마치지 못했다면 **CPU를 강제로 회수하고 해당 프로세스를 준비 큐의 맨 뒤로 보낸다**. 이후 큐의 다음 프로세스가 실행된다. 할당량을 다 쓰기 전에 작업을 마치거나 입출력을 기다리게 되면 그보다 먼저 CPU를 내놓는다.

예를 들어 A와 B가 실행을 기다리고 있고 시간 할당량이 10 ms라면, A가 10 ms를 사용한 뒤에도 할 일이 남아 있을 때 A를 큐의 뒤로 보내고 B를 실행한다. 이처럼 **시간 할당량 소진을 기준으로 실행 기회를 교대**하는 것이 문제의 조건이다. FCFS·SPN·HRRN은 기본적으로 비선점형이므로, 실행 중인 프로세스가 정해진 시간 할당량을 소진했다는 이유로 CPU를 강제로 회수하지 않는다.

NoteBase

문 4. 입력 안내에 따라 두 사람의 나이를 입력받고 그 합을 구하는 C 프로그램을 작성하려고 한다. 프로그램이 정상적으로 동작하도록 다음의 코드 조각을 올바른 순서로 나열한 것은?

ㄱ. scanf("%d%d", &age1, &age2);
ㄴ. result = age1 + age2;
ㄷ. int age1, age2, result;
ㄹ. printf("나이의 합은 %d살입니다.\n", result);
ㅁ. printf("철수와 영희의 나이를 입력하세요 :");

① ㄷ → ㅁ → ㄱ → ㄴ → ㄹ
② ㄷ → ㄱ → ㄴ → ㅁ → ㄹ
③ ㅁ → ㄱ → ㄷ → ㄹ → ㄴ
④ ㄷ → ㄱ → ㅁ → ㄴ → ㄹ

정답 1번

간단한 해설

변수를 사용하기 전에 선언하고, **입력 안내를 출력한 뒤 나이를 입력**받아야 한다. 입력된 두 값을 더해 result에 저장한 다음 그 값을 출력한다.

NoteBase

올바른 순서

ㄷ → ㅁ → ㄱ → ㄴ → ㄹ

ㄷ. int age1, age2, result;는 두 사람의 나이와 합을 저장할 정수형 변수를 선언한다. scanf나 계산식에서 변수를 사용하려면 **먼저 선언**되어 있어야 한다.

ㅁ. printf는 '철수와 영희의 나이를 입력하세요 :'라는 안내 문구를 출력한다. 문제에서 입력 안내에 따라 나이를 입력받도록 했으므로 **입력을 받는 scanf보다 앞**에 놓는다.

ㄱ. scanf는 두 정수를 읽어 age1과 age2에 각각 저장한다. %d는 정수 입력을 지정하고, &age1과 &age2는 입력값을 저장할 변수의 주소를 전달한다. 합을 계산하려면 이 단계에서 두 나이가 먼저 저장되어야 한다.

ㄴ. result = age1 + age2;는 입력받은 두 나이를 더한 값을 result에 저장한다. 여기서 =는 오른쪽 계산 결과를 왼쪽 변수에 저장하는 **대입 연산자**이므로, 계산이 끝나면 result에 출력할 합이 들어 있다.

ㄹ. printf는 result의 값을 %d 자리에 넣어 나이의 합을 출력하고, \n으로 줄을 바꾼다. result에 합이 저장되어 있어야 하므로 **합을 계산한 뒤** 실행한다.

NoteBase

문 5. 주소 변환을 위한 ARP(Address Resolution Protocol)에 대한 설명으로 옳지 않은 것은?

① ARP는 같은 네트워크 상에 있는 상대 호스트나 라우터의 논리 주소인 IP 주소를 획득하기 위해 사용된다.

② ARP 요청은 해당 네트워크 상의 모든 호스트와 라우터에게 브로드캐스트된다.

③ ARP 응답은 ARP 요청을 전송한 요청자에게 유니캐스트된다.

④ ARP 요청과 응답을 통해 획득한 주소 값을 ARP 캐시 테이블에 저장하여 통신 효율성을 높일 수 있다.

정답 1번

간단한 해설

ARP는 이미 알고 있는 IP 주소에 대응하는 **물리 주소인 MAC 주소**를 알아내는 프로토콜이다. ①의 'IP 주소를 획득하기 위해'는 'MAC 주소를 획득하기 위해'로 고쳐야 한다.

선지별 해설

① X: ARP는 같은 네트워크에 있는 호스트나 라우터의 IP 주소를 바탕으로 **그에 대응하는 MAC 주소**를 알아낸다. IP 주소는 논리 주소이고, MAC 주소는 링크 계층에서 프레임을 전달할 때 사용하는 물리 주소이다. 따라서 선지의 '논리 주소인 IP 주소를 획득'한다는 부분은 '물리 주소인 MAC 주소를 획득'한다고 고쳐야 한다.

② O: 상대의 IP 주소는 알지만 MAC 주소를 모르면, 그 상대에게만 프레임을 보낼 수 없으므로 ARP 요청을 **같은 네트워크에 브로드캐스트**한다. 요청을 받은 장치들은 요청 대상 IP 주소가 자신의 주소인지 확인한다. 이 브로드캐스트는 라우터를 넘어 다른 네트워크로 전달되지는 않는다.

③ O: 요청 대상 IP 주소를 가진 장치는 자신의 MAC 주소를 담아 응답한다. ARP 요청에는 요청자의 MAC 주소가 들어 있으므로, 응답은 **요청자에게 유니캐스트**할 수 있다. 이는 일반적인 ARP 요청·응답 과정에서의 전달 방식이다.

④ O: ARP 캐시 테이블은 **IP 주소와 MAC 주소의 대응 관계**를 저장한다. 통신할 상대의 유효한 대응 정보가 캐시에 있으면 다시 ARP 요청을 보내지 않고 MAC 주소를 사용할 수 있으므로, 주소 확인에 필요한 통신량과 지연을 줄일 수 있다.

NoteBase

문 6. QR코드에 대한 설명으로 옳지 않은 것은?

① 'Quick Response' 코드의 약자로 일본에서 개발되었다.

② 가로와 세로를 활용하는 2차원 형태로 이루어져 있다.

③ 기존 바코드보다 많은 양의 데이터를 넣을 수 있다.

④ 오류 정정(error correction) 기능이 없다.

정답 4번

간단한 해설

QR코드는 **오류 정정 기능이 있어** 일부가 오염되거나 손상되어도 복원 가능한 범위 안에서는 데이터를 읽을 수 있다. 따라서 '오류 정정 기능이 없다'는 설명은 잘못되었다.

선지별 해설

① O: QR은 **'Quick Response'**의 약자로, 빠르게 읽을 수 있다는 뜻을 담고 있다. QR코드는 일본의 덴소에서 개발되었다.

② O: QR코드는 검은색과 흰색의 작은 칸을 **가로와 세로 두 방향**으로 배열하여 정보를 표현하는 2차원 코드이다. 한 방향의 막대 폭과 간격으로 정보를 표현하는 일반적인 1차원 바코드와 구별된다.

③ O: QR코드는 가로와 세로를 모두 이용해 정보를 담으므로 **기존의 1차원 바코드보다 많은 데이터**를 저장할 수 있다. 숫자뿐 아니라 문자 등도 저장할 수 있어 인터넷 주소 같은 정보를 담는 데 쓰인다.

④ X: QR코드에는 **오류 정정 기능이 있다**. 데이터를 기록할 때 복원에 필요한 여분의 정보를 함께 넣기 때문에 일부가 오염되거나 손상되어도 원래 데이터를 복원할 수 있다. 다만 손상이 오류 정정 능력을 넘으면 복원하지 못할 수 있다.

NoteBase

문 7. 관계형 데이터베이스 설계에서의 정규화에 대한 설명으로 옳지 않은 것은?

① 질의처리 성능 향상을 위해 비효율적인 릴레이션들을 병합하는 과정이다.

② 데이터 중복을 감소시켜 저장 공간의 효율성을 향상시킨다.

③ 삽입, 삭제, 수정 시 발생할 수 있는 이상(anomaly) 현상을 제거한다.

④ 정규형에는 1NF, 2NF, 3NF, BCNF, 4NF, 5NF 등이 있다.

정답 1번

간단한 해설

정규화는 데이터 중복과 이상 현상을 줄이기 위해 릴레이션을 **분해**하는 과정이다. 질의처리 성능을 높이기 위해 릴레이션을 병합하는 것은 **반정규화**에 해당한다.

선지별 해설

① X: 정규화는 데이터의 종속 관계를 분석하여 릴레이션을 더 작은 릴레이션으로 분해하고, 불필요한 중복과 이상 현상을 줄이는 과정이다. 선지의 '질의처리 성능 향상을 위해 릴레이션들을 병합'하는 설명은 **반정규화**에 해당한다. 정규화로 분리된 데이터를 함께 조회할 때는 조인이 필요할 수 있으므로, 정규화가 언제나 질의 성능을 높이는 것은 아니다.

② O: 같은 사실을 여러 행에 반복하여 저장하지 않도록 구조를 나누므로 **불필요한 데이터 중복**을 줄일 수 있다. 예를 들어 학생마다 소속 학과의 이름과 전화번호를 반복 저장하는 대신, 학과 정보를 별도 릴레이션에 저장하고 학생 정보에는 학과를 식별하는 값만 둔다. 이렇게 반복 저장을 줄이면 저장 공간을 효율적으로 사용할 수 있다.

③ O: 이상 현상은 여러 종류의 사실을 한 릴레이션에 함께 저장하면서 삽입·삭제·수정이 의도하지 않은 제약이나 정보 손실·불일치를 일으키는 현상이다. 예를 들어 학과 전화번호가 여러 학생 행에 중복 저장되어 있으면 일부 행만 수정했을 때 서로 다른 번호가 남는 수정 이상이 발생한다. 정규화로 학과 정보를 분리하여 **한 곳에서 관리**하면 이러한 중복에 따른 이상 현상을 제거할 수 있다.

④ O: 정규형은 릴레이션이 충족하는 정규화 조건을 구분한 것으로, 1NF·2NF·3NF·BCNF·4NF·5NF 등이 있다. 각 정규형은 속성값의 원자성, 함수 종속, 다치 종속, 조인 종속 등의 조건을 기준으로 구분한다. 따라서 선지에 나열된 것들은 모두 정규형에 해당한다.

NoteBase

문 8. 속성 A, B, C로 정의된 릴레이션의 인스턴스가 아래와 같을 때, 후보키의 조건을 충족하는 것은?

A	B	C
1	12	7
20	12	7
1	12	3
1	1	4
1	2	6

① (A)

② (A, C)

③ (B, C)

④ (A, B, C)

정답 2번

간단한 해설

(A, C)의 값 쌍은 모든 행에서 달라 **유일성**을 충족한다. A와 C는 각각 단독으로는 중복값이 있으므로 둘 중 어느 속성도 뺄 수 없어 **최소성**도 충족한다.

NoteBase

공통 해설

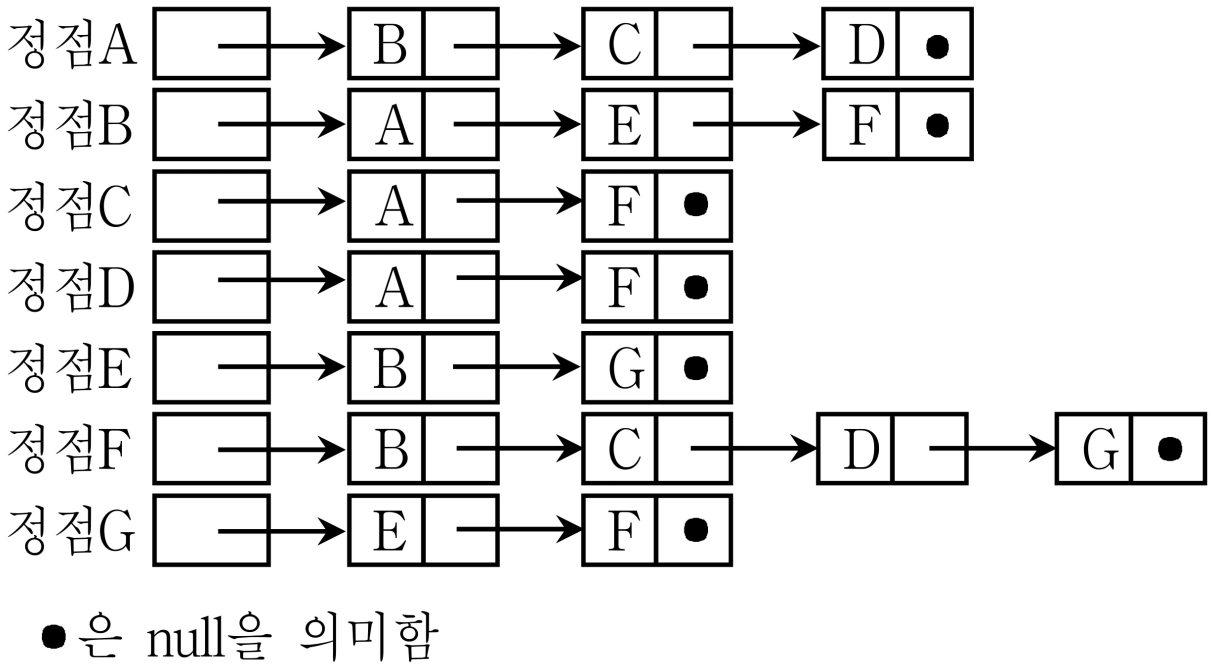
후보키는 릴레이션의 각 행을 유일하게 식별하면서 불필요한 속성을 포함하지 않는 속성 집합이다. 값의 조합이 행마다 달라야 한다는 '유일성'과, 그 집합에서 **어느 속성도 더 빼면 유일성을 유지할 수 없다**는 '최소성'을 모두 충족해야 한다.

표에서 A는 1이 여러 번 나오고, C는 7이 두 번 나오므로 각각 단독으로는 행을 구별하지 못한다. (B, C)도 첫째 행과 둘째 행의 값이 모두 (12, 7)이어서 유일성을 충족하지 못한다. 반면 (A, C)의 값은 위에서부터 (1, 7), (20, 7), (1, 3), (1, 4), (1, 6)으로 모두 다르다. 따라서 (A, C)는 유일성을 충족하며, A나 C 중 하나를 빼면 중복이 생기므로 최소성도 충족하는 후보키이다.

(A, B, C)는 후보키인 (A, C)를 포함하므로 각 행을 유일하게 식별할 수 있다. 그러나 **B를 빼도 유일성이 유지**되어 최소성을 충족하지 못한다. 따라서 (A, B, C)는 슈퍼키이지만 후보키는 아니다.

NoteBase

문 9. 다음의 인접리스트는 어떤 그래프를 표현한 것이다. 이 그래프를 정점 A에서부터 깊이 우선 탐색(depth first search)할 때, 정점이 방문되는 순서로 옳은 것은?



① A → B → C → D → F → G → E

② A → D → C → B → F → E → G

③ A → B → C → D → E → F → G

④ A → B → E → G → F → C → D

정답 4번

간단한 해설

깊이 우선 탐색은 현재 정점에서 **아직 방문하지 않은 인접 정점으로 계속 내려가는** 방식이다. 따라서 B를 방문한 뒤에는 A의 다음 인접 정점 C로 돌아가지 않고 E로 진행하며, C에서 더 갈 곳이 없어지면 F로 되돌아와 D를 방문한다.

NoteBase

올바른 순서

A → B → E → G → F → C → D

A: 깊이 우선 탐색은 정점을 방문한 뒤 미방문 인접 정점으로 탐색을 이어 가고, 더 갈 곳이 없으면 이전 정점으로 되돌아간다. 이 문제에서는 인접리스트에 나타난 왼쪽부터 확인한다. 시작 정점 A를 방문하고, A의 인접 정점 B·C·D 중 첫 번째인 B로 이동한다.

B: 인접리스트는 A·E·F 순서이다. A는 이미 방문했으므로 건너뛰고 **첫 미방문 정점 E**로 이동한다. A에 남아 있는 C·D는 B에서 이어지는 탐색을 마친 뒤 확인한다.

E: 인접리스트는 B·G 순서이다. B는 이미 방문했으므로 G로 이동한다.

G: 인접리스트는 E·F 순서이다. E는 이미 방문했으므로 F로 이동한다.

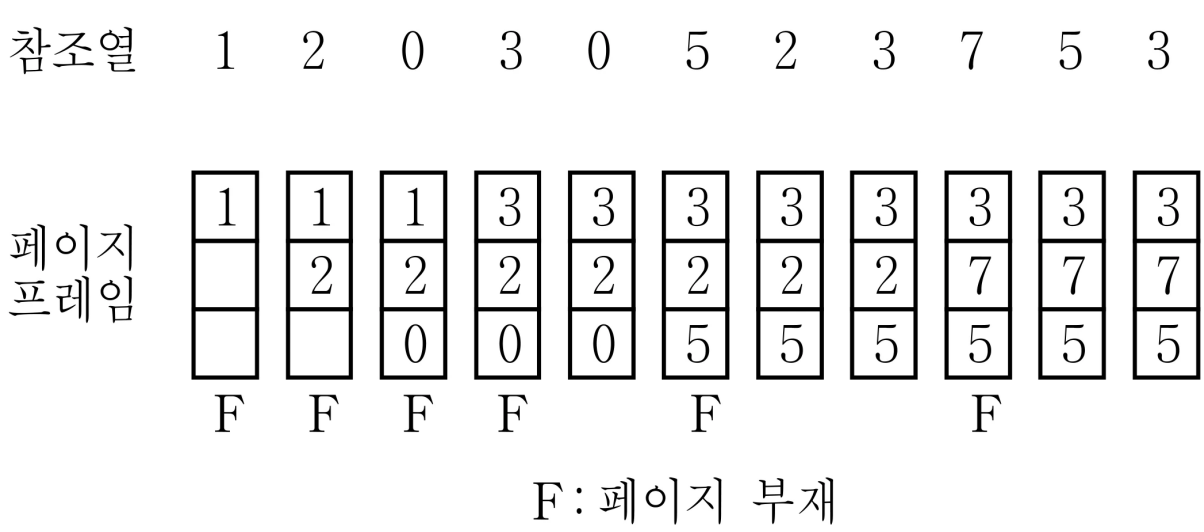
F: 인접리스트는 B·C·D·G 순서이다. B는 이미 방문했고 C는 아직 방문하지 않았으므로 **D보다 먼저 C**로 이동한다.

C: 인접 정점 A·F가 모두 이미 방문된 상태이므로 더 진행할 수 없다. C의 탐색을 마치고 **직전에 탐색하던 F로 되돌아간다**. 되돌아가는 F는 새로 방문한 정점으로 기록하지 않는다.

D: F에서 C 다음으로 확인하는 미방문 인접 정점은 D이므로 D를 방문한다. D의 인접 정점 A·F는 모두 이미 방문했다. 이후 이전 정점들로 되돌아가도 새로 방문할 정점이 없으므로 탐색을 마친다.

NoteBase

문 10. 다음은 가상 메모리의 페이지 교체 정책 중 최적(optimal) 알고리즘을 적용하여 페이지를 할당한 예이다. 참조열 순으로 페이지가 참조될 때, 페이지 부재(page fault)가 6회 발생하였다. 동일한 조건 하에서 LRU(Least Recently Used) 알고리즘을 적용할 경우 페이지 부재가 몇 회 발생하는가?



① 6

② 7

③ 8

④ 9

정답 4번

간단한 해설

LRU는 프레임이 가득 차면 **마지막으로 참조된 시점이 가장 오래된 페이지**를 교체한다. 주어진 참조열에서는 5번째와 11번째 참조에서만 페이지가 이미 메모리에 있으므로, 페이지 부재는 11 − 2 = 9회 발생한다.

NoteBase

계산 과정

그림의 페이지 프레임은 3개이고, 참조열은 1 → 2 → 0 → 3 → 0 → 5 → 2 → 3 → 7 → 5 → 3이다. 처음에는 모든 프레임이 비어 있다. 참조한 페이지가 메모리에 없으면 페이지 부재가 발생하며, 빈 프레임이 있으면 적재하고 모두 차 있으면 교체한다. 그림에 표시된 6회는 최적 알고리즘의 결과이므로 LRU의 교체 과정을 별도로 추적해야 한다.

LRU는 **과거의 마지막 참조 시점**을 기준으로 교체 대상을 정한다. 이미 메모리에 있는 페이지를 참조해도 그 페이지의 최근 사용 기록은 갱신된다. 아래의 괄호는 실제 프레임 위치가 아니라, 메모리에 있는 페이지를 '가장 오래전에 사용한 것 → 가장 최근에 사용한 것' 순서로 나타낸다.

1~3번째 참조 1, 2, 0: 각각 처음 등장하므로 빈 프레임에 적재한다. 페이지 부재는 3회이고, 사용 순서는 (1, 2, 0)이다.

4번째 참조 3: 메모리에 없으므로 가장 오래전에 사용한 1을 교체한다. 사용 순서는 (2, 0, 3), 누적 페이지 부재는 4회이다.

5번째 참조 0: 이미 메모리에 있으므로 페이지 부재는 없다. 다만 **0이 가장 최근에 사용한 페이지**가 되어 사용 순서는 (2, 3, 0)으로 바뀐다.

6번째 참조 5: 메모리에 없으므로 2를 교체한다. 사용 순서는 (3, 0, 5), 누적 페이지 부재는 5회이다.

7번째 참조 2: 앞서 교체되어 메모리에 없으므로 3을 교체한다. 사용 순서는 (0, 5, 2), 누적 페이지 부재는 6회이다.

8번째 참조 3: 메모리에 없으므로 0을 교체한다. 사용 순서는 (5, 2, 3), 누적 페이지 부재는 7회이다.

9번째 참조 7: 메모리에 없으므로 5를 교체한다. 사용 순서는 (2, 3, 7), 누적 페이지 부재는 8회이다.

10번째 참조 5: 메모리에 없으므로 2를 교체한다. 사용 순서는 (3, 7, 5), 누적 페이지 부재는 9회이다.

11번째 참조 3: 이미 메모리에 있으므로 페이지 부재는 없고, 사용 순서만 (7, 5, 3)으로 바뀐다.

총 11번의 참조 중 페이지 부재가 없는 경우는 5번째와 11번째의 2번이다. 따라서 페이지 부재는 11 − 2 = 9회로, ④의 값과 일치한다.

NoteBase

문 11. IT 기술에 관한 설명으로 옳지 않은 것은?

① IoT(Internet of Things)는 각종 사물에 센서와 통신 기능을 내장하여 인터넷에 연결하는 기술이다.

② 공용 클라우드(public cloud)는 한 기업의 정보 보안을 위해 내부 데이터 센터의 기능을 강화한 형태이다.

③ 빅데이터는 수집·저장된 대량의 정형 또는 비정형 데이터 집합으로부터 가치를 추출하고 결과를 분석하는 기술이다.

④ 가상현실은 가상의 공간과 사물을 컴퓨터에서 만들어, 인간 오감을 활용한 작용으로 현실 세계에서는 경험하지 못하는 상황을 간접적으로 체험할 수 있도록 해준다.

정답 2번

간단한 해설

공용 클라우드는 외부 사업자가 컴퓨팅 자원을 **불특정 다수에게 서비스로 제공**하는 형태이다. 한 기업이 보안과 통제를 위해 전용 환경을 사용하는 설명은 **사설 클라우드(private cloud)**에 해당한다.

선지별 해설

① O: IoT는 사물에 센서와 통신 기능을 갖추어 **인터넷으로 데이터를 주고받게 하는 기술**이다. 예를 들어 온도 센서가 측정값을 인터넷으로 전송하면 사용자가 원격으로 온도를 확인할 수 있다.

② X: 공용 클라우드는 클라우드 사업자가 서버·저장 공간 등의 자원을 마련하고 **불특정 다수의 이용자에게 서비스로 제공**하는 형태이다. 한 기업만을 위한 전용 환경을 구성하여 보안과 통제를 강화하는 것은 **사설 클라우드**에 해당한다. 사설 클라우드는 기업 내부 데이터 센터에 구축할 수도 있고 외부 사업자의 시설에 구축할 수도 있으므로, 핵심 구분 기준은 물리적 위치보다 이용 대상이 특정 조직으로 한정되는지에 있다.

③ O: 빅데이터는 대량의 정형·비정형 데이터와 이를 수집·저장·처리·분석하는 기술을 포괄한다. 단순히 많은 데이터를 보관하는 데 그치지 않고 **분석을 통해 유용한 정보와 가치를 추출**한다. 예를 들어 판매 기록과 고객의 후기 글을 함께 분석하여 소비 경향을 파악할 수 있다.

④ O: 가상현실은 컴퓨터로 만든 공간과 사물을 사용자가 실제처럼 체험하도록 하는 기술이다. 시각·청각 등의 감각 자극과 사용자의 움직임에 대한 반응을 통해 **가상 환경에 몰입**하게 하며, 현실에서 직접 경험하기 어렵거나 위험한 상황도 간접적으로 체험하게 한다.

NoteBase

문 12. 2진 부동소수점 수를 표현하기 위한 표준 형식의 요소가 아닌 것은?

① 지수(exponent)

② 가수(fraction 또는 mantissa)

③ 기수(base)

④ 부호(sign)

정답 3번

간단한 해설

2진 부동소수점 형식에서는 **기수가 2로 고정**되어 있으므로 기수를 따로 저장하지 않는다. 저장 형식의 필드는 부호·지수·가수로 구성된다.

공통 해설

부동소수점 표현은 수를 '부호 × 가수 × 기수의 거듭제곱' 형태로 나타내는 방식이다. 2진 부동소수점에서는 기수가 항상 2이므로, 수마다 달라지는 정보를 **부호·지수·가수** 필드에 나누어 저장한다. 부호는 수의 부호를, 지수는 2의 몇 제곱을 곱할지를, 가수는 수의 유효숫자를 나타낸다.

기수는 거듭제곱의 밑이지만, 2진 형식 자체에서 이미 2로 정해져 있다. 따라서 **기수를 위한 별도 필드는 없다**. 기수는 부동소수점 수를 설명하는 개념이지만, 이 문제에서 묻는 저장 형식의 구성 필드에는 해당하지 않는다.

NoteBase

문 13. 범용 컴퓨터의 시스템 버스(system bus)에 해당하지 않는 것은?

① 주소 버스(address bus)

② 데이터 버스(data bus)

③ 제어 버스(control bus)

④ 명령어 버스(instruction bus)

정답 4번

간단한 해설

명령어도 메모리에서 CPU로 읽어 올 때는 **데이터 버스**로 전달된다. '명령어 버스'는 범용 컴퓨터의 시스템 버스를 나누는 기본 분류에 포함되지 않는다.

공통 해설

시스템 버스는 CPU·주기억장치·입출력장치 사이에서 정보를 전달하는 공통 통로이며, **전달하는 정보의 역할**에 따라 주소 버스·데이터 버스·제어 버스로 구분한다. 주소 버스는 접근할 기억장소나 입출력장치의 위치를, 데이터 버스는 주고받는 정보의 내용을, 제어 버스는 읽기·쓰기 등 동작을 지시하거나 조정하는 신호를 전달한다.

예를 들어 CPU가 메모리에서 명령어를 가져올 때는 주소 버스로 명령어가 저장된 위치를 지정하고, 제어 버스로 읽기 신호를 보내며, 데이터 버스로 명령어를 전달받는다. 즉, **명령어도 데이터 버스를 통해 이동**하므로 명령어 전달을 별도의 기본 버스 종류로 분류하지 않는다. 따라서 '명령어 버스'는 이 세 종류로 구성되는 시스템 버스의 기본 분류에 해당하지 않는다.

NoteBase

문 14. 통신 프로토콜에 대한 설명으로 옳은 것은?

① MIME(Multipurpose Internet Mail Extensions)는 인터넷 상에서 디지털 오디오 및 비디오 신호를 실시간으로 전달하기 위한 전송 계층 프로토콜이다.

② TFTP(Trivial File Transfer Protocol)는 안전한 파일 전송을 위해 인증과 TCP를 필수 구성 요소로 한다.

③ TELNET는 가상 터미널 연결을 위한 응용 계층 프로토콜로 텍스트 기반 양방향 통신 기능을 제공한다.

④ DHCP(Dynamic Host Configuration Protocol)는 호스트의 인터넷 도메인 명을 IP 주소로 변환시켜 주는 것이다.

정답 3번

간단한 해설

TELNET는 원격 컴퓨터를 터미널처럼 이용하도록 하는 **응용 계층 프로토콜**이다. 사용자가 입력한 문자와 원격 컴퓨터의 응답을 주고받는 **텍스트 기반 양방향 통신**을 제공한다.

선지별 해설

① X: MIME은 전자우편에서 일반 텍스트 외에 이미지·오디오·비디오 등의 다양한 형식을 다룰 수 있도록 하는 **전자우편 형식의 확장 규격**이다. 첨부파일의 종류와 인코딩 방식 등을 표현하며, 실시간 오디오·비디오 전송을 위한 전송 계층 프로토콜은 아니다.

② X: TFTP는 단순한 파일 전송을 위해 **TCP가 아니라 UDP를 사용**하는 응용 계층 프로토콜이다. 기본 프로토콜 자체에는 사용자 인증 기능이 없으므로 인증과 TCP를 필수 구성 요소로 한다는 설명은 틀리다.

③ O: TELNET는 원격 시스템에 가상 터미널로 접속하기 위한 응용 계층 프로토콜이다. 사용자의 문자 입력을 원격 시스템에 전달하고 그 실행 결과를 돌려받으므로 **텍스트 기반 양방향 통신**을 제공한다. 예를 들어 사용자가 원격 컴퓨터에 명령을 입력하면 그 컴퓨터가 처리 결과를 문자로 보내 준다.

④ X: DHCP는 호스트에 **IP 주소와 네트워크 설정 정보를 자동으로 할당**하는 응용 계층 프로토콜이다. 서브넷 마스크와 기본 게이트웨이 등의 설정 정보도 제공할 수 있다. 인터넷 도메인 이름에 대응하는 IP 주소를 조회하는 기능은 DNS가 담당한다.

NoteBase

문 15. 브라우저가 웹 서버로부터 정보를 읽어 오기 위해 사용하는 응용 계층 프로토콜은?

① SMTP

② HTTP

③ IMAP

④ RTP

정답 2번

간단한 해설

브라우저는 **HTTP**를 사용해 웹 서버에 필요한 자원을 요청하고, 서버의 응답으로 웹 문서나 이미지 등을 받아 온다.

공통 해설

HTTP는 브라우저와 같은 웹 클라이언트와 웹 서버 사이에서 **요청과 응답**을 주고받는 응용 계층 프로토콜이다. 브라우저가 웹 서버의 자원을 읽어 오려면 HTTP의 GET 메서드로 해당 자원을 요청할 수 있다. 서버는 요청을 처리하여 상태 정보와 함께 웹 문서나 이미지 등의 내용을 응답으로 보낸다. 이처럼 웹 서버에 정보를 요청하고 그 결과를 받아 오는 기능에 해당하는 프로토콜은 HTTP이다.

NoteBase

문 16. 다음의 부울함수와 같은 논리식이 아닌 것은?

F(x,y,z)=Σ m(1,3,4,5,6)

① $\overline{x}\overline{y}z + \overline{x}yz + x\overline{y}\overline{z} + x\overline{y}z + xy\overline{z}$

② $(x + y + z)(x + \overline{y} + z)(\overline{x} + \overline{y} + \overline{z})$

③ $\overline{x}z + x\overline{z} + xy$

④ $\overline{x}z + x\overline{z} + \overline{y}z$

정답 3번

간단한 해설

③의 식은 입력 $(x, y, z) = (1, 0, 1)$에서 **0을 출력**하지만, 주어진 함수는 최소항 $m(5)$를 포함하므로 **1을 출력**한다. 따라서 두 식은 같은 부울함수가 아니다.

계산 과정

최소항 $m(i)$는 입력을 xyz 순서의 이진수로 읽은 값이 i일 때만 1이 되는 항이다. 주어진 함수는 입력 번호 1, 3, 4, 5, 6에서 1이고, 나머지 0, 2, 7에서 0이다. 여기서 +는 OR, 곱은 AND, 윗줄은 NOT을 뜻한다. 두 논리식이 같으려면 **모든 입력에서 출력이 같아야 한다**.

① 각 곱항이 나타내는 최소항 번호는 차례대로 1, 3, 4, 5, 6이다. 예를 들어 $\overline{x}\overline{y}z$는 입력 001에서만 1이므로 $m(1)$이다. 이 다섯 곱항을 OR로 연결했으므로 주어진 함수와 같다.

② 괄호 안의 OR 식은 **모든 항이 0일 때만 0**이다. 첫 괄호는 입력 000, 둘째 괄호는 입력 010, 셋째 괄호는 입력 111에서 각각 0이 된다. 전체는 세 괄호의 AND이므로 입력 번호 0, 2, 7에서만 0이고, 나머지 1, 3, 4, 5, 6에서 1이다. 따라서 주어진 함수와 같다.

③ $\overline{x}z$는 입력 번호 1, 3에서, $x\overline{z}$는 4, 6에서, xy는 6, 7에서 1이다. OR로 연결한 전체 식은 1, 3, 4, 6, 7에서 1이므로 주어진 함수와 다르다. 즉, **최소항 5가 빠지고 7이 추가된다**. 실제로 입력 101에서는 주어진 함수가 1인데 이 식은 0이며, 입력 111에서는 주어진 함수가 0인데 이 식은 1이다. 마지막 항 xy를 $\overline{y}z$로 바꾸면 주어진 함수와 같은 식이 된다.

④ $\overline{x}z$는 입력 번호 1, 3에서, $x\overline{z}$는 4, 6에서, $\overline{y}z$는 1, 5에서 1이다. OR에서는 같은 최소항이 겹쳐도 한 번 포함된 것과 같으므로 전체 식은 1, 3, 4, 5, 6에서 1이다. 따라서 주어진 함수와 같다.

각 식의 출력을 비교하면 주어진 함수와 다른 논리식은 ③이다.

NoteBase

문 17. 다음의 데이터 링크 계층 오류제어 기법들을 프레임 전송 효율이 좋은 것부터 나쁜 순으로 바르게 나열한 것은? (단, 여러 개의 프레임을 전송할 때 평균적으로 요구되는 전송 및 대기 시간만을 고려하되, 송신 및 수신단에 요구되는 구현의 복잡도나 운용에 따른 비용은 무시한다)

ㄱ. 정지 후 대기(stop-and-wait) ARQ
ㄴ. N 복귀(go-back-N) ARQ
ㄷ. 선택적 반복(selective-repeat) ARQ

① ㄱ → ㄴ → ㄷ
② ㄴ → ㄷ → ㄱ
③ ㄷ → ㄱ → ㄴ
④ ㄷ → ㄴ → ㄱ

정답 4번

간단한 해설

선택적 반복 ARQ는 **오류·유실된 프레임만 재전송**하므로 정상 프레임까지 다시 보내는 N 복귀 ARQ보다 효율이 높다. 정지 후 대기 ARQ는 **프레임 하나마다 확인 응답을 기다려** 전송 사이의 대기 시간이 크다.

NoteBase

올바른 순서

ㄷ → ㄴ → ㄱ

ㄷ. 선택적 반복(selective-repeat) ARQ: 송신 윈도 크기 내에서 여러 프레임을 연속으로 보내고, 수신 측은 순서가 어긋나도 정상적으로 받은 프레임을 저장한다. 오류·유실된 프레임만 다시 보내므로 **이미 정상 수신한 프레임의 불필요한 재전송**을 줄인다. 예를 들어 1~4번을 보냈는데 2번만 손상되었다면, 정상 수신한 3·4번을 저장해 두고 2번만 다시 받는다. 문제에서 구현 복잡도와 운용 비용을 제외하므로 이러한 저장·관리 부담은 효율 비교에 반영하지 않는다.

ㄴ. N 복귀(go-back-N) ARQ: 송신 윈도 크기 내에서 확인 응답을 기다리는 동안에도 다음 프레임을 연속으로 보낼 수 있다. 다만 수신 측은 기대한 순서의 프레임만 받아들이므로, 오류·유실이 생기면 **해당 프레임부터 이후의 미확인 프레임들까지 재전송**한다. 위 예에서 3·4번까지 이미 전송했다면 2·3·4번을 다시 보내게 되어 선택적 반복보다 전송 시간이 더 든다. 그래도 프레임마다 멈추지는 않으므로 정지 후 대기보다 대기 시간을 줄인다.

ㄱ. 정지 후 대기(stop-and-wait) ARQ: 프레임 하나를 보낸 뒤 **그 프레임의 확인 응답을 받아야** 다음 프레임을 보낸다. 오류·유실로 확인 응답을 받지 못하면 타임아웃 후 해당 프레임을 재전송한다. 여러 프레임을 보낼 때 매번 확인 응답을 기다리는 동안 새 프레임을 전송하지 못하므로 세 방식 중 전송 효율이 가장 낮다.

NoteBase

문 18. 다음과 같은 코드로 동작하는 원형 큐의 front와 rear의 값이 각각 7과 2일 때, 이 원형 큐(queue)가 가지고 있는 데이터(item)의 개수는? (단, MAX_QUEUE_SIZE는 12이고, front와 rear의 초깃값은 0이다)

```
int queue[MAX_QUEUE_SIZE];
int front, rear;
void enqueue(int item) {
    if( (rear + 1) % MAX_QUEUE_SIZE == front ) {
        printf("queue is full\n");
        return;
    }
    rear = (rear + 1) % MAX_QUEUE_SIZE;
    queue[rear] = item;
}
int dequeue() {
    if( front == rear ) {
        printf("queue is empty\n");
        return -1;
    }
    front = (front + 1) % MAX_QUEUE_SIZE;
    return queue[front];
}
```

① 5

② 6

③ 7

④ 8

정답 3번

간단한 해설

이 코드에서 데이터는 **front의 다음 칸부터 rear가 가리키는 칸까지** 들어 있다. 인덱스 11 다음은 0으로 이어지므로, 데이터가 있는 칸은 8·9·10·11·0·1·2의 7개이다.

NoteBase

계산 과정

위치의 의미: enqueue는 rear를 한 칸 이동시킨 뒤 데이터를 저장하고, dequeue는 front를 한 칸 이동시킨 뒤 그 칸의 데이터를 꺼낸다. 따라서 front는 **첫 데이터의 바로 앞 위치**이고, rear는 마지막 데이터의 위치이다. front와 rear의 초깃값이 모두 0이면 빈 큐이며, 현재 front = 7, rear = 2에서는 인덱스 7을 제외하고 2를 포함하여 센다.

원형 구간 계산: MAX_QUEUE_SIZE가 12이므로 배열 인덱스는 0~11이다. 코드의 나머지 연산(%) 때문에 인덱스 11 다음은 0으로 돌아간다. rear가 front보다 작으므로 데이터 구간은 **배열 끝을 지나 0으로 이어진다**. 인덱스 8~11의 데이터는 11 − 7 = 4개이고, 인덱스 0~2의 데이터는 2 + 1 = 3개이다.

전체 데이터 수: 두 구간을 합하면 4 + 3 = 7개이며, 이는 ③의 7에 해당한다.

NoteBase

문 19. 다음 자바 코드를 컴파일할 때, 문법 오류가 발생하는 부분은?

```
class Person {
   private String name;
   public int age;
   public void setAge(int age) {
        this.age = age;
   }
   public String toString() {
        return("name: " + this.name + ", age : " + this.age);
   }
}
public class PersonTest {
   public static void main(String[] args) {
        Person a = new Person();          // ㉠
        a.setAge(27);                     // ㉡
        a.name = "Gildong";               // ㉢
        System.out.println(a);            // ㉣
   }
}
```

① ㉠

② ㉡

③ ㉢

④ ㉣

정답 3번

간단한 해설

name은 **private으로 선언된 필드**이므로 별개의 클래스인 PersonTest에서 a.name으로 직접 접근할 수 없다. 이름을 설정하려면 Person 내부에서 name에 값을 대입하는 public 메서드를 두고 호출해야 한다.

NoteBase

핵심 풀이

㉠ 객체 생성: Person에는 생성자가 명시되어 있지 않으므로 **매개변수가 없는 기본 생성자**가 자동으로 제공된다. 따라서 new Person()으로 객체를 생성하여 Person형 참조 변수 a에 저장할 수 있다.

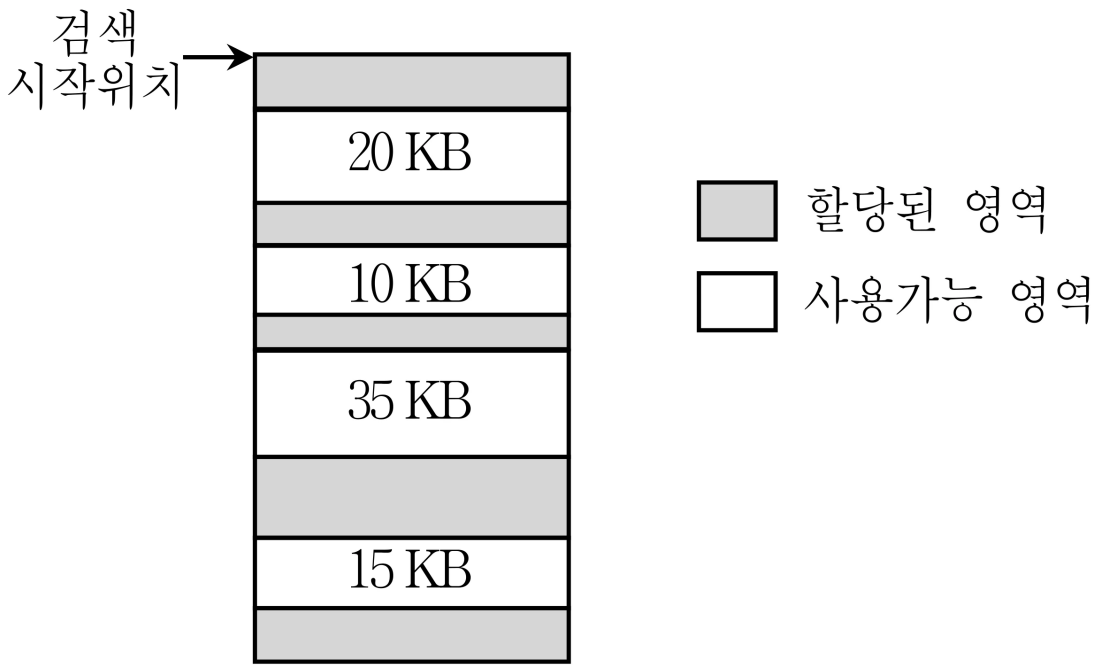
㉡ 메서드 호출: setAge(int age)는 public으로 선언되어 있으므로 PersonTest에서 호출할 수 있다. 인수 27은 int형 매개변수에 전달되며, 메서드 안의 this.age = age는 **객체의 필드 age에 매개변수의 값**을 대입한다. 이 호출 자체에는 컴파일 오류가 없다.

㉢ 필드 접근: name은 Person의 private 필드인데, a.name에 값을 대입하는 코드는 별개의 클래스인 PersonTest에 있다. 따라서 **접근 권한 위반으로 컴파일 오류**가 발생한다. 이를 고치려면 Person에 public void setName(String name) 메서드를 추가하여 그 안에서 this.name = name;을 수행하고, PersonTest에서는 a.setName("Gildong");으로 호출할 수 있다. 같은 파일에 두 클래스가 있어도 PersonTest에서 Person의 private 필드에 직접 접근할 수는 없다.

㉣ 객체 출력: System.out.println(a)는 객체를 인수로 받을 수 있으므로 이 문장 자체에는 컴파일 오류가 없다. ㉢의 오류를 고쳐 실행하면 **Person에서 재정의한 toString()**이 반환하는 문자열이 출력된다. toString() 안의 this.name 접근은 필드를 선언한 Person 내부에서 이루어지므로 허용된다.

NoteBase

문 20. 크기가 각각 12 KB, 30 KB, 20 KB인 프로세스가 다음과 같은 메모리 공간에 순차적으로 적재 요청될 때, 모든 프로세스를 적재할 수 있는 알고리즘만을 모두 고른 것은?



ㄱ. 최초 적합(first-fit)
ㄴ. 최적 적합(best-fit)
ㄷ. 최악 적합(worst-fit)

① ㄱ
② ㄴ
③ ㄱ, ㄴ
④ ㄴ, ㄷ

정답 2번

간단한 해설

최적 적합은 요청을 수용할 수 있는 빈 공간 중 **가장 작은 공간**을 선택한다. 12 KB를 15 KB 공간에 먼저 넣으면 35 KB와 20 KB 공간이 보존되어, 뒤의 30 KB와 20 KB 요청도 모두 수용할 수 있다.

NoteBase

보기별 해설

그림의 흰색 사용가능 영역은 검색 시작위치에서 아래쪽으로 20 KB, 10 KB, 35 KB, 15 KB 순서이다. 각 알고리즘은 이 초기 상태에서 따로 적용하며, 요청 순서는 12 KB → 30 KB → 20 KB이다. 한 프로세스는 하나의 연속된 빈 공간에 적재하고, 적재한 공간에서는 프로세스 크기만큼을 뺀 나머지를 빈 공간으로 남긴다. 회색의 할당된 영역 사이에 떨어져 있는 빈 공간을 합쳐 적재할 수는 없다.

ㄱ. 최초 적합(first-fit): 검색 시작위치부터 살펴보며 **요청을 수용하는 첫 번째 빈 공간**을 선택한다. 12 KB는 처음의 20 KB 공간에 들어가므로 20 − 12 = 8 KB가 남는다. 다음 30 KB는 35 KB 공간에 들어가므로 35 − 30 = 5 KB가 남는다. 이때 빈 공간은 위에서부터 8 KB, 10 KB, 5 KB, 15 KB이다. 어느 공간도 20 KB 이상이 아니므로 마지막 20 KB 프로세스를 적재할 수 없다.

ㄴ. 최적 적합(best-fit): 요청을 수용할 수 있는 빈 공간 중 **가장 작은 공간**을 선택한다. 12 KB를 수용하는 공간은 20 KB, 35 KB, 15 KB이므로 가장 작은 15 KB 공간에 적재하여 15 − 12 = 3 KB를 남긴다. 다음 30 KB는 35 KB 공간에 적재하여 35 − 30 = 5 KB를 남긴다. 마지막 20 KB는 그대로 보존된 20 KB 공간에 정확히 들어간다. 따라서 세 프로세스를 모두 적재할 수 있다.

ㄷ. 최악 적합(worst-fit): **가장 큰 빈 공간**을 선택한다. 첫 12 KB를 가장 큰 35 KB 공간에 적재하면 35 − 12 = 23 KB가 남는다. 이때 빈 공간은 위에서부터 20 KB, 10 KB, 23 KB, 15 KB이며, 가장 큰 공간도 23 KB이다. 따라서 두 번째 요청인 30 KB부터 적재할 수 없어 세 프로세스를 모두 적재하지 못한다.

모든 프로세스를 적재할 수 있는 알고리즘은 ㄴ뿐이므로 ②의 구성과 일치한다.

NoteBase