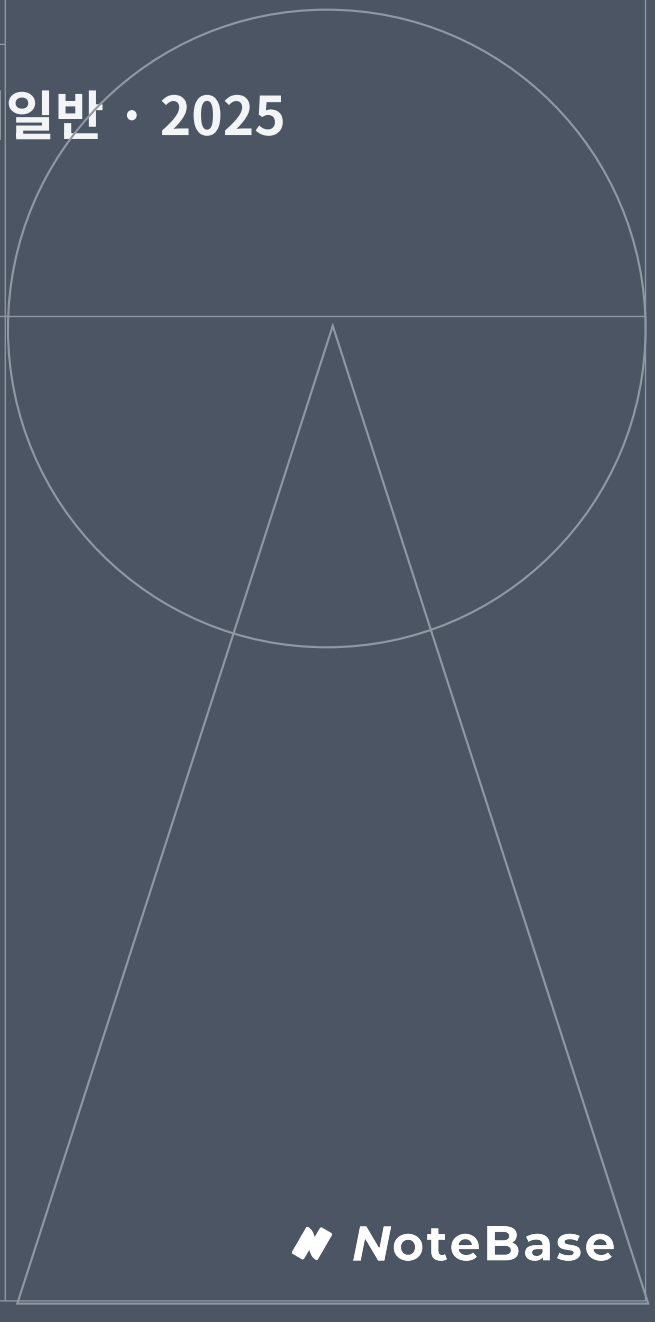
기출OX 해설집

공무원/국가직/9급 · 컴퓨터일반 · 2025



NoteBase

기출OX 해설집

공무원/국가직/9급 · 컴퓨터일반 · 2025

공무원/국가직/9급 · 컴퓨터일반 · 2025

문 1. 사용자가 언제 어디서나 필요한 컴퓨팅 자원(하드웨어/소프트웨어)에 접근하여 활용할 수 있는 유연한 환경을 의미하는 기술은?

① 빅데이터

② 가상현실

③ 사물인터넷

④ 클라우드 컴퓨팅

정답 4번

간단한 해설

클라우드 컴퓨팅은 **네트워크를 통해 필요한 컴퓨팅 자원을 제공받아 이용**하는 방식이다. 자원이 있는 장소에 직접 가지 않고도 접속하여 하드웨어와 소프트웨어를 활용할 수 있다.

공통 해설

클라우드 컴퓨팅은 서버의 처리 능력, 저장 공간, 소프트웨어 같은 컴퓨팅 자원을 **네트워크를 통해 필요할 때 이용**하는 방식이다. 사용자는 자원이 있는 물리적 장소에 직접 가지 않고, 네트워크에 접속할 수 있는 환경에서 필요한 자원을 활용한다. 예를 들어 클라우드 문서 편집 서비스에 접속하면 집이나 학교에서 같은 서비스를 이용해 문서를 작성할 수 있다. 따라서 발문의 '언제 어디서나 필요한 컴퓨팅 자원에 접근하여 활용'한다는 설명은 클라우드 컴퓨팅의 자원 제공 방식에 해당한다.

NoteBase

문 2. 스택을 이용하여 구현할 수 있는 것이 아닌 것은?

① 괄호 매칭

② 진법 변환

③ 후위표현 연산

④ 은행 대기열

정답 4번

간단한 해설

은행 대기열은 **먼저 온 고객을 먼저 처리**하는 선입선출 방식이므로 큐의 대표적인 활용이다. 스택은 나중에 들어온 자료를 먼저 꺼내는 후입선출 방식이다.

선지별 해설

스택은 한쪽 끝에서 자료를 넣고 꺼내는 후입선출 구조이다. A를 넣은 뒤 B를 넣으면 B가 먼저 나온다. 자료를 넣는 연산은 push, 꺼내는 연산은 pop이다.

① O: 괄호 매칭에서는 여는 괄호를 스택에 넣고, 닫는 괄호를 만나면 맨 위의 여는 괄호를 꺼내 종류가 맞는지 확인한다. 중첩된 괄호는 **가장 나중에 열린 괄호부터 닫혀야** 하므로 스택의 처리 순서와 맞는다. 도중에 꺼낼 괄호가 없거나 종류가 다르거나, 처리가 끝난 뒤 여는 괄호가 남으면 올바른 짝이 아니다.

② O: 정수를 다른 진법으로 변환할 때는 해당 진수로 반복해서 나눈 나머지를 스택에 넣을 수 있다. 나머지는 낮은 자리부터 얻어지지만 변환 결과는 높은 자리부터 읽어야 하므로, 스택에서 **나머지를 얻은 순서의 역순**으로 꺼내면 된다.

③ O: 후위표현 연산에서는 식을 왼쪽부터 읽으며 피연산자를 스택에 넣는다. 연산자를 만나면 필요한 피연산자를 꺼내 계산하고, **계산 결과를 다시 스택에 넣어** 다음 연산에 사용한다. 뺄셈이나 나눗셈처럼 순서가 중요한 이항 연산에서는 먼저 꺼낸 값이 오른쪽 피연산자이고, 그다음 꺼낸 값이 왼쪽 피연산자이다.

④ X: 은행 대기열은 먼저 도착한 고객을 먼저 처리해야 하므로 **선입선출 방식의 큐**를 사용하는 대표적인 사례이다. 고객을 스택에 넣고 그대로 꺼내면 나중에 도착한 고객이 먼저 나와 대기 순서와 맞지 않는다. 따라서 은행 대기열을 스택의 대표적인 활용으로 보는 것은 적절하지 않다.

NoteBase

문 3. 블록체인의 특징이 아닌 것은?

① 해시값을 사용하여 변조를 방지한다.

② 모든 거래는 거래 당사자만 확인할 수 있다.

③ 모든 노드는 P2P 네트워크를 이용해 연결되어 있다.

④ 원장을 조작한 이중 지불을 방지하기 위해 합의 알고리즘을 사용한다.

정답 2번

간단한 해설

공개형 블록체인에서는 거래 기록을 **거래 당사자 외의 참여자도 확인**할 수 있다. 거래 내역을 여러 참여자가 공유·검증하므로, '거래 당사자만 확인할 수 있다'는 블록체인의 일반적 특징이 아니다.

선지별 해설

① O: 해시 함수는 데이터를 일정한 길이의 해시값으로 바꾸며, 데이터가 바뀌면 해시값도 달라진다. 블록체인은 각 블록에 이전 블록의 해시값을 포함해 연결하므로, 과거 기록을 바꾸면 뒤 블록과의 연결이 맞지 않게 된다. 이처럼 **기록 변경을 탐지**하고 합의 절차와 함께 임의의 변조를 어렵게 한다.

② X: 블록체인은 여러 참여자가 거래 기록을 나누어 보관하고 검증하는 분산 원장이다. 공개형 블록체인에서는 **거래 당사자가 아닌 참여자도 거래 내역을 확인**할 수 있다. 따라서 '거래 당사자만 확인할 수 있다'는 설명은 잘못이다. 다만 허가형 블록체인에서는 설계에 따라 열람 권한을 제한할 수 있다.

③ O: 블록체인의 노드는 P2P 네트워크로 연결되어 거래와 블록 정보를 주고받는다. P2P는 각 노드가 다른 노드와 직접 정보를 교환하는 방식으로, **중앙 서버에만 의존하지 않고**원장을 분산하여 유지할 수 있게 한다.

④ O: 이중 지불은 같은 디지털 자산을 두 번 이상 지불하려는 것이다. 합의 알고리즘은 노드들이 유효한 거래와 기록 순서에 대해 **공통의 원장 상태에 합의**하게 한다. 이를 통해 서로 충돌하는 지불이 모두 유효한 거래로 확정되는 것을 막는다.

NoteBase

문 4. 기억장치에 저장될 데이터 또는 기억장치로부터 읽은 데이터가 일시적으로 저장되는 CPU 내부 레지스터는?

① 프로그램 카운터

② 명령어 레지스터

③ 메모리 주소 레지스터

④ 메모리 버퍼 레지스터

정답 4번

간단한 해설

메모리 버퍼 레지스터는 **기억장치와 주고받는 데이터**를 일시적으로 보관한다. 기억장치에 쓸 데이터와 기억장치에서 읽어 온 데이터가 모두 이 역할에 해당한다.

공통 해설

메모리 버퍼 레지스터(MBR)는 CPU와 기억장치 사이에서 전송되는 데이터를 잠시 보관하는 레지스터이다. 기억장치에 쓸 때는 CPU가 보낼 데이터를 담고, 기억장치에서 읽을 때는 읽어 온 데이터를 담는다. 따라서 문제의 두 표현은 데이터 이동 방향만 다를 뿐, 모두 메모리 버퍼 레지스터의 기능을 설명한다.

이때 구별할 기준은 **전송할 데이터 자체**를 보관하는가이다. 메모리 주소 레지스터가 보관하는 것은 데이터를 읽거나 쓸 위치를 나타내는 주소이므로, 데이터 자체를 임시 보관하는 역할과 구별된다.

NoteBase

문 5. 언어 번역 프로그램에 해당하지 않는 것은?

① 어셈블러

② 컴파일러

③ 인터프리터

④ 워드프로세서

정답 4번

간단한 해설

워드프로세서는 **문서를 작성·편집하는 응용 프로그램**이다. 프로그램의 소스 코드를 번역하거나 해석하여 실행하는 언어 번역 프로그램에 해당하지 않는다.

선지별 해설

① O: 어셈블러는 **어셈블리어로 작성된 프로그램을 기계어로 번역**한다. 어셈블리어의 기호로 표현한 명령을 컴퓨터가 처리할 수 있는 명령으로 바꾸므로 언어 번역 프로그램에 해당한다.

② O: 컴파일러는 고급 언어로 작성된 소스 프로그램을 **기계어 등의 목적 프로그램으로 번역**한다. 번역 결과를 만들어 낸 뒤 실행하는 방식이라는 점에서, 명령을 해석하면서 실행하는 인터프리터와 구별된다.

③ O: 인터프리터는 소스 프로그램의 명령을 **해석하면서 실행**한다. 컴파일러처럼 소스 프로그램 전체를 미리 목적 프로그램으로 만들어 두는 방식과는 다르지만, 프로그래밍 언어의 명령을 해석·실행하므로 언어 번역 프로그램에 포함된다.

④ X: 워드프로세서는 글자 입력, 문서 편집, 서식 지정, 인쇄 등을 위한 **응용 프로그램**이다. 처리 대상이 문서이며, 소스 프로그램의 명령을 기계어 등으로 번역하거나 해석·실행하는 언어 번역 프로그램은 아니다.

NoteBase

문 7. 다음 프로세스 메모리 배치에 대한 설명으로 옳은 것만을 모두 고르면?

ㄱ. 전역 변수는 스택(stack) 영역에 할당된다.
ㄴ. 프로그램 코드는 텍스트(text) 영역에 적재된다.
ㄷ. 힙(heap) 영역은 프로세스 실행 중 동적으로 할당된 메모리이다.
ㄹ. 함수의 매개변수, 복귀 주소 및 지역변수는 데이터(data) 영역에 할당된다.

① ㄱ, ㄴ
② ㄱ, ㄹ
③ ㄴ, ㄷ
④ ㄷ, ㄹ

정답 3번

간단한 해설

실행할 프로그램 명령어는 **텍스트 영역**에 적재되고, 실행 중 요청하여 동적으로 확보하는 메모리는 **힙 영역**에 할당된다. 함수 호출에 필요한 정보와 일반적인 지역변수는 스택 영역에 저장된다.

보기별 해설

ㄱ. X: 전역 변수는 함수 호출과 관계없이 프로그램 실행 동안 유지되므로, 스택이 아니라 **데이터 영역**에 할당된다. 스택은 함수 호출에 필요한 정보와 일반적인 지역변수를 저장하는 영역이다.

ㄴ. O: 텍스트 영역은 프로세스가 실행할 **프로그램의 명령어 코드**를 담는 영역이다. 따라서 프로그램 코드가 텍스트 영역에 적재된다는 설명은 이 영역의 역할에 부합한다.

ㄷ. O: 힙 영역은 프로세스 실행 중 필요에 따라 **동적으로 메모리를 할당**하는 데 사용하는 영역이다. 예를 들어 C 언어에서 malloc으로 확보한 메모리 블록은 힙에 놓이며, 더 이상 필요하지 않으면 free로 해제한다.

ㄹ. X: 일반적인 프로세스 메모리 배치에서 함수의 매개변수, 복귀 주소 및 지역변수는 데이터 영역이 아니라 **스택 영역**에 저장된다. 함수 호출마다 이 정보를 담는 스택 프레임이 만들어지며, 함수가 반환되면 해당 프레임이 해제된다. 복귀 주소는 함수 실행을 마친 뒤 돌아갈 명령어의 위치를 나타낸다.

옳은 보기는 ㄴ과 ㄷ이며, 이 조합은 ③에 해당한다.

NoteBase

문 8. 다음 TCP와 UDP의 공통 설명으로 옳은 것만을 모두 고르면?

ㄱ. 연결 지향 프로토콜이다.
ㄴ. 헤더에 근원지(source)와 목적지(destination) 포트 번호가 존재한다.
ㄷ. 네트워크에서 손실된 패킷에 대한 재전송 기능이 있다.
ㄹ. 체크섬(checksum)이 헤더에 존재한다.
ㅁ. 헤더 크기가 동일하다.

① ㄱ, ㄴ
② ㄴ, ㄹ
③ ㄷ, ㄹ
④ ㄹ, ㅁ

정답 2번

간단한 해설

TCP와 UDP는 모두 헤더에 **근원지·목적지 포트 번호**와 오류 검사용 **체크섬 필드**를 둔다. 체크섬 필드가 있다는 사실이 손실된 데이터를 재전송한다는 뜻은 아니다.

NoteBase

보기별 해설

ㄱ. X: TCP는 통신 전에 연결을 설정하는 연결 지향 프로토콜이지만, UDP는 **연결을 설정하지 않는 비연결형** 프로토콜이다. 연결 지향이라는 설명은 TCP에만 해당한다.

ㄴ. O: 두 프로토콜의 헤더에는 모두 **근원지 포트 번호와 목적지 포트 번호** 필드가 있다. 포트 번호는 한 호스트 안에서 통신하는 응용 프로그램이나 서비스를 구분하는 데 쓰인다.

ㄷ. X: TCP는 확인 응답과 재전송 타이머 등을 이용해 손실된 데이터를 재전송하지만, **UDP 자체에는 재전송 기능이 없다**. UDP를 사용하는 응용 프로그램이 필요에 따라 재전송을 구현할 수는 있으나, 이는 UDP가 제공하는 기능이 아니다.

ㄹ. O: 두 프로토콜의 헤더에는 모두 전송 중 데이터의 오류를 검사하기 위한 **체크섬 필드**가 있다. 체크섬은 오류 검출에 쓰이며, 손실 데이터를 다시 보내는 재전송 기능과는 구별된다.

ㅁ. X: UDP 헤더는 **8바이트로 고정**되어 있지만, TCP 헤더는 기본 20바이트이며 옵션에 따라 최대 60바이트까지 늘어날 수 있다. 따라서 두 프로토콜의 헤더 크기가 같다는 설명은 성립하지 않으며, TCP 헤더 크기가 항상 일정한 것도 아니다.

공통 설명에 해당하는 보기는 ㄴ, ㄹ이며, 이 조합은 ②에 해당한다.

NoteBase

문 9. 다음과 같이 접근 시간이 주어졌을 때, 캐시 적중률은? (단, 캐시 적재 시간과 캐시 적중 여부 검사 시간은 제외한다)

◯ 평균 기억장치 접근 시간 = 2.5 ms
◯ 주기억 장치 접근 시간 = 11.5 ms
◯ 캐시 기억장치 접근 시간 = 1.5 ms

① 0.8
② 0.85
③ 0.9
④ 0.95

정답 3번

간단한 해설

평균 접근 시간은 **적중·불적중 확률을 반영한 가중평균**이다. 주어진 접근 시간으로 계산한 캐시 적중률은 0.9, 즉 90%이다.

계산 과정

캐시 적중률을 h라 하면, 요청한 자료가 캐시에 있을 확률은 h이고 없을 확률은 1 − h이다. 모든 접근 시간이 ms로 주어졌으므로 단위 변환은 필요 없다.

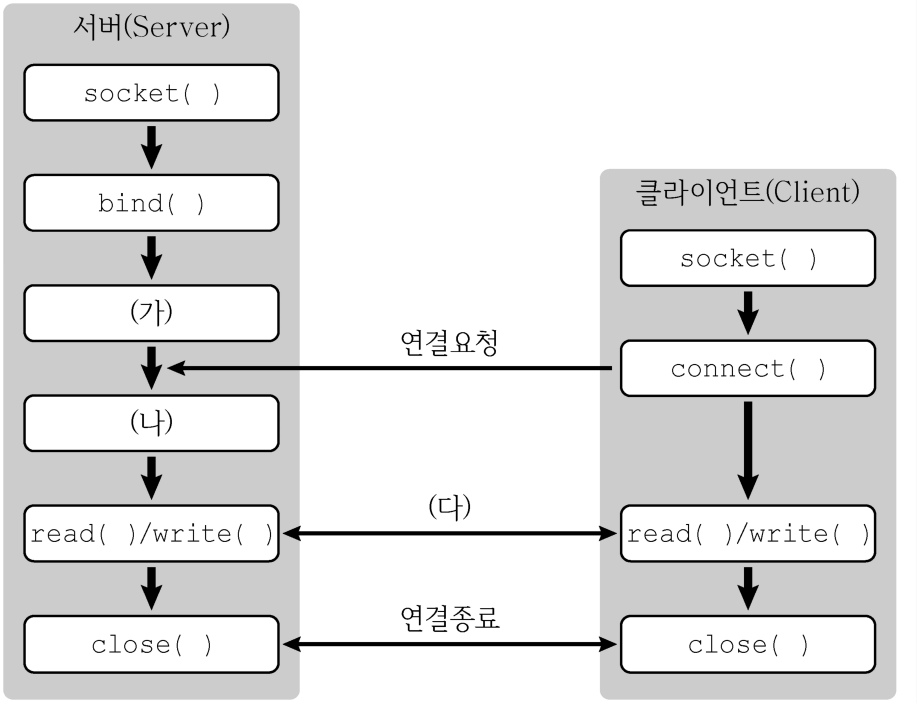
문제에서 캐시 적재 시간과 적중 여부 검사 시간을 제외하므로, 적중하면 캐시 접근 시간 1.5 ms를, 불적중하면 주기억 장치 접근 시간 11.5 ms를 적용한다. 각 경우의 시간에 **발생 확률을 곱해 더하면** 평균 접근 시간 2.5 ms가 된다. 따라서 $2.5 = 1.5h + 11.5(1 - h)$이다.

주기억 장치에 접근하는 경우를 기준으로 보면, 캐시에 적중할 때 줄어드는 시간은 11.5 − 1.5 = 10 ms이다. 실제 평균 접근 시간은 이 기준보다 11.5 − 2.5 = 9 ms 짧다. 따라서 **한 번 적중할 때의 절감 시간에 적중률을 곱한 값**이 평균 절감 시간과 같으므로 $10h = 9$, $h = \frac{9}{10} = 0.9$이다.

캐시 적중률은 단위가 없는 비율로 0.9이며, 백분율로는 90%이다. 이는 ③의 0.9에 해당한다.

NoteBase

문 10. 다음 그림은 TCP 기반 서버와 클라이언트 간 소켓 통신을 위한 함수 호출 관계를 나타낸다. (가) ~ (다)에 들어갈 내용을 바르게 연결한 것은?



① (가): request(), (나): accept(), (다): 연결 종료 요청

② (가): accept(), (나): connect(), (다): 연결 종료 요청

③ (가): listen(), (나): accept(), (다): 데이터 송수신

④ (가): request(), (나): connect(), (다): 데이터 송수신

정답 3번

간단한 해설

서버는 **listen()으로 연결 요청을 받을 준비**를 하고, accept()로 접수한 연결에 사용할 소켓을 얻는다. 연결된 뒤 read()/write() 사이의 양방향 화살표는 **데이터 송수신**을 뜻한다.

NoteBase

빈칸별 해설

(가) listen(): 서버의 bind() 다음에 위치한다. bind()는 소켓에 서버의 주소와 포트를 연결하고, listen()은 이 소켓을 **연결 요청을 받는 대기 상태**로 만든다. 클라이언트가 connect()로 요청하기 전에 서버가 연결을 받아들일 준비를 하는 단계이다.

(나) accept(): 클라이언트의 connect()에서 온 연결 요청을 서버가 접수하는 단계이다. accept()는 대기 중인 연결 요청을 받아 **해당 클라이언트와 통신할 새 소켓**을 반환한다. connect()가 클라이언트 쪽에서 연결을 요청하는 함수라면, accept()는 서버 쪽에서 그 요청을 받아들이는 함수이다.

(다) 데이터 송수신: 그림에서 서버와 클라이언트의 read()/write()를 잇는 양방향 화살표이다. 연결된 소켓에서 write()로 데이터를 보내고 read()로 상대가 보낸 데이터를 읽는다. **연결 종료는 아래쪽 close() 단계**에 따로 표시되어 있으므로, 이 화살표는 연결 종료 요청이 아니라 데이터 교환을 나타낸다.

(가)는 listen(), (나)는 accept(), (다)는 데이터 송수신으로 연결되며, 이 조합은 ③과 일치한다.

NoteBase

문 11. 프로세스 간 통신에 대한 설명으로 옳지 않은 것은?

① 파이프(pipe)는 두 프로세스가 통신할 수 있게 하는 전달자로서 동작한다.

② 지명 파이프(named pipe)는 서로 통신하는 프로세스 간 부모-자식 관계가 반드시 필요하다.

③ 신호(signal)는 알려 줄 사건의 근원지나 이유에 따라 비동기식으로도 전달될 수 있다.

④ 공유 메모리를 사용하는 프로세스 간 통신에서 프로세스들은 동시에 동일한 위치에 쓰지 않도록 책임져야 한다.

정답 2번

간단한 해설

지명 파이프는 **부모-자식 관계가 없는 프로세스 사이에서도** 통신할 수 있다. 파일 시스템에 부여된 이름으로 같은 파이프에 접근하므로, 부모-자식 관계가 반드시 필요하다는 설명은 잘못되었다.

선지별 해설

① O: 파이프는 한 프로세스가 쓴 데이터를 다른 프로세스가 읽을 수 있게 하는 **프로세스 간 데이터 전달 통로**이다. 예를 들어 한 프로세스가 파이프에 문자 A를 쓰면, 다른 프로세스가 그 파이프에서 A를 읽어 전달받는다.

② X: 지명 파이프는 파일 시스템에 이름을 가진 통신 통로이다. 접근 권한이 있는 프로세스는 그 이름으로 같은 파이프를 열어 통신할 수 있으므로, **부모-자식 관계가 반드시 필요한 것은 아니다**. 서로 독립적으로 실행된 프로세스 사이에서도 사용할 수 있다.

③ O: 신호는 프로세스에 특정 사건의 발생을 알리는 통지 수단이다. 다른 프로세스가 보내는 신호나 타이머 만료에 따른 신호는 수신 프로세스의 **현재 실행 흐름과 별개로 비동기적으로** 전달될 수 있다. 반면 잘못된 메모리 접근처럼 현재 실행 중인 명령에서 발생한 사건에 따른 신호는 동기적으로 발생할 수 있다.

④ O: 공유 메모리에서는 여러 프로세스가 같은 메모리 영역에 직접 접근한다. 이때 동시에 같은 위치를 갱신하면 한 프로세스의 값이 다른 프로세스의 값으로 덮어써지는 등 경쟁 상태가 발생할 수 있다. 공유 메모리 자체가 접근 순서를 자동으로 보장하지 않으므로, 프로세스들은 뮤텍스나 세마포어 등을 이용한 **상호 배제**로 쓰기 작업을 조정해야 한다.

NoteBase

문 12. 파이프라이닝 기법에 대한 설명으로 옳지 않은 것은?

① 데이터 해저드는 앞의 명령 결과가 다음 명령의 입력으로 사용될 때 발생한다.

② 이론적으로는 파이프라인의 단계 수만큼 빨라지지만, 이것을 방해하는 요소인 해저드가 발생할 수 있다.

③ 파이프라이닝은 명령어 하나를 여러 단계로 나누어 각각을 독립적인 장치에서 동시에 실행하는 기법이다.

④ 제어 해저드는 서로 다른 단계에서 실행되는 명령이 컴퓨터 내의 장치 하나를 동시에 사용하려고 할 때 발생한다.

정답 4번

간단한 해설

④의 '장치 하나를 동시에 사용하려고 할 때' 발생하는 것은 **구조적 해저드**이다. **제어 해저드**는 분기 등으로 다음에 실행할 명령어가 확정되지 않아 발생한다.

선지별 해설

① O: 앞 명령어의 결과를 뒤 명령어가 입력으로 사용하는데, **필요한 시점까지 결과가 준비되지 않으면** 데이터 해저드가 발생한다. 예를 들어 앞 명령어가 계산한 레지스터 값을 바로 다음 명령어가 읽어야 하는 경우이다. 이때 결과 전달이나 대기 처리가 없으면 뒤 명령어가 이전 값을 읽을 수 있다.

② O: 각 단계의 처리 시간이 같고 해저드와 추가 지연이 없는 이상적인 조건에서, 충분히 많은 명령어를 처리하면 성능 향상은 파이프라인의 단계 수에 가까워진다. 이는 **여러 명령어를 겹쳐 처리하여 단위 시간당 완료하는 명령어 수를 늘리기** 때문이다. 실제로는 해저드로 인한 대기 등으로 이론적인 성능 향상에 미치지 못한다.

③ O: 명령어 처리를 인출·해독·실행 등의 단계로 나누고, 단계별 장치를 이용하여 **서로 다른 명령어의 서로 다른 단계를 동시에 처리**한다. 예를 들어 앞 명령어를 해독하는 동안 다음 명령어를 인출할 수 있다. 각 명령어 자체는 필요한 단계를 순서대로 거친다.

④ X: 서로 다른 명령어가 같은 시점에 하나의 하드웨어 자원을 필요로 하여 충돌하는 것은 **구조적 해저드**이다. 예를 들어 단일 포트 메모리를 명령어 인출과 데이터 접근이 동시에 사용하려는 경우가 이에 해당한다. 제어 해저드는 분기 명령어 등의 처리로 다음에 인출할 명령어 주소가 확정되지 않거나 달라져 발생한다.

NoteBase

문 13. 원형 연결 리스트에 대한 설명으로 옳은 것은?

① 하나의 노드에서 모든 노드로의 접근이 가능하다.

② 하나의 노드에서 링크를 계속 따라가면 모든 노드를 거쳐도 자기 자신으로 되돌아올 수 없다.

③ 마지막 노드의 링크 필드 값은 NULL이다.

④ 리스트의 끝에 노드를 삽입하는 연산이 단순 연결 리스트보다 비효율적이다.

정답 1번

간단한 해설

원형 연결 리스트는 **마지막 노드의 링크가 첫 노드를 가리키는** 구조이다. 따라서 어느 노드에서 출발해도 링크를 따라 모든 노드에 접근하고 출발한 노드로 돌아올 수 있다.

선지별 해설

① O: 원형 연결 리스트에서는 마지막 노드가 첫 노드로 연결되므로 모든 노드가 하나의 고리를 이룬다. 따라서 **어느 노드에서 출발해도** 링크를 따라 리스트의 모든 노드에 접근할 수 있다.

② X: 링크를 계속 따라가면 모든 노드를 거친 뒤 **출발한 노드로 되돌아온다**. 예를 들어 A → B → C → A로 연결되어 있다면 B에서 출발해도 B → C → A → B로 순환한다.

③ X: 원형 연결 리스트의 마지막 노드 링크는 NULL이 아니라 **첫 노드의 주소**를 저장한다. 마지막 노드의 링크가 NULL인 것은 일반적인 비원형 단순 연결 리스트의 특징이다.

④ X: 끝에 삽입하는 연산이 원형 연결 리스트라는 이유로 더 비효율적인 것은 아니다. **마지막 노드를 가리키는 포인터를 유지하면** 그 노드의 링크로 첫 노드에도 바로 접근할 수 있다. 새 노드의 링크를 첫 노드로, 기존 마지막 노드의 링크를 새 노드로 연결하고 마지막 노드 포인터를 갱신하면 전체 순회 없이 끝에 삽입할 수 있다. 비원형 단순 연결 리스트도 마지막 노드 포인터를 유지하면 끝에 바로 삽입할 수 있으므로, 삽입 효율을 비교할 때는 포인터를 어떻게 유지하는지 함께 보아야 한다.

NoteBase

문 14. 다음 파이썬(Python) 프로그램의 출력 결과는?

```
a = [1, 3, 5, 7]
b = {1, 3, 5, 7}
c = (1, 3, 5, 7)

a.append(3)
b.add(3)

print(len(a) + len(b) + len(c))
```

① 10
② 12
③ 13
④ 14

정답 3번

간단한 해설

리스트는 **중복을 허용**하므로 3을 추가하면 원소가 5개가 되지만, 집합은 **중복을 허용하지 않아** 원소가 4개로 유지된다. 튜플의 원소도 4개이므로 출력값은 5 + 4 + 4 = 13이다.

NoteBase

계산 과정

a의 길이: len은 자료에 들어 있는 원소의 개수를 구한다. a는 처음에 원소가 4개인 리스트이다. append(3)은 **리스트 끝에 원소 하나를 추가**하므로, 이미 3이 있어도 a는 [1, 3, 5, 7, 3]이 된다. 따라서 len(a)는 5이다.

b의 길이: b는 {1, 3, 5, 7}인 집합이다. 집합은 **같은 값을 중복하여 저장하지 않으므로** b.add(3)을 실행해도 이미 있는 3이 하나 더 추가되지 않는다. 따라서 b는 그대로이고 len(b)는 4이다.

c의 길이: c는 (1, 3, 5, 7)인 튜플이다. 프로그램에는 c를 변경하거나 다른 값으로 재대입하는 문장이 없으므로 len(c)는 처음과 같은 4이다.

출력값: print는 세 자료의 원소 개수를 더한 값을 출력한다. len(a) + len(b) + len(c)는 5 + 4 + 4 = 13이므로, 출력 결과는 ③의 13과 일치한다.

NoteBase

문 15. 외부 인터럽트(External Interrupt)의 요인에 해당하지 않는 것은?

① 입·출력

② 오버플로

③ 전원 이상

④ 기계 오작동

정답 2번

간단한 해설

오버플로는 연산 결과가 표현 가능한 수의 범위를 벗어나는 현상이다. 이를 검출해 발생하는 인터럽트는 **명령어 실행에 따른 내부 인터럽트**로 분류한다.

선지별 해설

외부 인터럽트는 실행 중인 명령어의 연산과 직접 관계없는 입·출력 요청이나 하드웨어 이상 등으로 발생한다. 내부 인터럽트는 명령어 실행 중 검출되는 예외 상황으로 발생한다.

① O: 입·출력 장치는 데이터 전송이 완료되거나 처리가 필요할 때 CPU에 인터럽트를 요청할 수 있다. 이는 실행 중인 명령어의 연산 오류가 아니라 **입·출력 장치의 요청**으로 발생하므로 외부 인터럽트에 해당한다.

② X: 오버플로는 산술 연산의 결과가 해당 자료형으로 표현할 수 있는 범위를 벗어나는 현상이다. 이를 검출해 발생하는 인터럽트는 **명령어 실행 중의 연산 예외**이므로 외부 인터럽트가 아니라 내부 인터럽트에 해당한다.

③ O: 전원 이상은 정전이나 공급 전압의 이상처럼 정상적인 동작을 위협하는 상태이다. 명령어의 연산 내용과 관계없이 발생하는 **전원 상태의 이상**을 알리는 인터럽트이므로 외부 인터럽트에 해당한다.

④ O: 기계 오작동은 하드웨어 고장이나 오류로 장치가 정상적으로 동작하지 않는 상황이다. 명령어 자체의 연산 예외와 구별되는 **하드웨어 이상**을 알리는 인터럽트이므로 외부 인터럽트에 해당한다.

NoteBase

문 16. 컴퓨터의 버스에 대한 설명으로 옳지 않은 것은?

① 시스템 버스의 종류에는 제어 버스, 데이터 버스, 주소 버스가 있다.

② 제어 버스는 제어 신호를 전달하기 위해 사용된다.

③ 버스의 폭은 전송량을 의미하며, 컴퓨터의 성능에 영향을 준다.

④ 컴퓨터의 각 장치 간에 데이터를 전달할 때 사용하는 데이터 버스는 단방향성을 가지고 있다.

정답 4번

간단한 해설

데이터 버스는 데이터를 읽어 올 때와 내보낼 때 모두 사용되므로 **양방향**으로 데이터를 전달한다. ④의 '단방향성'은 '양방향성'으로 고쳐야 한다.

선지별 해설

① O: 시스템 버스는 CPU·메모리·입출력장치 사이의 통신 경로이다. 전달하는 정보에 따라 **데이터 버스·주소 버스·제어 버스**로 구분한다. 각각 실제 데이터, 접근할 위치의 주소, 동작을 지시하거나 상태를 알리는 신호를 전달한다.

② O: 제어 버스는 장치 사이에서 동작을 조정하는 **제어 신호**를 전달한다. 예를 들어 CPU가 메모리의 데이터를 읽으려면 읽기 신호로 수행할 동작을 알려야 한다. 메모리 읽기·쓰기 신호와 인터럽트 요청 신호 등이 이에 해당한다.

③ O: 버스의 폭은 **한 번에 병렬로 전달할 수 있는 비트 수**를 뜻한다. 예를 들어 32비트 데이터 버스는 한 번에 32비트를 전달할 수 있다. 같은 전송 주기라면 폭이 넓을수록 한 번에 더 많은 데이터를 옮길 수 있으므로 컴퓨터의 성능에 영향을 준다. 다만 실제 전송 속도는 버스의 동작 주파수 등에도 영향을 받는다.

④ X: 데이터 버스는 **양방향성**을 가진다. CPU가 메모리를 읽을 때는 데이터가 메모리에서 CPU로 이동하고, 메모리에 쓸 때는 CPU에서 메모리로 이동한다. 따라서 데이터를 한쪽 방향으로만 전달한다는 설명은 잘못이다.

NoteBase

문 17. NoSQL의 특징으로 옳지 않은 것은?

① 동적 스키마를 지원할 수 있다.

② 다양한 형태의 비정형 데이터들도 다룬다.

③ 빠른 속도로 증가하는 대용량 데이터를 처리하는 데 적합하지 않다.

④ 관계형 데이터베이스보다는 ACID 특성을 중요시하지 않는다.

정답 3번

간단한 해설

NoSQL은 데이터를 여러 서버에 분산하고 서버를 늘려 처리 용량을 확장할 수 있어, **빠르게 증가하는 대용량 데이터 처리에 적합하다**. 따라서 '적합하지 않다'는 설명은 NoSQL의 특징과 반대이다.

선지별 해설

① O: 동적 스키마는 저장할 데이터의 구조를 유연하게 바꿀 수 있다는 뜻이다. 예를 들어 문서형 NoSQL에서는 한 문서에 이름과 연락처를, 다른 문서에는 이름과 주소를 저장하는 식으로 **문서마다 필드를 다르게 구성**할 수 있다.

② O: NoSQL은 고정된 행과 열로 이루어진 관계형 테이블 외에 문서, 키-값, 그래프 등 다양한 데이터 모델을 사용한다. 이를 통해 **구조가 일정하지 않은 데이터**도 저장하고 다룰 수 있어, 반정형·비정형 데이터 처리에 활용된다.

③ X: NoSQL은 데이터를 여러 서버에 분산해 저장·처리하고, 서버 수를 늘리는 **수평적 확장**을 통해 처리 용량을 키울 수 있다. 따라서 빠르게 증가하는 대용량 데이터 처리에 '적합하지 않다'가 아니라 '적합하다'로 고쳐야 한다.

④ O: ACID는 트랜잭션의 원자성·일관성·격리성·지속성을 뜻한다. 전통적인 관계형 데이터베이스가 이 특성을 중시하는 데 비해, NoSQL은 일반적으로 확장성과 가용성을 확보하기 위해 일부 보장 범위를 완화하기도 한다. 다만 **모든 NoSQL이 ACID를 지원하지 않는다는 뜻은 아니다**. 제품과 기능에 따라 ACID 트랜잭션을 지원할 수 있다.

NoteBase

문 18. 힙(Heap)에 대한 설명으로 옳지 않은 것은?

① 삽입 시간 복잡도는 $O(1)$이다.

② 힙은 우선순위 큐의 한 종류이다.

③ 힙은 완전 이진 트리를 사용한다.

④ 최대 힙(Max Heap)은 부모노드의 키값이 자식노드의 키값보다 크거나 같다.

정답 1번

간단한 해설

힙의 삽입은 새 원소를 마지막 자리에 넣은 뒤, 힙의 대소 관계를 만족할 때까지 **부모와 비교하며 위로 이동**시키는 연산이다. 원소 수가 n일 때 루트까지 이동할 수 있으므로 최악의 시간 복잡도는 $O(\log n)$이다.

선지별 해설

① X: 원소 수가 n인 이진 힙에서 삽입의 **최악 시간 복잡도**는 $O(\log n)$이다. 새 원소를 완전 이진 트리의 마지막 자리에 넣는 것만으로 끝나지 않고, 부모와 비교하여 힙의 대소 관계가 어긋나면 자리를 바꾸며 올라간다. 한 번에 한 레벨씩 최대 루트까지 이동하며, 완전 이진 트리의 높이가 $O(\log n)$이므로 삽입을 $O(1)$이라고 할 수 없다.

② O: 우선순위 큐는 먼저 들어온 순서보다 **우선순위에 따라 원소를 꺼내는** 자료구조이다. 힙은 이를 구현하는 대표적인 자료구조로, 최대 힙은 키값이 가장 큰 원소를, 최소 힙은 키값이 가장 작은 원소를 루트에 두어 우선적으로 꺼낼 수 있게 한다.

③ O: 이 문제의 이진 힙은 **완전 이진 트리** 형태를 유지한다. 마지막 레벨을 제외한 모든 레벨이 가득 차 있고, 마지막 레벨의 노드는 왼쪽부터 빈자리 없이 채워진다. 삽입할 때도 이 형태를 유지하도록 다음 빈자리에 새 원소를 놓는다.

④ O: 최대 힙에서는 모든 부모·자식 관계에서 **부모의 키값이 자식의 키값보다 크거나 같다**. 이 관계가 루트부터 아래까지 유지되므로 루트에는 전체 원소 중 최댓값이 놓인다. 부모와 자식의 키값이 같은 경우도 허용된다.

NoteBase

문 19. 운영체제 구조에 대한 설명으로 옳지 않은 것은?

① 커널은 운영체제의 핵심 기능을 모아 놓은 것이다.

② 다양한 장치를 효율적으로 사용할 수 있는 하드웨어와의 인터페이스를 디바이스 드라이버라고 한다.

③ 사용자 인터페이스는 커널에 명령을 전달하고, 실행 결과를 사용자와 응용 프로그램에 전달한다.

④ 사용자나 응용 프로그램이 커널을 거치지 않고 하드웨어에 직접 접근할 수 있도록 제공하는 함수의 집합을 시스템 호출이라고 한다.

정답 4번

간단한 해설

시스템 호출은 사용자나 응용 프로그램이 **커널의 서비스를 요청**하는 인터페이스이다. ④의 '커널을 거치지 않고 하드웨어에 직접 접근'한다는 설명은 잘못되었으며, 시스템 호출을 통해 커널이 요청을 처리한다.

선지별 해설

① O: 커널은 프로세스 관리, 메모리 관리, 입출력 관리 등 **운영체제의 핵심 기능**을 수행하는 부분이다. 여러 프로그램이 CPU와 메모리, 입출력 장치를 사용할 수 있도록 자원을 관리한다.

② O: 디바이스 드라이버는 운영체제가 특정 하드웨어 장치를 제어할 수 있도록 하는 소프트웨어이다. 장치마다 다른 동작 방식에 맞게 **입출력 요청을 장치 제어 명령으로 연결**하므로 운영체제와 하드웨어 사이의 인터페이스 역할을 한다.

③ O: 사용자 인터페이스는 사용자와 운영체제 사이에서 **명령과 실행 결과를 주고받는 창구**이다. 사용자의 명령을 받아 커널이 처리하도록 전달하고, 처리 결과를 사용자에게 보여 준다. 응용 프로그램에서 발생한 요청과 그 실행 결과도 운영체제의 인터페이스를 통해 전달된다.

④ X: 시스템 호출은 응용 프로그램이 파일 읽기, 프로세스 생성 등 **커널의 서비스를 요청하는 인터페이스**이다. 예를 들어 파일 읽기를 요청하면 커널이 접근 권한 등을 확인하고 필요한 입출력을 수행한 뒤 결과를 돌려준다. 따라서 '커널을 거치지 않고 하드웨어에 직접 접근'하는 것이 아니라, 커널을 거쳐 운영체제가 제공하는 기능을 이용하는 것이다.

NoteBase

문 20. 다음 제시된 표에서 (가) ~ (라)에 들어갈 애플리케이션 계층 프로토콜 및 하위 전송 계층 프로토콜을 바르게 연결한 것은?

프로토콜 / 애플리케이션	애플리케이션 계층 프로토콜	하위 전송 계층 프로토콜
전자메일	(가)	(나)
원격접속	(다)	(라)

① (가): SMTP, (나): TCP, (다): Telnet, (라): TCP

② (가): SNMP, (나): UDP, (다): DNS, (라): UDP

③ (가): SNMP, (나): TCP, (다): Telnet, (라): UDP

④ (가): SMTP, (나): UDP, (다): DNS, (라): TCP

정답 1번

간단한 해설

전자메일 전송에 쓰는 SMTP와 원격접속에 쓰는 Telnet은 모두 **TCP의 신뢰성 있는 바이트 스트림**을 이용한다.

빈칸별 해설

(가) SMTP: 전자메일을 보내거나 메일 서버 사이에서 전달할 때 사용하는 애플리케이션 계층 프로토콜이다. SNMP는 **네트워크 장비의 상태를 조회하고 관리**하는 프로토콜이므로 전자메일 전송에 대응하지 않는다.

(나) TCP: SMTP는 **TCP 연결 위에서 명령과 메일 데이터**를 주고받는다. TCP는 순서 제어와 손실 데이터의 재전송 등을 통해 신뢰성 있는 전송을 제공한다. 따라서 전자메일 행의 하위 전송 계층에는 UDP가 아니라 TCP가 들어간다.

(다) Telnet: 원격 컴퓨터에 접속하여 명령을 입력하고 실행 결과를 받는 원격 터미널 접속 프로토콜이다. DNS는 **도메인 이름에 대응하는 IP 주소 등의 정보**를 조회하는 프로토콜이므로 원격접속에 대응하지 않는다.

(라) TCP: Telnet은 **TCP 연결을 통해 터미널 데이터**를 주고받는다. 입력한 문자와 응답이 순서대로 전달되어야 하므로 TCP의 신뢰성 있는 바이트 스트림을 이용한다.

따라서 (가) SMTP, (나) TCP, (다) Telnet, (라) TCP로 연결되며, 이는 ①의 조합과 일치한다.

NoteBase