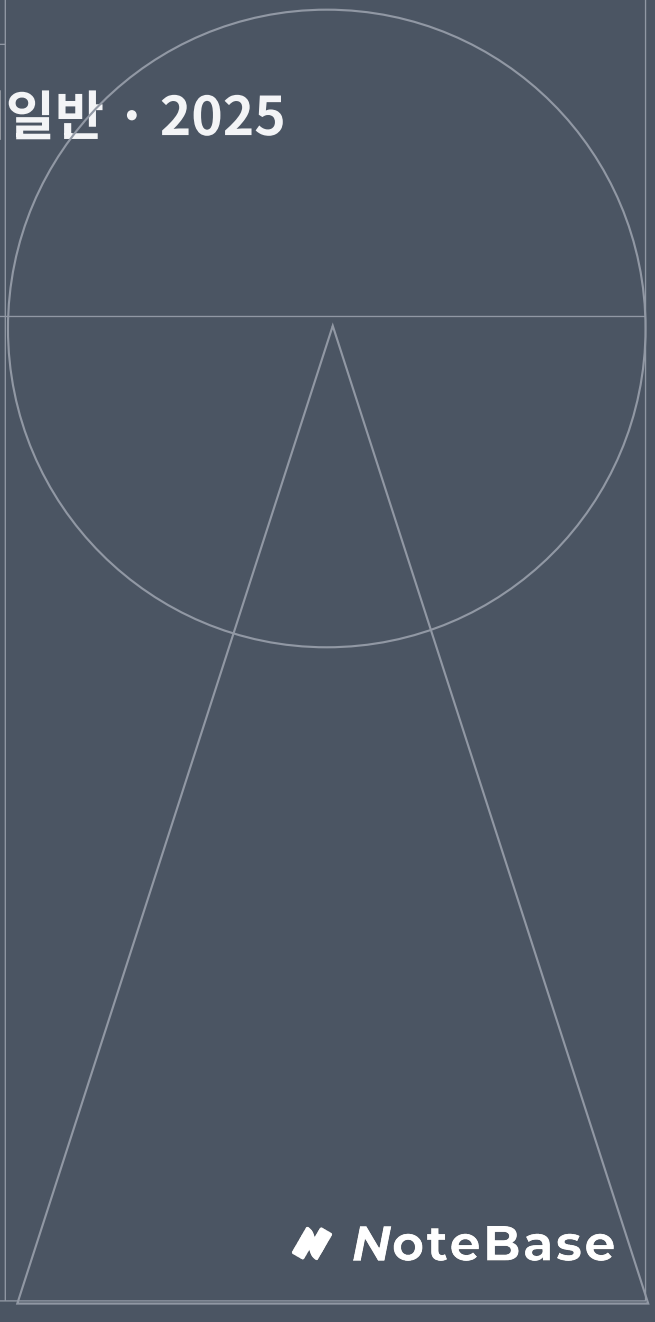
기출OX 해설집

공무원/국회직/9급 · 컴퓨터일반 · 2025



NoteBase

기출OX 해설집

공무원/국회직/9급 · 컴퓨터일반 · 2025

공무원/국회직/9급 · 컴퓨터일반 · 2025

1. 다음 전위 표기 식을 후위 표기 식으로 변환한 것으로 옳은 것은?

+ − + a * e f / b c d

① a e f * + b − c / d +
② a e f * + b c / d + −
③ a e f * + b c / − d +
④ a e + f * b / − c d +
⑤ a e + f * − b / c d +

정답 3번

출제 포인트

전위 표기 식을 연산자의 우선순위와 피연산자 결합 관계를 보존하여 후위 표기 식으로 변환할 수 있는지를 묻는 문제이다.

올바른 순서

주어진 전위식은 + - + a * e f / b c d이다.
1. * e f를 e f *로 변환한다.
2. + a * e f를 a e f * +로 변환한다.
3. / b c를 b c /로 변환한다.
4. - (+ a * e f) (/ b c)를 a e f * + b c / -로 변환한다.
5. 맨 바깥의 +와 오른쪽 피연산자 d를 반영하면 a e f * + b c / - d +가 된다.

왜 이 순서: 후위 표기에서는 각 연산자가 자신의 두 피연산자 뒤에 와야 하므로 안쪽 부분식을 먼저 완성한 뒤 바깥 연산자를 붙인다.

NoteBase

선지별 해설

① X
- a e f * +까지는 맞지만, 이어지는 b - c /는 b와 c의 나눗셈을 먼저 나타내지 못한다.
- 틀린 부분: b - c /
- 올바르게는: b c / -로 고쳐 전체 식을 a e f * + b c / - d +로 써야 한다.

② X
- 마지막의 -는 d까지 오른쪽 피연산자에 포함시키는 구조가 되어 원래 전위식의 최상위 연산자 +를 보존하지 못한다.
- 틀린 부분: b c / d + -
- 올바르게는: b c / - d +로 고쳐 전체 식을 a e f * + b c / - d +로 써야 한다.

③ O
- a e f * +는 a + (e * f), b c /는 b / c를 나타낸다. 두 결과에 -를 적용한 뒤 d와의 최상위 +를 마지막에 배치했으므로 a e f * + b c / - d +가 정확하다.

④ X
- a e + f *는 (a + e) * f를 뜻하므로 원래의 a + (e * f)와 결합 순서가 다르다. 또한 뒤쪽 연산자와 피연산자의 배치도 원래 구조와 일치하지 않는다.
- 틀린 부분: a e + f *
- 올바르게는: a e f * +로 고치고, 전체 식은 a e f * + b c / - d +로 써야 한다.

⑤ X
- a e + f *는 (a + e) * f를 뜻하고, 이어지는 연산 순서도 원래 전위식의 부분식 구조를 보존하지 못한다.
- 틀린 부분: a e + f * - b / c d +
- 올바르게는: a e f * + b c / - d +로 써야 한다.

오답 포인트

전위식의 연산자를 단순히 뒤로 옮기면 피연산자의 결합 범위가 달라진다. 특히 * e f와 / b c를 각각 하나의 완성된 부분식으로 처리하고, 그다음 -와 맨 바깥 +를 차례로 붙여야 한다.

다시 나오면 이렇게 판단

전위식을 왼쪽부터 읽어 각 연산자가 필요로 하는 두 피연산자 또는 두 부분식을 먼저 확정한다. 각 부분식은 '왼쪽 피연산자, 오른쪽 피연산자, 연산자' 순서로 바꾸고, 가장 바깥 연산자는 마지막에 둔다.

NoteBase

암기 포인트

전위는 연산자가 앞, 후위는 연산자가 뒤이며 피연산자의 결합 관계는 그대로 유지한다.

NoteBase

2. 다음 C언어 프로그램의 출력 결과는?

```
#include <stdio.h>

int x = 5;

void func(int a) {
    a += x++;
    printf("func: %d\n", a);
}

int main() {
    int x = 2;
    func(x++);
    printf("main: %d\n", x);
    return 0;
}
```

① func: 7
main: 3

② func: 7
main: 2

③ func: 8
main: 3

④ func: 8
main: 2

⑤ func: 8
main: 1

정답 1번

출제 포인트

이 문제는 전역변수와 지역변수의 유효 범위, 함수 인수의 값 전달, 후위 증가 연산자 *x* + +의 동작을 순서대로 추적하여 출력값을 구하는 문제이다.

NoteBase

조건 정리

전역변수 *x*의 초깃값은 5이다.
main 함수 안의 지역변수 *x*의 초깃값은 2이며, 이 변수는 main 함수 안에서 같은 이름의 전역변수를 가린다.
func 함수의 매개변수 *a*는 전달받은 값을 복사한 별도의 지역변수이다.
func 함수 안에는 지역변수 *x*가 없으므로 그 안의 *x*는 전역변수이다.

사용할 원리

후위 증가식 *x* + +는 식의 값으로 증가 전 값을 사용하고, 그 뒤 해당 변수의 값을 1 증가시킨다. 또한 C 언어의 함수 인수는 값으로 전달되므로 func의 *a*를 바꾸어도 main의 지역변수 *x*가 그 변경을 되돌려 받지는 않는다.

계산 과정

1. main에서 func(x++)를 실행하면 후위 증가이므로 인수로 2가 전달되고, main의 지역변수 *x*는 3이 된다. 따라서 func의 *a*는 2이다.
2. func의 a += x++에서 *x*는 전역변수이다. 전역변수 *x*의 증가 전 값 5를 더하므로 $a = 2 + 5 = 7$이 되고, 전역변수 *x*는 6이 된다.
3. func는 func: 7을 출력한다. 함수가 끝난 뒤 main의 지역변수 *x*는 3이므로 main은 main: 3을 출력한다.

선지 판단

① O — func: 7, main: 3은 위 실행 순서와 일치한다.
② X — ‘main: 2’가 틀렸다. func(x++)에서 main의 지역변수 *x*가 2에서 3으로 증가하므로 올바른 출력은 func: 7, main: 3이다.
③ X — ‘func: 8’이 틀렸다. func에 전달되는 값은 증가 전 값 2이고, 전역변수 *x*도 증가 전 값 5가 더해지므로 *a*는 7이다. 올바른 출력은 func: 7, main: 3이다.
④ X — ‘func: 8’과 ‘main: 2’가 모두 틀렸다. 각 후위 증가식은 증가 전 값을 연산에 사용한 뒤 실제 변수는 1 증가시키므로 올바른 출력은 func: 7, main: 3이다.
⑤ X — ‘func: 8’과 ‘main: 1’이 모두 틀렸다. main의 지역변수 *x*는 감소하지 않고 2에서 3으로 증가하며, func의 *a*는 7이 된다. 올바른 출력은 func: 7, main: 3이다.

오답 포인트

후위 증가를 ‘증가하지 않는다’고 해석하거나 증가된 값을 즉시 식에 넣으면 틀리기 쉽다. 또한 main의 지역변수 *x*와 전역변수 *x*는 이름만 같을 뿐 서로 다른 변수이다. func(x++)는 main의 지역변수를 3으로 만들고, func 안의 x++는 전역변수를 6으로 만든다.

다시 나오면 이렇게 판단

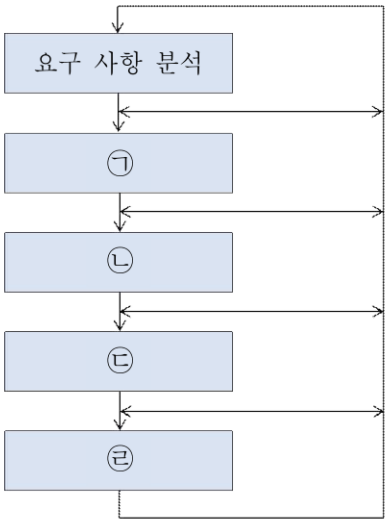
먼저 각 식별자가 가리키는 전역변수·지역변수를 구분한 뒤, 후위 증가식은 ‘현재 값을 사용하고 나서 1 증가’로 나누어 기록한다. 함수 호출에서는 전달되는 값과 호출자 변수에 남는 값을 따로 추적한다.

NoteBase

암기 포인트

후위 증가 *x* + +는 선사용·후증가이고, 같은 이름의 지역변수는 해당 범위에서 전역변수를 가린다.

3. 다음 데이터베이스 설계 과정에서, ㉠~㉤에 들어갈 것으로 옳은 것은?



	㉠	㉡	㉢	㉣
①	논리적 설계	개념적 설계	구현	물리적 설계
②	개념적 설계	물리적 설계	구현	튜닝
③	논리적 설계	개념적 설계	물리적 설계	구현
④	개념적 설계	논리적 설계	물리적 설계	구현
⑤	개념 스키마 정의	관계형 모델 설계	구현	백업 구성

정답 4번

출제 포인트

이 문제는 데이터베이스 설계 과정에서 요구 사항 분석 다음에 이어지는 개념적 설계, 논리적 설계, 물리적 설계, 구현의 순서를 묻는 문제이다.

올바른 순서

1. 요구 사항 분석 — 사용자의 요구와 처리할 데이터를 파악한다.
2. 개념적 설계(㉠) — 특정 DBMS와 무관하게 개체, 속성, 관계를 표현한 개념 스키마를 만든다.
3. 논리적 설계(㉡) — 개념 스키마를 관계형 모델 등 선택한 데이터 모델의 논리 스키마로 변환한다.
4. 물리적 설계(㉢) — 저장 구조, 접근 경로, 인덱스 등 실제 저장 방법을 결정한다.
5. 구현(㉣) — 선택한 DBMS에서 스키마를 정의하고 데이터베이스를 구축한다.

이 순서가 되는 이유는 앞 단계의 산출물이 다음 단계의 입력이 되기 때문이다. 요구 사항을 개념 구조로 정리한 뒤 논리 구조로 변환하고, 그 논리 구조에 맞는 물리적 저장 방식을 정해야 실제 DBMS에 구현할 수 있다.

NoteBase

선지별 해설

① X — ㉠의 '논리적 설계'와 ㉡의 '개념적 설계'가 뒤바뀌었고, ㉢과 ㉣도 '구현'과 '물리적 설계'가 뒤바뀌었다. 올바르게는 ㉠ 개념적 설계, ㉡ 논리적 설계, ㉢ 물리적 설계, ㉣ 구현이다.

② X — ㉡을 물리적 설계로 두어 논리적 설계를 누락했으며, ㉣의 튜닝은 구현 이후 운영·성능 개선 활동이지 도식의 마지막 설계 단계가 아니다. 올바르게는 ㉠ 개념적 설계, ㉡ 논리적 설계, ㉢ 물리적 설계, ㉣ 구현이다.

③ X — ㉠과 ㉡의 순서가 뒤바뀌었다. DBMS에 맞는 논리 스키마를 만들기 전에 DBMS와 독립적인 개념 스키마를 먼저 설계한다. 올바르게는 ㉠ 개념적 설계, ㉡ 논리적 설계, ㉢ 물리적 설계, ㉣ 구현이다.

④ O — 요구 사항 분석 다음에 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현이 이어지는 올바른 순서이다.

⑤ X — '개념 스키마 정의'와 '관계형 모델 설계'는 각각 개념적 설계와 논리적 설계의 일부 표현에 가깝지만, ㉢에 구현을 두어 물리적 설계를 누락했고 ㉣의 '백업 구성'도 구현 단계 전체를 대신할 수 없다. 올바르게는 ㉠ 개념적 설계, ㉡ 논리적 설계, ㉢ 물리적 설계, ㉣ 구현이다.

오답 포인트

개념적 설계와 논리적 설계의 선후관계를 바꾸거나, 물리적 설계보다 구현을 먼저 배치하는 것이 핵심 함정이다. 개념적 설계는 현실 세계의 요구를 DBMS와 무관하게 표현하고, 논리적 설계는 이를 선택한 데이터 모델에 맞게 변환한다는 차이를 구분해야 한다.

다시 나오면 이렇게 판단

데이터베이스 설계 절차는 현실의 요구를 파악한 뒤 추상적인 개념 구조를 만들고, 이를 데이터 모델의 논리 구조로 변환한 다음, 저장 방식인 물리 구조를 정하고 실제 DBMS에 구현하는 흐름으로 판단한다.

암기 포인트

요구 사항 분석 → 개념적 설계 → 논리적 설계 → 물리적 설계 → 구현의 순서로 기억한다.

NoteBase

4. Docker와 VM(Virtual Machine) 방식의 차이점으로 옳은 것만을 <보기>에서 모두 고르면?

< 보 기 >

ㄱ. Docker는 호스트 OS 위에서 컨테이너를 실행하고, VM은 하이퍼바이저 위에서 전체 OS를 실행한다.

ㄴ. Docker는 경량화된 가상화 방식으로 부팅 속도가 빠르며, VM은 별도의 OS가 필요해 더 무겁다.

ㄷ. VM은 애플리케이션 간 자원 격리가 가능하지만 Docker는 그렇지 않다.

ㄹ. Docker는 커널을 공유하므로 VM보다 자원 사용 효율이 높다.

ㅁ. VM은 호스트 OS와 커널을 공유하므로 컨테이너보다 보안성이 높다.

① ㄱ, ㄴ, ㄷ
② ㄱ, ㄴ, ㄹ
③ ㄱ, ㄴ, ㅁ
④ ㄱ, ㄹ, ㅁ
⑤ ㄴ, ㄷ, ㅁ

정답 2번

출제 포인트

이 문제는 Docker 컨테이너와 VM의 운영체제 구조, 자원 격리, 실행 부담의 차이를 묻는 문제이다.

핵심 개념

Docker 컨테이너: 호스트 OS의 커널을 공유하면서 애플리케이션과 그 실행 환경을 격리해 실행하는 방식이다.
별도의 게스트 OS를 부팅하지 않아 가볍고 시작이 빠르다.
VM: 하이퍼바이저가 가상 하드웨어를 제공하고 각 VM이 독립된 게스트 OS와 커널을 실행하는 방식이다.
컨테이너보다 자원 부담은 크지만 격리 경계가 더 강하다.
핵심 대비: Docker는 커널 공유, VM은 게스트 OS와 커널의 독립 실행이다. 따라서 ㄱ, ㄴ, ㄹ이 옳고 ㄷ, ㅁ은 틀리다.

NoteBase

선지별 해설

① X — ㄷ이 틀리고 ㄹ이 빠져 있다. Docker도 네임스페이스와 제어 그룹 등의 기능으로 프로세스와 자원을 격리할 수 있다. 올바른 조합은 ㄱ, ㄴ, ㄹ이다.
② O — ㄱ, ㄴ, ㄹ은 모두 옳다. Docker는 호스트 OS의 커널을 공유해 컨테이너를 실행하고, VM은 하이퍼바이저 위에서 별도의 게스트 OS를 실행한다. 이 구조 때문에 Docker는 일반적으로 부팅이 빠르고 자원 사용 효율이 높다.
③ X — ㅁ이 틀리고 ㄹ이 빠져 있다. 호스트 OS의 커널을 공유하는 쪽은 VM이 아니라 Docker 컨테이너이다. VM은 각 게스트 OS가 독립된 커널을 사용하므로 올바른 조합은 ㄱ, ㄴ, ㄹ이다.
④ X — ㅁ이 틀리고 ㄴ이 빠져 있다. VM이 호스트 OS의 커널을 공유한다는 표현은 잘못이며, 실제로는 Docker가 호스트 커널을 공유하고 VM은 별도의 게스트 OS와 커널을 실행한다. 올바른 조합은 ㄱ, ㄴ, ㄹ이다.
⑤ X — ㄷ과 ㅁ이 틀리고 ㄱ과 ㄹ이 빠져 있다. Docker도 자원 격리가 가능하며, 호스트 커널을 공유하는 쪽도 Docker이다. 올바른 조합은 ㄱ, ㄴ, ㄹ이다.

오답 포인트

ㄷ은 '격리 가능 여부'를 VM만의 특징처럼 표현한 것이 함정이다. Docker도 자원을 격리하지만 VM과 격리 방식과 강도가 다르다. ㅁ은 커널 공유의 주체를 뒤바꾼 표현이다. 커널 공유는 Docker의 특징이고, VM은 독립된 게스트 OS와 커널을 실행한다.

다시 나오면 이렇게 판단

먼저 커널 구조를 확인한다. 호스트 커널을 공유하면 컨테이너, 게스트 OS와 커널을 각각 실행하면 VM이다. 그다음 컨테이너는 가볍고 빠르며 자원 효율이 높고, VM은 자원 부담이 더 크지만 격리 경계가 더 강하다고 판단한다.

암기 포인트

Docker는 커널 공유로 가볍고 빠르며, VM은 게스트 OS까지 따로 실행해 무겁지만 격리가 강하다.

NoteBase

5. 릴레이션 R1을 릴레이션 R2와 R3으로 다음과 같이 분해하여 정규화를 성공적으로 수행하였다. 이 정규화 과정을 통해 R2와 R3이 속하게 되는 최고 차수의 정규형은?

R1:

학생번호	학과명	학과장
001	전자	홍길동
002	컴퓨터	이순신
003	건축	임꺽정
004	컴퓨터	이순신

⇩

R2:

학생번호	학과명
001	전자
002	컴퓨터
003	건축
004	컴퓨터

R3:

학과명	학과장
전자	홍길동
컴퓨터	이순신
건축	임꺽정

① 제1정규형
② 제2정규형
③ 제3정규형
④ BCNF
⑤ 제4정규형

정답 3번

출제 포인트

이 문제는 R1에서 학생번호 → 학과명 → 학과장으로 이어지는 이행적 함수 종속을 찾아 R2와 R3로 분해한 결과의 정규형을 판단하는 문제이다.

NoteBase

선지별 해설

① X
- 제1정규형은 각 속성값이 원자값이면 충족한다. R2와 R3는 제1정규형보다 높은 정규형도 충족하므로 최고 차수에 해당하지 않는다.

② X
- 제2정규형은 복합키의 일부에만 종속되는 부분 함수 종속을 제거한 단계이다. R2와 R3는 제2정규형보다 높은 정규형도 충족하므로 최고 차수에 해당하지 않는다.

③ 공식 채점상 O, 엄밀한 최고 차수 판정상 X
- R1의 학생번호 → 학과명과 학과명 → 학과장으로부터 학과장이 학생번호에 이행적으로 종속된다. 이를 R2와 R3로 분해하면 이행적 함수 종속이 제거되므로 제3정규형을 충족한다. 다만 제3정규형 충족은 분해 결과의 최고 정규형이 제3정규형이라는 뜻은 아니다.

④ X
- R2에서는 학생번호가 후보키이자 비자명 함수 종속 학생번호 → 학과명의 결정자이고, R3에서는 학과명이 후보키이자 비자명 함수 종속 학과명 → 학과장의 결정자이다. 따라서 두 릴레이션은 모든 결정자가 후보키인 BCNF를 충족한다. 그러나 두 릴레이션은 제4정규형도 충족하므로 BCNF가 최고 차수는 아니다.

⑤ 엄밀한 최고 차수 판정상 O
- R2와 R3는 각각 속성이 두 개인 이항 릴레이션이다. 이항 릴레이션에서는 다치 종속이 성립하더라도 모두 자명하므로 비자명 다치 종속이 존재하지 않는다. 따라서 두 릴레이션은 제4정규형을 충족하며, 제시된 선지 중 최고 차수는 제4정규형이다. 이는 공식 정답 ③과 충돌한다.

오답 포인트

이행적 함수 종속을 제거하는 분해 절차가 제3정규형을 목표로 했다는 사실과, 분해된 각 릴레이션이 실제로 충족하는 최고 정규형은 구분된다. R2와 R3의 결정자는 후보키이므로 BCNF를 충족하고, 두 릴레이션이 모두 이항 릴레이션이므로 제4정규형도 충족한다.

다시 나오면 이렇게 판단

정규화 절차의 명칭만으로 최고 정규형을 정하지 않고, 분해된 각 릴레이션의 함수 종속과 다치 종속을 차례로 점검한다. 모든 결정자가 후보키이면 BCNF이고, 모든 비자명 다치 종속의 결정자가 슈퍼키이면 제4정규형이다. 속성이 두 개인 이항 릴레이션에는 비자명 다치 종속이 없으므로 제4정규형을 충족한다.

암기 포인트

부분 종속 제거는 제2정규형, 이행 종속 제거는 제3정규형, 모든 결정자가 후보키이면 BCNF이다.

NoteBase

6. 접속하려는 웹 서버 URL로부터 해당 서버의 IP 주소를 반환해주는 것은?

① DNS
② ARP
③ RARP
④ DHCP
⑤ NAC

정답 1번

출제 포인트

웹 서버 URL에 포함된 호스트 이름을 해당 서버의 IP 주소로 변환하는 DNS의 역할을 묻는 문제이다.

핵심 개념

DNS는 사람이 읽기 쉬운 도메인 이름을 통신에 필요한 IP 주소로 변환하는 분산 이름 해석 시스템이다. URL 전체를 변환하는 것이 아니라 URL에서 호스트 이름 부분을 찾아 IP 주소로 해석한다.
DNS는 도메인 이름→IP 주소, ARP는 같은 네트워크의 IP 주소→MAC 주소, RARP는 MAC 주소→IP 주소, DHCP는 호스트에 IP 설정을 동적으로 할당, NAC는 정책에 따라 네트워크 접속을 통제한다.

선지별 해설

① O — DNS는 URL에 포함된 도메인 이름을 조회하여 해당 웹 서버의 IP 주소를 반환하므로 옳다.
② X — ARP가 수행한다는 표현이 틀리다. ARP는 같은 로컬 네트워크에서 알고 있는 IP 주소에 대응하는 MAC 주소를 알아내는 프로토콜이다. 도메인 이름을 IP 주소로 변환하는 것은 DNS이다.
③ X — RARP가 수행한다는 표현이 틀리다. RARP는 장치의 MAC 주소를 바탕으로 IP 주소를 알아내기 위한 프로토콜이다. 도메인 이름을 IP 주소로 변환하는 것은 DNS이다.
④ X — DHCP가 수행한다는 표현이 틀리다. DHCP는 호스트에 IP 주소, 서브넷 마스크, 기본 게이트웨이, DNS 서버 주소 등의 네트워크 설정을 동적으로 할당한다. 도메인 이름을 IP 주소로 변환하는 것은 DNS이다.
⑤ X — NAC가 수행한다는 표현이 틀리다. NAC는 사용자나 단말의 인증 상태와 보안 정책에 따라 네트워크 접속을 허용하거나 제한한다. 도메인 이름을 IP 주소로 변환하는 것은 DNS이다.

오답 포인트

ARP와 RARP에도 주소 변환 기능이 있어 DNS와 혼동하기 쉽다. 그러나 변환 대상이 다르다. 도메인 이름이 출발점이면 DNS이고, IP 주소와 MAC 주소 사이의 대응을 다루면 ARP 계열이다.

NoteBase

다시 나오면 이렇게 판단

문제에서 변환 전후의 주소 종류를 먼저 확인한다. 도메인 이름→IP 주소는 DNS, IP 주소→MAC 주소는 ARP, MAC 주소→IP 주소는 RARP로 판단한다. IP 설정의 자동 할당은 DHCP, 접속 허용·차단은 NAC로 구분한다.

암기 포인트

이름은 DNS, IP와 MAC의 연결은 ARP, IP 설정 배포는 DHCP, 접속 통제는 NAC이다.

NoteBase

7. 다음은 리눅스에서 ls -l set.exe 명령을 실행한 결과이다. set.exe 파일에 대한 설명으로 옳지 않은 것은?

```
-rwsr-xr-x 1 user1 user1 4096 11월 17 13:50 set.exe
```

① 파일 소유자의 로그인 ID와 파일 소유자의 그룹 이름은 user1이다.
② 하드 링크의 개수는 1이고 파일의 크기는 4096 바이트이다.
③ 파일 소유자를 제외한 다른 사용자는 파일 쓰기 권한이 없다.
④ 모든 사용자가 이 파일을 실행하는 동안 사용자 자신의 권한이 적용된다.
⑤ 파일 소유자는 chmod 4755 set.exe 명령으로 파일의 접근 권한을 설정할 수 있다.

정답 4번

출제 포인트

이 문제는 ls -l 출력에서 파일 형식·접근 권한, 하드 링크 수, 소유자·그룹, 파일 크기를 읽고 setuid가 실행 권한에 미치는 효과를 구분하는 문제이다.

핵심 개념

출력 -rwsr-xr-x 1 user1 user1 4096에서 첫 문자는 일반 파일(-), 소유자 권한은 읽기·쓰기·실행과 setuid(rws), 그룹과 기타 사용자 권한은 읽기·실행(r-x)이다. 이어지는 1은 하드 링크 수, 첫 번째 user1은 소유자, 두 번째 user1은 소유 그룹, 4096은 파일 크기(바이트)이다.
setuid는 실행 파일을 실행할 때 프로세스의 유효 사용자 ID를 실행한 사용자가 아니라 파일 소유자의 ID로 설정하는 특수 권한이다. chmod 4755에서 앞의 4는 setuid, 뒤의 755는 소유자 rwx·그룹 r-x·기타 사용자 r-x를 뜻한다.

선지별 해설

① O — 첫 번째 user1은 파일 소유자의 로그인 ID이고, 두 번째 user1은 파일 소유 그룹의 이름이다.
② O — 권한 정보 뒤의 1은 하드 링크 수이고, 4096은 파일 크기이므로 4096바이트이다.
③ O — 소유자 권한에는 w가 있지만 그룹과 기타 사용자 권한은 각각 r-x이므로 소유자를 제외한 사용자는 쓰기 권한이 없다.
④ X — 틀린 표현은 '사용자 자신의 권한이 적용된다'이다. setuid가 설정된 실행 파일을 실행하면 프로세스의 유효 사용자 ID가 파일 소유자인 user1로 설정되므로, 올바르게는 파일 소유자의 권한이 적용된다고 해야 한다.
⑤ O — 파일 소유자는 chmod로 접근 권한을 변경할 수 있으며, chmod 4755 set.exe는 setuid와 rwxr-xr-x를 설정하여 화면의 rwsr-xr-x와 같은 권한을 만든다.

NoteBase

오답 포인트

④는 모든 사용자에게 x가 있다는 사실과 실행 중 어떤 사용자의 권한이 적용되는지를 혼동하게 한다. x는 실행 가능 여부를 나타내지만, 소유자 실행 위치의 s는 setuid이므로 실행 시 유효 사용자 ID가 파일 소유자로 바뀐다.

다시 나오면 이렇게 판단

ls -l은 파일 형식, 소유자 권한, 그룹 권한, 기타 사용자 권한, 하드 링크 수, 소유자, 그룹, 크기 순으로 읽는다. 소유자 실행 자리의 s를 발견하면 setuid로 판단하고, 실행자의 실제 ID가 아니라 파일 소유자의 유효 ID로 동작한다는 점을 적용한다.

암기 포인트

s가 소유자 x 자리에 있으면 setuid이며, chmod의 첫 자리 4가 setuid를 뜻한다.

NoteBase

8. 하나의 주소에 하나의 원소가 저장되는 해시 테이블에서의 충돌 문제를 해결하기 위해 더블 해싱을 사용한다. 다음과 같이 해시 함수가 주어져 있고 현재 테이블에 5개의 원소만 저장되어 있는 경우, 원소 19가 저장될 주소는?

$h_i(x) = (h(x) + i \bullet f(x)) \bmod 17 \qquad i = 0, 1, 2, \ldots$

$h(x) = x \bmod 17$

$f(x) = 1 + (x \bmod 13)$

$h_0(17) = 0, h_0(36) = 2, h_0(20) = 3, h_0(5) = 5, h_0(26) = 9$

① 2
② 7
③ 9
④ 15
⑤ 16

정답 5번

출제 포인트

이 문제는 더블 해싱에서 초기 해시 주소가 충돌할 때 두 번째 해시 함수로 정한 간격만큼 이동하여 최초의 빈 주소를 찾는 문제이다.

조건 정리

주어진 값: 테이블 크기는 17이고, 원소 19에 대해 h(x)=x 17, f(x)=1+(x 13)을 사용한다.
구해야 할 값: h_i(19)=(h(19)+i· f(19)) 17로 조사했을 때 처음으로 비어 있는 주소이다.
주의할 조건: 현재 주소 2와 9는 각각 원소 36과 26이 저장되어 있어 사용할 수 없다.

사용할 공식

h_i(x)=(h(x)+i· f(x)) 17
이 공식을 쓰는 이유: 더블 해싱은 충돌이 발생할 때 $f(x)$를 조사 간격으로 사용하여 $i = 0, 1, 2, \ldots$ 순서로 다음 주소를 계산하기 때문이다.

NoteBase

계산 과정

1. 초기 주소는 h(19)=19 17=2이다. 주소 2에는 원소 36이 있으므로 충돌한다.
2. 조사 간격은 f(19)=1+(19 13)=1+6=7이다.
3. $i = 1$이면 h_1(19)=(2+1·7)17=9이다. 주소 9에는 원소 26이 있으므로 다시 충돌한다.
4. $i = 2$이면 h_2(19)=(2+2·7)17=16이다. 주소 16은 비어 있으므로 원소 19는 주소 16에 저장된다.

선지 판단

① X — 주소 2는 $h_0(19)$로 계산되는 초기 주소이지만 이미 원소 36이 저장되어 있다. 올바르게는 충돌 후 다음 조사 주소를 계속 계산해야 한다.
② X — 조사 간격 자체는 7이지만 주소 7을 조사하는 것은 아니다. 올바르게는 초기 주소 2에 간격 7을 더해 주소 9부터 조사한다.
③ X — 주소 9는 $h_1(19)$로 계산되는 두 번째 조사 주소이지만 이미 원소 26이 저장되어 있다. 올바르게는 한 번 더 간격 7을 더해 주소 16을 조사한다.
④ X — 주소 15는 조사 순서 2, 9, 16에 포함되지 않는다. 올바르게 계산한 최초의 빈 주소는 16이다.
⑤ O — $h_2(19) = 16$이고 주소 16이 비어 있으므로 원소 19가 저장될 주소이다.

오답 포인트

조사 간격 7을 저장 주소로 착각하거나, 계산된 주소 2 또는 9가 이미 점유되어 있다는 사실을 놓치기 쉽다. 더블 해싱에서는 계산값이 나올 때마다 그 주소의 점유 여부를 확인하고, 비어 있을 때까지 i를 1씩 증가시킨다.

다시 나오면 이렇게 판단

먼저 $h(x)$로 초기 주소를 구하고, 충돌하면 $f(x)$로 조사 간격을 구한다. 이후 (초기 주소+i· 조사 간격) 테이블 크기를 $i = 1, 2, \ldots$에 차례로 적용하여 최초의 빈 주소를 선택한다.

암기 포인트

더블 해싱은 '초기 주소 + 조사 횟수 × 두 번째 해시값'을 테이블 크기로 나눈 나머지를 순서대로 확인한다.

NoteBase

9. 머신러닝에서 과적합(overfitting)을 해결하는 방법으로 옳지 않은 것은?

① 정규화(regularization)
② 드롭아웃(dropout)
③ 교차 검증(cross validation)
④ 더 많은 데이터 수집
⑤ 더 많은 특성 추가

정답 5번

출제 포인트

과적합을 줄이는 방법과 오히려 모델 복잡도를 높여 과적합을 악화시킬 수 있는 방법을 구별하는 문제이다.

핵심 개념

과적합은 모델이 훈련 데이터의 잡음과 우연한 패턴까지 지나치게 학습하여 새로운 데이터에 대한 일반화 성능이 떨어지는 현상이다. 정규화와 드롭아웃은 모델의 복잡도나 특정 뉴런에 대한 의존을 낮추고, 교차 검증은 검증 성능을 기준으로 모델과 하이퍼파라미터를 선택하게 하며, 더 많은 데이터 수집은 훈련 표본의 다양성을 높여 과적합을 완화한다. 반면 검증 없이 특성을 더 많이 추가하면 모델 복잡도와 잡음 학습 가능성이 커질 수 있다.

선지별 해설

① O — 정규화는 큰 가중치에 페널티를 주어 모델이 지나치게 복잡해지는 것을 억제하므로 과적합 완화에 사용된다.
② O — 드롭아웃은 학습 중 일부 뉴런을 무작위로 비활성화하여 특정 뉴런들의 조합에 과도하게 의존하는 현상을 줄이므로 일반화에 도움이 된다.
③ O — 교차 검증은 여러 데이터 분할에서 검증 성능을 확인하여 과적합된 모델이나 하이퍼파라미터 선택을 피하는 데 사용된다.
④ O — 더 많은 데이터를 수집하면 학습 표본의 다양성이 커져 모델이 일부 사례의 잡음이나 우연한 패턴을 외우기 어려워지므로 과적합이 완화될 수 있다.
⑤ X — '더 많은 특성 추가'가 틀린 표현이다. 불필요하거나 잡음이 많은 특성을 추가하면 모델 복잡도가 커져 과적합이 심해질 수 있다. 올바르게는 유효한 특성을 선별하거나 차원을 축소하여 불필요한 특성을 줄이는 것이 과적합 완화 방법이다.

오답 포인트

데이터의 양을 늘리는 것과 특성의 수를 늘리는 것을 같은 의미로 보면 ④와 ⑤를 혼동하기 쉽다. 데이터 사례를 더 모으는 것은 표본의 다양성을 높이지만, 특성을 무조건 더 추가하는 것은 모델의 자유도와 잡음 학습 가능성을 높인다.

NoteBase

다시 나오면 이렇게 판단

모델 복잡도를 억제하거나 검증 절차를 강화하거나 학습 사례의 다양성을 늘리는 방법은 과적합 완화에 해당한다. 반대로 입력 특성이나 모델 자유도를 근거 없이 늘리는 방법은 과적합 위험을 키우는지 판단한다.

암기 포인트

데이터 수 증가는 일반화에 도움을 주지만, 특성 수의 무조건적 증가는 복잡도와 과적합 위험을 높인다.

10. 강화학습(reinforcement learning)의 특징으로 옳은 것은?

① 라벨이 있는 데이터로 학습을 진행한다.
② 보상(reward)을 기반으로 최적의 행동을 학습한다.
③ 데이터를 군집(cluster)으로 분류한다.
④ 정적 데이터 분석에 특화되어 있다.
⑤ 인간의 감정을 이해하는 데 최적화되었다.

정답 2번

출제 포인트

강화학습을 지도학습·비지도학습 등 다른 학습 방식과 구별하는 핵심 특징을 묻는 문제이다.

핵심 개념

강화학습은 에이전트가 환경과 상호작용하면서 행동의 결과로 받은 보상을 바탕으로 누적 보상을 최대화하는 행동 전략을 학습하는 방식이다.
지도학습은 정답 라벨이 있는 데이터로 입력과 정답의 관계를 학습하고, 비지도학습의 군집화는 라벨 없이 유사한 데이터를 집단으로 묶는다.

NoteBase

선지별 해설

① X
틀린 표현은 '라벨이 있는 데이터로 학습'이다. 이는 지도학습의 특징이며, 강화학습은 환경과 상호작용하여 얻는 보상을 학습 신호로 사용한다.

② O
강화학습은 행동에 따른 보상을 기반으로 장기적인 누적 보상을 크게 만드는 최적의 행동 또는 정책을 학습한다.

③ X
틀린 표현은 '데이터를 군집으로 분류'이다. 이는 비지도학습의 군집화에 해당하며, 강화학습은 보상을 기준으로 행동 전략을 학습한다.

④ X
틀린 표현은 '정적 데이터 분석에 특화'이다. 강화학습은 일반적으로 환경의 상태를 관찰하고 행동을 선택하며 보상을 받는 순차적 상호작용 문제에 활용된다.

⑤ X
틀린 표현은 '인간의 감정을 이해하는 데 최적화'이다. 감정 인식은 감성 컴퓨팅이나 자연어 처리·컴퓨터 비전 등의 응용 영역이며, 강화학습의 고유한 특징은 보상을 통해 행동 전략을 학습하는 것이다.

오답 포인트

①의 라벨, ③의 군집은 각각 지도학습과 비지도학습의 대표 키워드이다. 강화학습의 결정적 단서는 ②의 '보상'과 '최적의 행동'이다.

다시 나오면 이렇게 판단

선지에서 학습 신호를 확인한다. 정답 라벨이면 지도학습, 유사성에 따른 군집이면 비지도학습, 환경과의 상호작용에서 얻는 보상으로 행동을 개선하면 강화학습이다.

암기 포인트

강화학습은 '상태에서 행동하고, 보상을 받아, 더 나은 정책을 학습한다'로 기억한다.

NoteBase

11. 다음 SQL문의 ㉠~㉣에 들어갈 키워드로 옳은 것은?

```
SELECT 학과, COUNT(*) AS 인원수
  ㉠   학생
  ㉡   성적 >= 70
  ㉢   학과
  ㉣   COUNT(*) >= 10
ORDER BY 인원수 DESC
```

	㉠	㉡	㉢	㉣
①	FROM	WHERE	ORDER BY	GROUP BY
②	FROM	GROUP BY	HAVING	WHERE
③	GROUP BY	FROM	WHERE	HAVING
④	FROM	WHERE	GROUP BY	HAVING
⑤	FROM	WHERE	HAVING	GROUP BY

정답 4번

출제 포인트

이 문제는 집계 함수와 그룹 조건이 포함된 SELECT문의 절 순서를 묻는 문제이다.

올바른 순서

1. FROM 학생 — 조회할 테이블을 지정한다.
2. WHERE 성적 >= 70 — 그룹화하기 전에 개별 행을 선별한다.
3. GROUP BY 학과 — 남은 행을 학과별로 묶는다.
4. HAVING COUNT(*) >= 10 — 그룹별 집계 결과가 10명 이상인 학과만 선별한다.
5. ORDER BY 인원수 DESC — 최종 결과를 인원수의 내림차순으로 정렬한다.

왜 이 순서인가: WHERE는 그룹화 전의 개별 행을 조건으로 걸고, HAVING은 GROUP BY로 만들어진 그룹의 집계 결과를 조건으로 걸기 때문이다.

NoteBase

선지별 해설

① X
ㄷ의 ORDER BY와 ㄹ의 GROUP BY가 틀렸다. 올바르게는 ㄷ에 GROUP BY, ㄹ에 HAVING이 들어가야 한다. ORDER BY는 이미 마지막에 제시되어 있다.

② X
ㄴ의 GROUP BY, ㄷ의 HAVING, ㄹ의 WHERE가 틀렸다. 올바르게는 FROM 다음에 WHERE로 행을 선별하고, GROUP BY로 묶은 뒤 HAVING으로 그룹 조건을 적용한다.

③ X
ㄱ의 GROUP BY, ㄴ의 FROM, ㄷ의 WHERE가 틀렸다. 올바르게는 ㄱ FROM, ㄴ WHERE, ㄷ GROUP BY, ㄹ HAVING의 순서이다.

④ O
FROM 학생 → WHERE 성적 >= 70 → GROUP BY 학과 → HAVING COUNT(*) >= 10 → ORDER BY 인원수 DESC의 올바른 절 순서를 완성한다.

⑤ X
ㄷ의 HAVING과 ㄹ의 GROUP BY가 서로 바뀌었다. 올바르게는 먼저 GROUP BY 학과로 그룹을 만든 뒤 HAVING COUNT(*) >= 10으로 집계 조건을 적용한다.

오답 포인트

WHERE와 HAVING은 모두 조건을 나타내지만 적용 대상이 다르다. 성적 >= 70은 개별 학생 행의 조건이므로 WHERE에 쓰고, COUNT(*) >= 10은 학과별 그룹의 집계 조건이므로 GROUP BY 뒤의 HAVING에 써야 한다.

다시 나오면 이렇게 판단

SELECT문의 기본 절 순서를 SELECT → FROM → WHERE → GROUP BY → HAVING → ORDER BY로 놓고, 개별 행 조건은 WHERE, 그룹·집계 조건은 HAVING에 배치한다.

암기 포인트

프롬-웨어-그룹-해빙-오더: 행을 고른 뒤 묶고, 묶은 결과를 거른 뒤 정렬한다.

NoteBase

12. 소프트웨어 개발에 사용되는 Git의 주요 기능으로 옳은 것은?

① 데이터를 암호화한다.
② 데이터베이스를 관리한다.
③ 소프트웨어를 평가한다.
④ 코드를 컴파일한다.
⑤ 코드 변경 사항을 추적한다.

정답 5번

출제 포인트

Git의 핵심 기능이 소스 코드의 변경 이력을 기록하고 추적하는 버전 관리임을 묻는 문제이다.

핵심 개념

Git은 분산 버전 관리 시스템이다. 파일의 변경 내용을 커밋 단위로 기록하고, 과거 버전 확인·복구, 브랜치를 이용한 병렬 작업, 변경 내용 병합과 협업을 지원한다.
Git은 코드 자체를 암호화하거나 컴파일하고 소프트웨어의 품질을 평가하는 도구가 아니며, 일반적인 데이터베이스 관리 시스템도 아니다.

선지별 해설

① X — '데이터를 암호화한다'는 설명이 틀리다. 올바르게는 암호화 도구가 데이터를 암호화하며, Git은 파일의 버전과 변경 이력을 관리한다.
② X — '데이터베이스를 관리한다'는 설명이 틀리다. 올바르게는 데이터베이스 관리 시스템이 데이터베이스를 관리하며, Git은 주로 소스 코드 등 파일의 버전을 관리한다.
③ X — '소프트웨어를 평가한다'는 설명이 틀리다. 올바르게는 테스트·품질 관리 도구가 소프트웨어의 동작이나 품질을 평가하며, Git은 개발 산출물의 변경 이력을 관리한다.
④ X — '코드를 컴파일한다'는 설명이 틀리다. 올바르게는 컴파일러가 소스 코드를 목적 코드나 실행 가능한 형태로 변환하며, Git은 코드 변경 사항을 기록하고 추적한다.
⑤ O — Git은 각 변경 사항을 커밋으로 기록하여 누가, 언제, 무엇을 변경했는지 추적하고 이전 상태도 확인할 수 있다.

오답 포인트

①~④는 모두 소프트웨어 개발 과정에서 접할 수 있는 기능이지만 Git의 기능은 아니다. 특히 Git이 개발 도구라는 이유만으로 컴파일 기능까지 수행한다고 혼동하기 쉽다.

NoteBase

다시 나오면 이렇게 판단

도구의 대표 동사를 기준으로 구분한다. Git은 기록·추적·분기·병합, 암호화 도구는 암호화, 데이터베이스 관리 시스템은 데이터 저장·조회·관리, 테스트 도구는 검증, 컴파일러는 번역을 담당한다.

암기 포인트

Git = 코드 변경 이력의 기록과 추적을 위한 분산 버전 관리 시스템이다.

NoteBase

13. 자료구조 큐(queue)의 적용 대상으로 옳은 것은?

① 실행 취소/재실행(undo/redo) 기능
② 운영체제의 작업 스케줄링
③ 깊이 우선 탐색(Depth First Search)
④ 괄호의 짝 검사
⑤ 재귀(recursion) 구현

정답 2번

출제 포인트

큐와 스택의 처리 순서 및 대표적인 적용 대상을 구별하는 문제이다.

핵심 개념

큐(queue): 먼저 들어온 자료를 먼저 꺼내는 선입선출(FIFO) 구조이다. 작업 스케줄링, 대기열 처리, 너비 우선 탐색 등에 사용된다.
스택(stack): 나중에 들어온 자료를 먼저 꺼내는 후입선출(LIFO) 구조이다. 실행 취소·재실행, 깊이 우선 탐색, 괄호 짝 검사, 재귀 호출 처리 등에 사용된다.

NoteBase

선지별 해설

① X
실행 취소/재실행은 가장 최근의 작업부터 되돌리거나 복원하므로 후입선출 방식의 스택을 사용한다. 틀린 표현은 '큐의 적용 대상'이며, 올바르게는 '스택의 적용 대상'이다.

② O
운영체제는 도착하거나 준비된 작업을 대기열에 넣고 순서에 따라 처리할 수 있으므로 작업 스케줄링은 큐의 대표적인 적용 대상이다.

③ X
깊이 우선 탐색은 현재 경로에서 가장 최근에 발견한 정점을 먼저 탐색하므로 스택 또는 재귀 호출을 사용한다. 틀린 표현은 '큐의 적용 대상'이며, 올바르게는 '스택의 적용 대상'이다. 큐는 너비 우선 탐색에 사용된다.

④ X
괄호의 짝 검사는 여는 괄호를 쌓아 두었다가 가장 최근의 여는 괄호부터 닫는 괄호와 대응시키므로 스택을 사용한다. 틀린 표현은 '큐의 적용 대상'이며, 올바르게는 '스택의 적용 대상'이다.

⑤ X
재귀 호출은 호출된 함수의 복귀 주소와 지역 상태를 호출 순서대로 쌓고 가장 최근 호출부터 반환하므로 호출 스택으로 구현된다. 틀린 표현은 '큐의 적용 대상'이며, 올바르게는 '스택의 적용 대상'이다.

오답 포인트

②를 제외한 선지는 모두 '최근에 들어온 항목을 먼저 처리한다'는 공통점이 있는 스택의 사례이다. 특히 깊이 우선 탐색과 재귀를 큐의 사례로 혼동하기 쉽지만, 둘 다 가장 최근의 탐색 경로나 호출을 먼저 되돌아간다.

다시 나오면 이렇게 판단

처리 대상을 줄 세워 먼저 들어온 것부터 처리하면 큐이고, 가장 최근에 저장한 상태나 작업부터 꺼내면 스택이다. 탐색에서는 너비 우선 탐색은 큐, 깊이 우선 탐색은 스택으로 구별한다.

암기 포인트

큐는 FIFO·대기열·작업 스케줄링, 스택은 LIFO·실행 취소·DFS·괄호·재귀이다.

NoteBase

14. 다음은 NumPy 라이브러리를 사용할 때 가장 먼저 작성하는 코드이다. NumPy 배열을 생성하는 코드로 옳은 것만을 <보기>에서 모두 고르면?

```
import numpy as np
```

< 보 기 >

ㄱ. a = np.array[1, 2, 3]
ㄴ. b = np.array((1, 2, 3))
ㄷ. c = np.arrange(0, 5)
ㄹ. d = np.linspace(0, 1, num=5)
ㅁ. e = np.zeros([2, 3])

① ㄴ, ㅁ
② ㄱ, ㄷ, ㄹ
③ ㄴ, ㄷ, ㅁ
④ ㄴ, ㄹ, ㅁ
⑤ ㄱ, ㄷ, ㄹ, ㅁ

정답 1번

출제 포인트

NumPy 함수의 정확한 이름과 호출 문법을 구별하여 실제로 ndarray를 생성하는 코드를 찾는 문제이다. 다만 제시된 정답키 1번은 실제 NumPy 동작과 충돌하며, 코드 기준으로는 ㄴ, ㄹ, ㅁ을 묶은 4번이 옳다.

핵심 개념

np.array(객체)는 리스트나 튜플 같은 객체를 ndarray로 변환한다. np.arange(시작, 끝)는 일정한 간격의 값을, np.linspace(시작, 끝, num=개수)는 지정한 구간에서 같은 간격으로 나눈 지정 개수의 값을 ndarray로 만든다. np.zeros(형상)는 주어진 형상의 0 배열을 만든다. 함수 호출에는 소괄호를 사용하며, 함수 이름의 철자도 정확해야 한다.

NoteBase

선지별 해설

① X
- ㄴ의 np.array((1, 2, 3))과 ㅁ의 np.zeros([2, 3])은 모두 ndarray를 생성하지만, ㄹ의 np.linspace(0, 1, num=5)도 ndarray를 생성하므로 ㄴ, ㅁ만 고른 부분이 틀렸다. 올바르게는 ㄴ, ㄹ, ㅁ이다.

② X
- ㄱ은 함수 호출에 대괄호를 사용했고, ㄷ은 존재하지 않는 np.arrange를 사용했으므로 배열을 생성하지 못한다. 올바르게는 ㄱ을 np.array([1, 2, 3])로, ㄷ을 np.arange(0, 5)로 고쳐야 하며, 옳은 항목의 조합은 ㄴ, ㄹ, ㅁ이다.

③ X
- ㄷ의 np.arrange는 함수 이름의 철자가 틀렸고, 정상적으로 배열을 만드는 ㄹ이 빠졌다. 올바르게는 np.arange(0, 5)라고 쓰며, 제시된 코드 중 옳은 항목은 ㄴ, ㄹ, ㅁ이다.

④ O
- ㄴ은 튜플을 1차원 ndarray로 변환하고, ㄹ은 0부터 1까지 같은 간격의 값 5개를 담은 ndarray를 만들며, ㅁ은 2행 3열의 0 배열을 만든다. 따라서 실제 코드 기준의 올바른 조합이다.

⑤ X
- ㄱ과 ㄷ이 잘못된 코드인데 포함되어 있다. 올바르게는 ㄱ의 대괄호를 소괄호로 바꾸고 ㄷ의 arrange를 arange로 고쳐야 하며, 수정 전 제시된 코드에서 옳은 것은 ㄴ, ㄹ, ㅁ이다.

오답 포인트

ㄱ의 리스트 표기 [1, 2, 3] 자체가 아니라 np.array 뒤에 소괄호 없이 대괄호를 붙인 점이 오류이다. 또한 arange와 arrange의 철자 차이를 놓치기 쉽다. 가장 큰 함정은 ㄹ의 linspace도 결과로 ndarray를 생성한다는 점이며, 이 때문에 정답키 1번은 코드의 실제 동작과 맞지 않는다.

다시 나오면 이렇게 판단

각 표현에서 함수 이름, 호출 소괄호, 인수 형식을 차례로 확인한 뒤 반환값이 ndarray인지 판단한다. array, arange, linspace, zeros는 모두 배열 생성에 사용할 수 있다.

암기 포인트

array는 소괄호로 호출하고, arange에는 r이 하나이며, linspace와 zeros도 ndarray를 만든다.

NoteBase

15. 전송 제어 프로토콜, 라우팅 테이블, 네트워크 인터페이스 등의 정보를 나타내주는 네트워크 명령어는?

① ping
② tracert
③ netstat
④ nslookup
⑤ ifconfig

정답 3번

출제 포인트

이 문제는 네트워크 상태 정보를 조회하는 명령어인 netstat과 연결 확인·경로 추적·DNS 조회·인터페이스 설정 명령어를 구별하는 문제이다.

핵심 개념

netstat: 현재 네트워크 연결 상태와 프로토콜별 통계, 라우팅 테이블, 네트워크 인터페이스 통계 등을 확인하는 명령어이다.
ping은 상대 호스트와의 통신 가능 여부를 확인하고, tracert는 목적지까지의 경로를 추적하며, nslookup은 DNS 정보를 조회하고, ifconfig는 네트워크 인터페이스의 설정과 상태를 확인·변경한다.

선지별 해설

① X — ping은 ICMP 메시지를 이용해 상대 호스트의 도달 가능 여부와 응답 시간을 확인한다. 틀린 표현은 '전송 제어 프로토콜, 라우팅 테이블, 네트워크 인터페이스 등의 정보를 나타내는 명령어'이며, 올바르게는 '호스트의 도달 가능 여부와 응답 시간을 확인하는 명령어'이다.
② X — tracert는 목적지까지 패킷이 거치는 라우터의 홉과 경로를 추적한다. 틀린 표현은 '여러 네트워크 상태 정보를 종합적으로 나타내는 명령어'이며, 올바르게는 '목적지까지의 네트워크 경로를 추적하는 명령어'이다.
③ O — netstat은 활성 TCP 연결, 프로토콜별 통계, 라우팅 테이블, 네트워크 인터페이스 통계 등 네트워크 상태 정보를 표시하므로 문제의 설명에 해당한다.
④ X — nslookup은 도메인 이름과 IP 주소의 대응 관계 등 DNS 정보를 조회한다. 틀린 표현은 '라우팅 테이블과 네트워크 인터페이스 정보를 나타내는 명령어'이며, 올바르게는 'DNS 서버에 질의하여 도메인 이름이나 IP 주소 정보를 조회하는 명령어'이다.
⑤ X — ifconfig는 주로 Unix 계열 운영체제에서 네트워크 인터페이스의 주소와 상태를 확인하거나 설정한다. 틀린 표현은 'TCP 연결과 라우팅 테이블까지 종합적으로 나타내는 명령어'이며, 올바르게는 '네트워크 인터페이스의 설정과 상태를 확인·변경하는 명령어'이다.

NoteBase

오답 포인트

⑤ ifconfig도 네트워크 인터페이스 정보를 보여 주므로 헷갈릴 수 있다. 그러나 문제는 인터페이스 정보만이 아니라 TCP 연결·프로토콜 통계와 라우팅 테이블까지 함께 제시하므로 netstat을 묻는다.

다시 나오면 이렇게 판단

명령어가 확인하는 대상을 기준으로 구별한다. 연결·프로토콜 통계·라우팅 테이블은 netstat, 도달 여부는 ping, 경로는 tracert, DNS는 nslookup, 인터페이스 설정은 ifconfig이다.

암기 포인트

netstat은 네트워크 상태, ping은 도달 확인, tracert는 경로 추적, nslookup은 DNS 조회, ifconfig는 인터페이스 설정이다.

NoteBase

16. 데브옵스(DevOps)의 주요 목적으로 옳은 것은?

① 개발팀과 운영팀을 분리한다.
② 소프트웨어 개발과 배포를 자동화한다.
③ 코드 품질을 저하시킨다.
④ 수동 테스트를 늘린다.
⑤ 개발, 테스트, 운영 환경을 격리한다.

정답 2번

출제 포인트

데브옵스의 핵심 목적인 개발·운영 간 협업과 소프트웨어 전달 과정의 자동화를 이해하는 문제이다.

핵심 개념

데브옵스(DevOps)는 개발(Development)과 운영(Operations)이 협업하여 소프트웨어의 개발, 테스트, 배포, 운영 과정을 지속적으로 개선하는 문화이자 실천 방식이다. 핵심 키워드는 협업, 자동화, 지속적 통합·배포, 빠르고 안정적인 전달이다.

선지별 해설

① X — 틀린 표현은 '개발팀과 운영팀을 분리한다'이다. 데브옵스는 두 팀 사이의 장벽을 낮추고 협업과 공동 책임을 강화한다.
② O — 개발, 테스트, 빌드, 배포 등의 반복 과정을 자동화하여 소프트웨어를 빠르고 안정적으로 제공하는 것이 데브옵스의 주요 목적이다.
③ X — 틀린 표현은 '코드 품질을 저하시킨다'이다. 자동화된 테스트와 지속적 피드백을 활용하여 결함을 조기에 발견하고 코드와 서비스의 품질을 높인다.
④ X — 틀린 표현은 '수동 테스트를 늘린다'이다. 반복 가능한 테스트를 자동화하여 수동 작업을 줄이고 검증의 속도와 일관성을 높인다.
⑤ X — 틀린 표현은 '개발, 테스트, 운영 환경을 격리한다'이다. 필요한 환경 구분은 유지할 수 있지만, 데브옵스의 목적은 팀과 전달 절차를 단절하는 것이 아니라 환경의 일관성을 높이고 개발부터 운영까지의 흐름을 연계하는 것이다.

오답 포인트

①과 ⑤의 '분리·격리'는 안정성을 위한 조치처럼 보일 수 있지만, 데브옵스의 핵심은 조직과 절차의 단절을 줄이는 데 있다. ④의 수동 테스트도 자동화를 중시하는 데브옵스의 방향과 반대이다.

NoteBase

다시 나오면 이렇게 판단

선지에 협업, 자동화, 지속적 통합·배포, 빠른 피드백, 안정적 전달이 제시되면 데브옵스의 목적에 부합하는지 판단한다. 반대로 팀의 분리, 반복 작업의 수동화, 품질 저하, 전달 과정의 단절은 데브옵스의 방향과 어긋난다.

암기 포인트

DevOps는 개발과 운영의 협업, 과정의 자동화, 빠르고 안정적인 배포이다.

NoteBase

17. 다중 스레드(thread) 기반의 프로그래밍 환경에서 바이너리 세마포어(binary semaphore)의 용법으로 옳지 않은 것은?

① 여러 스레드 간에 공유되는 자원 하나를 한 번에 하나의 스레드만 접근하도록 한다.

② 여러 스레드 간에 공유되는 동일한 자원이 2개가 있을 때 동시에 접근하는 스레드의 개수를 2개로 제한한다.

③ 데이터를 수신하는 스레드에 데이터가 도착하면 데이터를 처리하는 스레드가 수신된 데이터의 처리를 시작한다.

④ 특정 이벤트의 완료 시 다른 스레드에게 알려준다.

⑤ 상호 배제(mutual exclusion)를 구현한다.

정답 2번

출제 포인트

이 문제는 값이 0과 1만 가능한 바이너리 세마포어의 용도와, 여러 개의 동일 자원을 관리하는 계수 세마포어의 용도를 구별하는 문제이다.

핵심 개념

바이너리 세마포어: 상태값이 0 또는 1인 동기화 도구로, 한 번에 하나만 통과시키는 상호 배제나 스레드 간 사건 통지에 사용할 수 있다.
계수 세마포어: 0 이상의 카운트로 사용 가능한 자원 수를 나타내며, 동일한 자원이 여러 개일 때 동시에 접근할 수 있는 스레드 수를 그 개수만큼 허용한다.
핵심 차이는 허용 수가 1이면 바이너리 세마포어, 2 이상이면 계수 세마포어가 필요하다는 점이다.

선지별 해설

① O — 바이너리 세마포어를 잠금처럼 사용하면 공유 자원에 진입하기 전에 획득하고 사용 후 해제하므로 한 번에 하나의 스레드만 접근하게 할 수 있다.
② X — 틀린 표현은 '동시에 접근하는 스레드의 개수를 2개로 제한한다'이다. 바이너리 세마포어는 0과 1만 표현하므로 동시 접근을 2개까지 허용할 수 없다. 올바르게는 초기값이 2인 계수 세마포어를 사용하여 동일한 자원 2개에 최대 2개의 스레드가 동시에 접근하게 한다.
③ O — 수신 스레드가 데이터 도착 시 세마포어를 해제하여 신호를 보내고 처리 스레드가 이를 획득해 실행하도록 하면, 생산자와 소비자 사이의 사건 통지를 구현할 수 있다.
④ O — 이벤트를 기다리는 스레드를 대기시키고 이벤트 완료 시 다른 스레드가 세마포어를 해제하면 완료 사실을 알리는 동기화 신호로 사용할 수 있다.
⑤ O — 바이너리 세마포어의 값이 1일 때 한 스레드만 획득하도록 하면 임계구역에 대한 동시 진입을 막아 상호 배제를 구현할 수 있다.

NoteBase

오답 포인트

②의 '자원이 2개'와 '스레드를 2개로 제한'이라는 표현이 함정이다. 세마포어라는 이름만 보고 모두 가능하다고 판단하면 안 된다. 바이너리 세마포어는 허용 수가 최대 1이므로, 허용 수 2에는 계수 세마포어가 필요하다.

다시 나오면 이렇게 판단

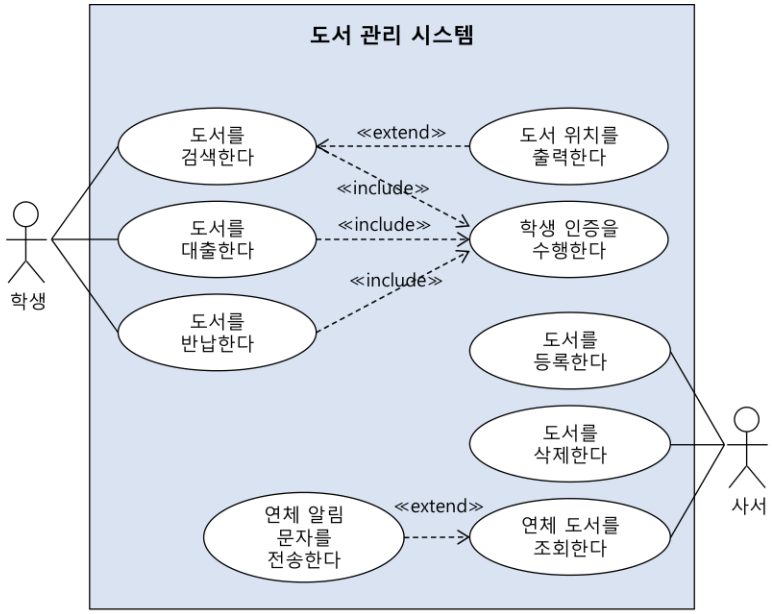
선지에 제시된 동시 허용 개수를 먼저 확인한다. 한 번에 하나만 허용하거나 사건 발생 여부만 나타내면 바이너리 세마포어를 적용할 수 있고, 둘 이상의 동일 자원을 세어 관리하면 계수 세마포어를 적용한다.

암기 포인트

바이너리는 0·1로 잠금과 신호, 여러 자원은 개수를 세는 계수 세마포어이다.

NoteBase

18. 다음은 '도서 관리 시스템'의 유스케이스 다이어그램이다. 다이어그램의 구조와 기능 간 관계에 대한 설명으로 옳지 않은 것은?



① 학생이 도서를 대출하거나 반납하기 위해서는, 본인임을 인증하는 절차가 선행되어야 한다.

② '도서 위치를 출력한다' 기능은, 사용자가 책을 검색하는 과정에서 상황에 따라 선택적으로 수행된다.

③ 사서는 도서를 등록하거나 삭제할 수 있으며, 이들 기능은 서로 독립적으로 수행된다.

④ 연체 도서를 조회한 이후, 사서는 연체 알림 문자를 발송할 수 있다.

⑤ 학생이 도서를 반납하는 기능은, 도서 검색 기능에 포함되어 있는 구조로 표현되어 있다.

정답 5번

출제 포인트

이 문제는 유스케이스 다이어그램에서 액터의 연관 관계와 `«include»`, `«extend»`의 화살표 방향을 읽는 문제이다. `«include»`는 기본 유스케이스가 반드시 사용하는 공통 기능을, `«extend»`는 일정 조건에서 선택적으로 추가되는 기능을 나타낸다.

NoteBase

선지별 해설

① O — '도서를 대출한다'와 '도서를 반납한다'에서 '학생 인증을 수행한다'로 `«include»` 관계가 연결되어 있다. 따라서 대출이나 반납에는 학생 인증이 필수적으로 포함된다. 다만 유스케이스 다이어그램은 엄밀한 시간 순서를 표현하는 도식은 아니므로, 여기서 '선행'은 인증이 필수 절차라는 의미로 이해한다.

② O — '도서 위치를 출력한다'에서 '도서를 검색한다' 쪽으로 `«extend»` 화살표가 향한다. 따라서 도서 위치 출력은 도서 검색 중 조건에 따라 선택적으로 추가되는 기능이다.

③ O — 사서는 '도서를 등록한다'와 '도서를 삭제한다'에 각각 연관되어 있고, 두 유스케이스 사이에는 `«include»`나 `«extend»` 관계가 없다. 따라서 도식상 두 기능은 서로 독립적으로 수행된다.

④ O — '연체 알림 문자를 전송한다'에서 '연체 도서를 조회한다' 쪽으로 `«extend»` 화살표가 향한다. 따라서 연체 도서 조회를 기본으로 하여 조건이 충족되면 연체 알림 문자 전송 기능이 추가될 수 있다.

⑤ X — '도서를 반납한다'와 '도서를 검색한다' 사이에는 `«include»` 관계가 없다. 틀린 표현은 '도서 검색 기능에 포함되어 있는 구조'이며, 올바르게는 '도서 반납 기능은 학생 인증 기능을 포함하고, 도서 검색 기능과는 별개의 유스케이스로 표현되어 있다'이다.

오답 포인트

⑤는 학생 액터가 도서 검색과 도서 반납에 모두 연결되어 있다는 사실을 기능 간 포함 관계로 오해하게 만든다. 같은 액터와 연관되어 있다는 것만으로 두 유스케이스 사이에 `«include»` 관계가 생기지는 않는다.

다시 나오면 이렇게 판단

먼저 액터와 유스케이스의 실선은 단순한 참여 관계로 읽는다. 그다음 점선 화살표의 표기를 확인하여 `«include»`이면 화살표가 가리키는 기능이 필수로 포함되고, `«extend»`이면 화살표가 가리키는 기본 기능에 조건부 기능이 추가된다고 판단한다.

암기 포인트

`include`는 필수 공통 기능, `extend`는 조건부 추가 기능이며, 둘 다 화살표가 가리키는 쪽을 기준으로 관계를 읽는다.

NoteBase

19. 클라우드 컴퓨팅(cloud computing)의 핵심 특징으로 옳지 않은 것은?

① 자원 풀링을 통한 효율적인 자원 관리
② 필요에 따른 주문형 서비스 기능
③ 물리적 서버를 사용자가 직접 유지 관리
④ 컴퓨팅 자원의 확장성과 탄력성
⑤ 광범위한 네트워크를 통한 접근

정답 3번

출제 포인트

클라우드 컴퓨팅의 핵심 특징과 전통적인 자체 서버 운영 방식을 구별하는 문제이다.

핵심 개념

클라우드 컴퓨팅은 인터넷 등 네트워크를 통해 공유된 컴퓨팅 자원을 필요할 때 제공받아 사용하는 방식이다. 자원 풀링, 주문형 서비스, 신속한 확장·축소, 광범위한 네트워크 접근이 대표적인 특징이다. 물리적 서버의 설치·유지 관리는 일반적으로 클라우드 제공자가 담당하고, 사용자는 제공된 가상화 자원과 서비스를 이용한다.

선지별 해설

① O — 여러 사용자가 공유하는 컴퓨팅 자원을 하나의 풀로 구성하고 수요에 따라 배분하므로 효율적인 자원 관리가 가능하다.
② O — 사용자는 필요할 때 서버·저장 공간 등의 자원을 요청하여 사용할 수 있으므로 주문형 서비스 기능에 해당한다.
③ X — 틀린 표현은 '물리적 서버를 사용자가 직접 유지 관리'이다. 클라우드에서는 물리적 기반 시설을 일반적으로 서비스 제공자가 유지·관리하고, 사용자는 제공받은 컴퓨팅 자원과 서비스를 이용한다.
④ O — 수요 변화에 따라 컴퓨팅 자원을 신속하게 늘리거나 줄일 수 있으므로 확장성과 탄력성이 핵심 특징이다.
⑤ O — 인터넷 등 광범위한 네트워크를 통해 다양한 단말에서 서비스에 접근할 수 있으므로 클라우드 컴퓨팅의 특징에 해당한다.

오답 포인트

③의 '사용자가 직접 유지 관리'는 자체 구축형 서버 운영의 특징이다. 클라우드 서비스도 사용자가 운영체제나 애플리케이션 일부를 관리할 수는 있지만, 물리적 서버 자체의 유지 관리를 사용자가 직접 담당한다는 설명은 클라우드의 핵심 특징이 아니다.

NoteBase

다시 나오면 이렇게 판단

선지에 자원 공유, 필요 시 즉시 사용, 신속한 확장·축소, 네트워크 접근이 나오면 클라우드의 특징으로 판단한다. 물리 장비를 사용자가 직접 설치·유지한다는 표현은 자체 서버 운영 방식으로 구별한다.

암기 포인트

클라우드는 '공유하고, 필요할 때 쓰고, 빠르게 늘리고 줄이며, 네트워크로 접근한다'로 정리한다. 물리 서버 관리는 제공자 측의 역할이다.

NoteBase

20. 사용자 A와 TCP 연결을 유지하고 있는 사용자 B가 A로부터 다음 <정보>의 헤더필드 값과 500 바이트의 데이터가 포함된 세그먼트를 정상적으로 수신하였다. 이에 대한 응답으로 B가 A에게 보내는 세그먼트 헤더의 확인응답 번호로 옳은 것은?

< 정 보 >

순서(sequence) 번호: 2000

확인응답(acknowledgement) 번호: 190

① 500
② 690
③ 2001
④ 2500
⑤ 2501

정답 4번

출제 포인트

이 문제는 TCP의 확인응답 번호가 수신자가 다음에 받기를 기대하는 바이트의 순서 번호라는 원리를 이용해 응답 세그먼트의 확인응답 번호를 구하는 문제이다.

조건 정리

주어진 값: A가 보낸 세그먼트의 순서 번호는 2000이고 데이터 길이는 500바이트이다.
구해야 할 값: B가 A에게 보내는 응답 세그먼트의 확인응답 번호이다.
주의할 조건: 수신한 세그먼트에 들어 있던 확인응답 번호 190은 A가 B에게 기대하는 다음 바이트 번호이므로, B가 A의 데이터를 확인응답할 때 계산에 사용하지 않는다.

사용할 공식

TCP의 누적 확인응답 번호는 다음과 같이 계산한다.
응답 ACK 번호=수신 세그먼트의 순서 번호+수신 데이터 길이
이 공식을 쓰는 이유는 ACK 번호가 마지막으로 받은 바이트 번호가 아니라 다음에 받기를 기대하는 바이트 번호를 나타내기 때문이다.

NoteBase

계산 과정

1. 첫 데이터 바이트의 순서 번호는 2000이다.
2. 500바이트를 정상 수신했으므로 받은 바이트의 순서 번호 범위는 2000부터 2499까지이다.
3. 다음에 기대하는 바이트의 순서 번호는 2000 + 500 = 2500이다. 따라서 B가 보내는 확인응답 번호는 2500이다.

선지 판단

① X — 500은 데이터의 길이일 뿐 다음에 기대하는 바이트의 순서 번호가 아니다. 올바른 값은 순서 번호 2000에 데이터 길이 500을 더한 2500이다.
② X — 690은 수신 세그먼트의 확인응답 번호 190에 데이터 길이 500을 잘못 더한 값이다. 올바르게는 수신 세그먼트의 순서 번호 2000에 500을 더해 2500으로 계산한다.
③ X — 2001은 데이터가 1바이트뿐이라고 잘못 본 값이다. 500바이트 전체를 받았으므로 올바른 확인응답 번호는 2500이다.
④ O — 2000 + 500 = 2500이며, 이는 B가 A로부터 다음에 받기를 기대하는 바이트의 순서 번호이다.
⑤ X — 2501은 500바이트의 마지막 번호 2499 뒤에 1을 더한 뒤 다시 1을 더한 값이다. 올바르게는 2000부터 2499까지를 수신했으므로 다음 번호는 2500이다.

오답 포인트

핵심 함정은 ACK 번호를 마지막으로 받은 바이트 번호로 보거나, 반대 방향 통신을 확인하는 기존 ACK 번호 190을 계산에 섞는 것이다. 이 문항에서는 A의 데이터가 2000부터 시작하므로 B의 응답 ACK는 190이 아니라 2000과 데이터 길이 500으로 계산한다.

다시 나오면 이렇게 판단

TCP 데이터 세그먼트를 정상 수신한 뒤의 ACK를 묻는 문제는 송신 세그먼트의 순서 번호와 데이터 길이를 먼저 찾고, 두 값을 더해 다음에 기대하는 바이트 번호를 구한다. 세그먼트 헤더의 기존 ACK 값은 반대 방향 데이터에 대한 확인 값인지 구분한다.

암기 포인트

TCP 누적 ACK는 '받은 마지막 번호'가 아니라 '다음에 받을 번호'이며, 데이터만 있는 정상 세그먼트에서는 ACK=SEQ+데이터 길이이다.

NoteBase