2018년 계리직 9급 컴퓨터일반 풀이

by 호이호이꿀떡

정답 체크

01	02	03	04	05	06	07	08	09	10
②	②	④	①	①	④	②	②	③	④
11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
③	②	①	④	①	③	④	③	③	①

문 1. 다음에서 설명하는 입·출력 장치로 옳은 것은?

> ◦ 중앙처리장치로부터 입·출력을 지시받은 후에는 자신의 명령어를 실행시켜 입·출력을 수행하는 독립된 프로세서이다.
> ◦ 하나의 명령어에 의해 여러 개의 블록을 입·출력할 수 있다.

① 버스(Bus)
② 채널(Channel)
③ 스풀링(Spooling)
④ DMA(Direct Memory Access)

답 ②

② 채널과 DMA 모두 CPU와 별도로 자신의 명령어를 통해 입출력을 실행하는 방식이다.
하나의 명령어에 의해 여러 개의 블록을 입출력이 가능한 것은 채널 방식이고, DMA는 하나의 명령어를 통해 하나의 입출력만 가능하다.

<오답 체크> ③ 스풀링과 버퍼링은 CPU와 입출력장치의 속도 차이를 해소하기 위한 방법으로, 데이터를 입력하거나 출력할 때 미리 임시공간에 저장하여 입출력장치가 데이터를 처리하는 동안 CPU는 다른 일을 병행할 수 있도록 해준다.
스풀링(Spooling)은 보조기억장치(디스크)를 임시 저장공간으로 사용하고, 버퍼링(Buffering)은 주기억장치를 임시 저장공간으로 사용한다.

문 2. 고객계좌 테이블에서 잔고가 100,000원에서 3,000,000원 사이인 고객들의 등급을 '우대고객'으로 변경하고자 <보기>와 같은 SQL문을 작성하였다. ㉠과 ㉡의 내용으로 옳은 것은?

> 〈 보 기 〉
> UPDATE 고객계좌
> (㉠) 등급 = '우대고객'
> WHERE 잔고 (㉡) 100000 AND 3000000

	㉠	㉡
①	SET	IN
②	SET	BETWEEN
③	VALUES	IN
④	VALUES	BETWEEN

답 ②

데이터를 수정, 변경하고자 할 땐 **UPDATE ~ SET ~** 문을 사용한다.
WHERE 절을 통해 범위를 지정할 땐, **BETWEEN ~ AND ~** 문을 사용한다.

문 3. 네트워크 장치에 대한 설명으로 옳지 <u>않은</u> 것은?

① 허브(Hub)는 여러 대의 단말 장치가 하나의 근거리 통신망(LAN)에 접속할 수 있도록 지원하는 중계 장치이다.
② 리피터(Repeater)는 물리 계층(Physical Layer)에서 동작하며 전송 신호를 재생·중계해 주는 증폭 장치이다.
③ 브리지(Bridge)는 데이터 링크 계층(Data Link Layer)에서 동작하며 같은 MAC 프로토콜(Protocol)을 사용하는 근거리 통신망 사이를 연결하는 통신 장치이다.
④ 게이트웨이(Gateway)는 네트워크 계층(Network Layer)에서 동작하며 동일 전송 프로토콜을 사용하는 분리된 2개 이상의 네트워크를 연결해주는 통신 장치이다.

답 ④

④ 네트워크 계층에서 동작하며 같은 프로토콜을 사용하는 네트워크 간의 최적의 경로를 설정하는 장치는 라우터(router)이다.
게이트웨이(gateway)는 전송 계층부터 응용 계층 사이에서 동작하며 프로토콜 변환을 통해 프로토콜이 다른 네트워크를 연결하는 장치이다.

문 4. ㉠에 들어갈 용어로 옳은 것은?

> (㉠) (은)는 유사한 문제를 해결하기 위해 설계들을 분류하고 각 문제 유형별로 가장 적합한 설계를 일반화하여 체계적으로 정리해 놓은 것으로 소프트웨어 개발에서 효율성과 재사용성을 높일 수 있다.

① 디자인 패턴
② 요구사항 정의서
③ 소프트웨어 개발 생명주기
④ 소프트웨어 프로세스 모델

답 ①

◈ **디자인 패턴**(Design pattern)
객체 지향 프로그래밍 설계를 할 때 자주 발생하는 문제들을 피하기 위해 사용되는 패턴

문 5. 결합도(Coupling)는 모듈 간의 상호 의존 정도 또는 모듈 간의 연관 관계를 의미한다. 아래에 나타낸 결합도를 약한 정도에서 강한 정도 순으로 올바르게 나열한 것은?

ㄱ. 내용 결합도(Content Coupling)
ㄴ. 제어 결합도(Control Coupling)
ㄷ. 자료 결합도(Data Coupling)
ㄹ. 공통 결합도(Common Coupling)

① ㄷ - ㄴ - ㄹ - ㄱ
② ㄷ - ㄹ - ㄱ - ㄴ
③ ㄹ - ㄴ - ㄷ - ㄱ
④ ㄹ - ㄷ - ㄱ - ㄴ

답 ①

◈ **결합도**(coupling) 낮은(좋은) 순서대로
▶ 자료 결합도(Data Coupling): 모듈간의 인터페이스가 자료 요소로만 구성된 경우. 파라미터를 통해서만 모듈간의 상호 작용이 일어나는 경우. Call by value
▶ 스탬프 결합도(Stamp Coupling): 모듈간의 인터페이스로 배열이나 오브젝트, 스트럭쳐등이 전달되는 경우
▶ 제어 결합도(Control Coupling): 단순히 처리를 해야 할 대상인 값만 전달되는 게 아니라 어떻게 처리를 해야 한다는 제어 요소(DCD, Flag등)이 전달되는 경우.
▶ 외부 결합도(External Coupling): 어떤 모듈에서 반환한 값을 다른 모듈에서 참조해서 사용하는 경우
▶ 공통 결합도(Common Coupling): 파라미터가 아닌 모듈 밖에 선언되어 있는 전역 변수를 참조하고 전역변수를 갱신하는 식으로 상호작용하는 경우
▶ 내용 결합도(Content Coupling): 다른 모듈 내부에 있는 변수나 기능을 다른 모듈에서 사용 하는 경우

◈ **응집도**(cohesion) 높은(좋은) 순서대로
▶ 기능적 응집도(Functional Cohesion): 모듈 내부의 모든 기능 요소들이 단일한 목적을 위해 수행되는 경우
▶ 순차적 응집도(Sequential Cohesion): 모듈 내에서 한 기능 요소로부터 나온 출력값을 다른 기능 입력값으로 사용할 경우
▶ 교환적(통신적) 응집도(Communication Cohesion): 동일한 입력과 출력을 사용하여 다른 기능을 수행하는 활동들이 모여있을 경우
▶ 절차적 응집도(Procedural Cohesion): 모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성요소들이 그 기능을 순차적으로 수행할 경우
▶ 시간적 응집도(Temporal Cohesion): 연관된 기능이라기보다 특정 시간에 처리되어야 하는 활동들을 한 모듈에서 처리할 경우
▶ 논리적 응집도(Logical Cohesion): 유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들이 한 모듈에서 처리되는 경우
▶ 우연적 응집도(Coincidental Cohesion): 모듈 내부의 각 구성요소들이 연관이 없는 경우

문 6. 컴퓨터 알고리즘에 대한 설명으로 옳지 않은 것을 <보기>에서 모두 고른 것은?

〈 보 기 〉

ㄱ. 힙 정렬(Heap Sort) 알고리즘의 시간 복잡도는 $O(n^2)$이다.

ㄴ. 0/1 배낭(0/1 Knapsack) 문제에 대하여 다항시간(Polynomial time) 내에 해결 가능한 알고리즘이 개발되었다.

ㄷ. 모든 NP(Non-deterministic Polynomial time) 문제는 컴퓨터를 이용하여 다항시간에 해결할 수 없다.

① ㄱ ② ㄱ, ㄴ

③ ㄴ, ㄷ ④ ㄱ, ㄴ, ㄷ

답 ④

<오답 체크> ㄱ. 정렬 알고리즘 시간복잡도(Time complexity)

알고리즘	최선	평균	최악
삽입 정렬	O(n)	$O(n^2)$	$O(n^2)$
선택 정렬	$O(n^2)$	$O(n^2)$	$O(n^2)$
버블 정렬	$O(n^2)$	$O(n^2)$	$O(n^2)$
셸 정렬	O(n)	$O(n^{1.5})$	$O(n^2)$
퀵 정렬	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n^2)$
힙 정렬	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$
합병 정렬	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$
기수 정렬	O(dn)	O(dn)	O(dn)

ㄴ. 0/1 배낭 문제에 대해 다항시간 내에 해결 가능한 알고리즘은 아직 개발되지 않았다.

ㄷ. P 문제(Polynomial problem)는 결정론적 튜링머신으로 다항시간 내에 풀 수 있는 문제이며, NP 문제(Non-deterministic Polynomial problem)는 비 결정론적 튜링머신으로 다항시간 내에 풀 수 있는 문제

NP 문제 중 NP-hard 문제는 다항시간 내에 풀기 매우 어려운 문제를 말하며, NP-complete는 다항시간 내에 풀 수 있는 알고리즘이 없다고 알려진 문제들을 말한다.

따라서 NP-hard나 NP-complete가 아닌 NP 문제들은 다항시간 내에 해결이 가능하므로 보기는 틀린 설명이다.

문 7. JAVA 프로그램의 실행 결과로 옳은 것은?

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

① 0 ② 200 ③ 404 ④ 600

답 ②

'a >> 2' 는 시프트 명령어로, 데이터 스트림 a를 오른쪽으로 2칸 밀어내는 명령어이다.

a는 01100101 이므로,

오른쪽으로 2칸 밀면 00011001 = 25가 된다.(왼쪽에서 새로 추가되는 자리에는 0이 들어간다)

그 다음 '<< 3' 명령어를 통해 왼쪽으로 3칸 밀어내어,

11001000 = 200 이 된다.

cf) 프로그램 데이터는 2진수로 표현되기 때문에, 오른쪽으로 한 칸 밀면 나누기 2가 되고, 왼쪽으로 한 칸 밀면 곱하기 2가 된다. (2로 나눌 때 소수점 이하는 절삭한다. 10진수를 이용해 테스트 해보면 금방 이해가 될 것이다.)

따라서 위의 문제는 처음 101에서 나누기 2를 두 번 하면,

101 -> 50 -> 25 가 되고,

그 다음 곱하기 2를 세 번 하면 25 -> 50 -> 100 -> 200 이 된다.

문 8. 암호 방식에 대한 설명으로 옳은 것을 <보기>에서 모두 고른 것은?

〈 보 기 〉

ㄱ. 대칭키 암호 방식(Symmetric Key Cryptosystem)은 암호화 키와 복호화 키가 동일하다.

ㄴ. 공개키 암호 방식(Public Key Cryptosystem)은 사용자 수가 증가하면 관리해야 할 키의 수가 증가하여 키 변화의 빈도가 높다.

ㄷ. 대칭키 암호 방식은 공개키 암호 방식에 비하여 암호화 속도가 빠르다.

ㄹ. 공개키 암호 방식은 송신자와 발신자가 서로 같은 키를 사용하여 통신을 수행한다.

① ㄱ, ㄴ ② ㄱ, ㄷ ③ ㄴ, ㄷ ④ ㄴ, ㄹ

답 ②

ㄱ. 대칭키 암호 방식은 암호화 키와 복호화 키가 동일하며, 공개키 암호 방식은 암호화 키와 복호화 키가 다르다.

ㄷ. 대칭키 암호 방식이 공개키 암호 방식에 비해 암호화 속도가 빠르며 키의 길이가 짧다.

<오답 체크> ㄴ. 공개키 암호 방식에서는 사용자 수가 증가하더라도 (개인 한 사람이 각자) 관리해야 할 키의 수는 공개키와 개인키 2개뿐이다.

반면 대칭키 방호 방식에서는 개인은 각 사용자들마다 공유해야 할 키가 모두 다르기 때문에, 전체 사용자 수가 늘어날수록 개인이 관리해야 할 키의 수는 증가하게 된다. (전체 사용자가 n일 때, 한 사람이 관리해야 할 키의 수는 n-1개)

ㄹ. (해당 보기의 지문은 잘못되었다. 송신자와 발신자는 같은 의미이기 때문이다. 문제의 의도는 송신자와 수신자였을 것이다.)

공개키 암호 방식에서는 송신자와 수신자는 서로 다른 키를 사용하여 통신을 수행한다.

공개키를 이용하여 암호화를 할 경우 송신자는 공개키를 사용하고 수신자는 개인키를 사용한다.

공개키를 이용하여 전자서명을 할 경우 송신자는 개인키를 사용하고 수신자는 공개키를 사용한다.

문 9. 학생 테이블에 튜플들이 아래와 같이 저장되어 있을 때, <NULL, '김영희', '서울'> 튜플을 삽입하고자 한다. 해당 연산에 대한 [결과]와 [원인]으로 옳은 것은?(단, 학생 테이블의 기본키는 학번이다.)

학번	이름	주소
1	김철희	경기
2	이철수	천안
3	박민수	제주

	[결과]		[원인]
①	삽입 가능	-	무결성 제약조건 만족
②	삽입 불가	-	관계 무결성 위반
③	삽입 불가	-	개체 무결성 위반
④	삽입 불가	-	참조 무결성 위반

답 ③

③ 학번은 기본키이기 때문에 중복값이나 NULL 값이 들어갈 수 없다. 따라서 삽입이 불가능하며 이는 개체 무결성 위반에 해당한다.

▷ **개체 무결성 제약조건**(Entity Integrity Constraint)
릴레이션의 기본 키를 구성하는 어떤 애트리뷰트도 널(NULL)값을 가질 수 없고 튜플을 유일하게 식별해야 한다.

▷ **도메인 무결성 제약조건**(Domain Constraint)
각 애트리뷰트 값이 반드시 원자값이어야 하며, 허용된 유형의 데이터 값들만 입력이 가능하다.

▷ **참조 무결성 제약조건**(Referential Integrity Constraint)
두 개의 릴레이션이 기본키·외래키를 통해 참조 관계를 형성할 경우, 참조하는 외래키의 값은 참조되는 릴레이션에 기본키로 존재해야 한다.

▷ **키 제약조건**(Key Constraint)
애트리뷰트에 중복된 값이 존재해서는 안 된다.

▷ **사용자 정의 무결성 제약조건**(user define integrity constraint)
사용자가 다른 무결성 범주에 포함되지 않는 특정 업무 규칙을 정의하여 사용한다.

문 10. 10진수 –2.75를 아래와 같이 IEEE 754 표준에 따른 32비트 단정도 부동소수점(Single Precision Floating Point) 표현 방식에 따라 2진수로 표기했을 때 옳은 것은?

부호	지수부	가수부

(부호: 1비트, 지수부: 8비트, 가수부: 23비트)

① 1000 0000 0000 0000 0000 0000 0000 1011
② 1000 0000 1011 0000 0000 0000 0000 0000
③ 1010 0000 0110 0000 0000 0000 0000 0000
④ 1100 0000 0011 0000 0000 0000 0000 0000

답 ④

IEEE 754 표준 단정도는

$$(-1)^S \times 1.F \times 2^E$$ 로 표현된다.

S는 부호 1비트, E는 지수부 8비트, F는 가수부 23비트
단 지수부는 127 바이어스(bias)법을 따른다.

10진수 -2.75를 2진수로 바꾸면 -10.11이다.

$-10.11 = -1.011 \times 2^1$

음수(-)이기 때문에 부호는 1
지수부는 1이며, 127 바이어스법을 따르기 때문에 127을 더하면 128, 즉 10000000 이다.
가수부는 소수점 01100000000000000000000 이다.

따라서 정답은 1 / 10000000 / 01100000000000000000000

문 11. ㉠에 들어갈 용어로 옳은 것은?

> 주기억장치의 물리적 크기의 한계를 해결하기 위한 기법으로 주기억장치의 크기에 상관없이 프로그램이 메모리의 주소를 논리적인 관점에서 참조할 수 있도록 하는 것을 (㉠)라고 한다.

① 레지스터(Register)
② 정적 메모리(Static Memory)
③ 가상 메모리(Virtual Memory)
④ 플래시 메모리(Flash Memory)

답 ③

③ **가상메모리**(Virtual Memory)
실제 물리적인 메모리 공간 주기억장치(RAM) 외에 하드 디스크에 파일 형태로 따로 준비하는 가상의 메모리 공간으로, 부족한 시스템 메모리를 보조해주는 역할을 한다.
가상 메모리는 실제 주기억장치에 존재하는 게 아니며 논리적인 주소 공간, 논리적인 관점이라는 표현을 쓴다.

<오답 체크> ① **레지스터**(Register)
프로세스 내에서 처리 중인 데이터를 저장하는 아주 빠른 저장공간이다.

② **정적 메모리**(Static Memory)
프로그램 실행 중 저장공간이 해제되거나 재배치되지 않고 고정되어 있는 저장공간이다.

④ **플래시 메모리**(Flash Memory)
전기적으로 데이터를 지우고 다시 쓸 수 있는 비휘발성 저장장치이다.

문 12. C 프로그램의 실행 결과로 옳은 것은?

```
#include <stdio.h>
int main(    )
{
    int i, sum=0;
    for(i=1; i<=10; i+=2) {
        if(i%2 && i%3) continue;
        sum += i;
    }
    printf("%d\n", sum);
    return 0;
}
```

① 6 ② 12 ③ 25 ④ 55

답 ②

for문에서 i는 1부터 시작해 2씩 증가해서 10과 같거나 작은 동안 반복된다.

따라서 i = 1, 3, 5, 7, 9 이렇게 다섯 번 실행한다.

if (i%2 && i%3) 에서 'i%2'는 i를 2로 나누고 남은 나머지를 의미하며, 0은 false를 0 이외의 수는 true를 의미한다.

&&는 and 논리 연산자로 i%2와 i%3이 모두 true일 때는 전체가 true가 되어 continue 명령어를 실행하여 아랫줄을 건너뛴다.

하나라도 false일 경우 전체가 false가 되어 아랫줄을 실행한다.

따라서 i를 하나씩 실행하면

i = 1) 1%2 = 1(true), 1%3 = 1(true)
if (true && true) = true -> continue 명령어 실행

i = 3) 3%2 = 1(true), 3%3 = 0(false)
if (true && false) = false -> 아래 sum += i 명령문 실행
sum에 3을 더하여 3이 된다.

i = 5) 5%2 = 1(true), 5%3 = 2(true)
if (true && true) = true -> continue 명령어 실행

i = 7) 7%2 = 1(true), 7%3 = 1(true)
if (true && true) = true -> continue 명령어 실행

i = 9) 9%2 = 1(true), 9%3 = 0(false)
if (true && false) = false -> 아래 sum += i 명령문 실행
sum에 9을 더하여 12가 된다.

i = 11) 11이 되면 for문을 빠져나간다.

그래서 결과적으로 sum = **12를 출력**한다.

문 13. 다음에서 설명하는 소프트웨어 개발 방법론으로 옳은 것은?

> 프로세스와 도구 중심이 아닌 개발 과정의 소통을 중요하게 생각하는 소프트웨어 개발 방법론으로 반복적인 개발을 통한 잦은 출시를 목표로 한다.

① 애자일 개발 방법론
② 구조적 개발 방법론
③ 객체지향 개발 방법론
④ 컴포넌트 기반 개발 방법론

답 ①

① **애자일 개발 방법론**(Agile software development)

일정한 주기를 가지고 끊임없이 반복적으로 프로토타입을 만들어 내며 그때 그때 필요한 요구를 더하고 수정하여 하나의 커다란 소프트웨어를 개발해 나가는 방법론이다.

애자일 개발 프로세스란 어느 특정 개발 방법론을 가리키는 말은 아니고, 애자일(Agile=기민한, 좋은것을 빠르고 낭비없게 만드는 것) 개발을 가능하게 해 주는 다양한 방법론 전체를 일컫는 말이다. 대표적인 기법으로 익스트림 프로그래밍 (XP, eXtreme Programming)이 있다.

<오답 체크> ② **구조적 개발 방법론**(Structed Development)

정형화된 분석 절차에 따라 요구사항을 파악하고, 전체 시스템을 기능과 흐름에 따라 분할하고 이를 통합하는 방식의 개발 방법론이다.

③ **객체지향 개발 방법론**(Object Oriented Development)

현실 세계에서의 개체를 속성과 메소드를 결합시킨 형태로 표현하여 객체간의 메시지 통신을 통해 시스템을 구현하는 개발 방법론이다. 추상화, 캡슐화, 정보은폐, 상속성, 다형성 등의 특징을 가지며 재사용과 이식이 용이하다.

④ **컴포넌트 기반 개발 방법론**
(CBD, Component Based Development)

컴포넌트 단위로 개발하여 각 컴포넌트를 조립·통합하여 시스템을 개발하는 방법론이다. 각 컴포넌트는 재사용이과 확장이 용이하여 빠른 개발과 비용 절감이 가능하다.

문 14. 불 대수(Boolean Algebra)에 대한 최소화로 옳지 않은 것은?

① A (A + B) = A
② A + A` B = A + B
③ A (A` + B) = AB
④ AB + AB` + A` B = A

답 ④

④ A B + A B` + A` B = A (B + B` ) + A`B
= A · 1 + A` B
= A + A` B
= (A + A`) (A + B)
= 1 · (A + B)
= **A + B**

<오답 체크> ① A (A + B) = A (불 대수 흡수법칙)
② A + A` B = ( A + A`)·(A + B)
= 1·(A + B) = A + B
③ A A` + A B = 0 + A B
= A B

문 15. 배열(Array)과 연결리스트(Linked List)에 대한 설명으로 옳지 않은 것은?

① 연결리스트는 배열에 비하여 희소행렬을 표현하는데 비효율적이다.
② 연결리스트에 비하여 배열은 원소를 임의의 위치에 삽입하는 비용이 크다.
③ 연결리스트에 비하여 배열은 임의의 위치에 있는 원소를 접근할 때 효율적이다.
④ n개의 원소를 관리할 때, 연결리스트가 n 크기의 배열보다 메모리 사용량이 더 크다.

답 ①

① 희소행렬이란 행렬의 값 중 상당수가 0으로 구성되어 있는 행렬을 의미한다. 예를 들어, 3X4 행렬의 12개 원소들 중 8개 정도가 0의 값을 가진 행렬이다.
배열을 이용해 희소행렬을 저장하려면 12개 모든 메모리 공간을 할당해 저장해야 하지만, 연결 리스트를 이용해 저장할 땐 값이 있는 4개만 링크로 연결하여 저장할 수 있다.(할당되지 않은 위치는 0을 의미) 따라서 연결 리스트를 이용하는 것이 배열에 비해 효율적이다.

<오답 체크> ② 연결 리스트에서는 중간에 노드만 연결해주면 되므로 비용이 크게 들지 않는다.
반면 배열은 순서대로 메모리가 이미 할당되어 있으므로, 중간에 값을 삽입하기 위해서는 뒤의 값들을 모두 한 칸씩 밀어내 저장하여야 하므로 비용이 더 많이 든다.

③ 연결 리스트에서는 임의의 주소값을 계산할 수 없기 때문에 앞에서부터 하나씩 연결된 노드를 타고 찾아가야 한다.
반면 배열은 임의의 위치에 접근하기 위해서는 몇 번째 위치인지만 알면 바로 주소값을 계산해 접근이 가능하기 때문에 더 효율적이다.

④ 똑같은 n개의 원소를 저장할 때 배열은 데이터 크기만큼만 할당받으면 되지만, 연결 리스트는 다음 원소의 주소를 가리키는 노드값을 추가로 저장해야 한다. 따라서 연결 리스트가 메모리 사용량이 더 크다.

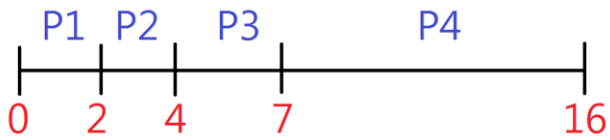
문 16. 프로세스 P1, P2, P3, P4를 선입선출(First In First Out) 방식으로 스케줄링을 수행할 경우 평균응답시간으로 옳은 것은? (단, 응답시간은 프로세스 도착시간부터 처리가 종료될 때까지의 시간을 말한다)

프로세스	도착시간	처리시간
P1	0	2
P2	2	2
P3	3	3
P4	4	9

① 3 ② 4 ③ 5 ④ 6

답 ③

FIFO(First In First Out): 준비상태 큐에 도착한 순서에 따라 차례로 CPU를 할당



1. (0~2초) 가장 먼저 들어온 P1을 수행한다.
2. (2~4초) 두 번째로 들어온 P2를 수행한다.
3. (4~7초) 세 번째로 들어온 P3를 수행한다.
4. (7~16초) 마지막으로 들어온 P4를 수행한다.

프로세스	도착	완료	응답시간
P1	0	2	2
P2	2	4	2
P3	3	7	4
P4	4	16	12

평균 응답시간 = (2 + 2 + 4 + 12) / 4 = 20 / 4 = **5초**

문 17. TCP/IP 프로토콜에 대한 설명으로 옳은 것은?

① TCP는 비연결형 프로토콜 방식을 사용한다.
② TCP는 네트워크 계층(Network Layer)에 속한다.
③ IP는 잘못 전송된 패킷에 대하여 재전송을 요청하는 기능을 제공한다.
④ IP는 각 패킷의 주소 부분을 처리하여 패킷이 목적지에 도달할 수 있도록 한다.

답 ④

④ IP의 주요 기능 중 하나인 라우팅 기능에 대한 설명이다.

<오답 체크> ①② TCP는 OSI 4계층 전송계층에서 작동하는 신뢰성, 연결 지향형의 프로토콜이다.
전이중 전송방식/양방향성(full-duplex)의 특성을 가지고 있으며, 흐름제어, 혼잡제어 등의 기능을 수행한다.
UDP와 IP가 비연결형 프로토콜이다.

③ IP는 OSI 3계층 네트워크 계층 프로토콜로, 비연결형이며 신뢰성을 보장하지 않는다. 인터넷 경로설정을 위한 라우팅 역할을 수행하며, 패킷의 분해(단편화), 조립, 논리적 경로지정 기능을 제공한다.
순서 검사(sequence checking)와 흐름 제어, 오류 복구 기능 등이 없다.

문 18. 다음에서 설명하는 용어로 가장 옳은 것은?

> 프랭크 로젠블라트(Frank Rosenblatt)가 고안한 것으로 인공신경망 및 딥러닝의 기반이 되는 알고리즘이다.

① 빠른 정렬(Quick Sort)
② 맵리듀스(MapReduce)
③ 퍼셉트론(Perceptron)
④ 디지털 포렌식(Digital Forensics)

답 ③

◈ **인공신경망**(ANN, Artificial Neural Network)
생물학의 신경망에서 영감을 얻은 통계학적 학습 알고리즘으로, 시냅스의 결합으로 네트워크를 형성한 인공 뉴런(노드)이 학습을 통해 시냅스의 결합 세기를 변화시켜 문제 해결 능력을 가지는 기계학습 모델이다.

▷ **퍼셉트론**(Perceptron)은 뉴런의 수학적 모델을 일컫는 용어이기도 하고, 최초로 제안된 신경망 프로그램 알고리즘이기도 하다. 퍼셉트론은 선형함수를 이용해 하나의 뉴런을 사용하며 학습 데이터를 가장 잘 설명할 수 있는 최적의 패러미터 찾는다.
각 노드의 가중치와 입력치를 곱한 것을 모두 합한 값이 활성함수에 의해 판단되는데, 그 값이 임계치(보통 0)보다 크면 뉴런이 활성화되고 결과값으로 1을 출력한다. 뉴런이 활성화되지 않으면 결과값으로 -1을 출력한다.
그러나 (단층) 퍼셉트론은 비선형함수를 학습할 수 없으며, 입력층과 출력층만 있기에 XOR 문제도 해결할 수 없다.

▷ **다층 퍼셉트론**(**MLP,** Multi Layer Perceptron)
여러 개의 퍼셉트론을 연결시켜 층(Layer)을 만들고, 이 층들을 중첩시켜 다층(Multi Layer)으로 만든 것으로, 단층 퍼셉트론에서 문제가 되었던 XOR 연산이 가능하다고 한다.
입력층(Input layer), 은닉층(Hidden layer), 출력층(Output layer) 으로 구분되어 있으며, 필요이상으로 많은 층을 두는 것은 오히려 성능이 떨어진다고 알려져 있다.

▷ 입력층(Input Layer): 외부로부터 입력자료를 받아들여 시스템으로 전달

▷ 은닉층(Hidden Layer): 시스템 안쪽에 위치하고 있으며, 입력값을 넘겨받아 처리한 뒤 결과를 산출

▷ 출력층(Output Layer): 입력값과 시스템 상태를 기준하여 시스템 출력값을 산출

<오답 체크> ② **맵리듀스**(MapReduce)
구글에서 대용량 데이터 처리를 분산 병렬 컴퓨팅을 통해 처리하기 위한 목적으로 제작한 소프트웨어 프레임워크

문 19. 관계형 데이터베이스의 뷰(View)에 대한 장점으로 옳지 <u>않은</u> 것은?

① 뷰는 데이터의 논리적 독립성을 일정 부분 제공할 수 있다.
② 뷰를 통해 데이터의 접근을 제어함으로써 보안을 제공할 수 있다.
③ 뷰에 대한 연산의 제약이 없어서 효율적인 응용프로그램의 개발이 가능하다.
④ 뷰는 여러 사용자의 상이한 응용이나 요구를 지원할 수 있어서 데이터 관리를 단순하게 한다.

답 ③

③ 뷰에 대한 검색은 기본 테이블과 거의 동일하지만, 삽입, 갱신, 삭제 연산에는 제약이 있다.

<오답 체크> ②④ 뷰를 통하면 데이터베이스에 직접 접근하지 않고도 데이터를 활용할 수 있기 때문에 보안이 제공되며 데이터베이스 구조에 구애받지 않아 다양한 응용 프로그램을 지원할 수 있다.

문 20. 다음에서 설명하는 알고리즘 설계 기법으로 가장 알맞은 것은?

> 해결하고자 하는 문제의 최적해(Optimal Solution)가 부분 문제들의 최적해들로 구성되어 있을 경우, 이를 이용하여 문제의 최적해를 구하는 기법이다.

① 동적 계획법(Dynamic Programming)
② 탐욕적 알고리즘(Greedy Algorithm)
③ 재귀 프로그래밍(Recursive Programming)
④ 근사 알고리즘(Approximation Algorithm)

답 ①

① **동적 계획법**(Dynamic Programming)이란 복잡한 문제를 간단한 여러 개의 문제로 나누어 푸는 방법을 말한다. 이것은 부분 문제 반복과 최적 부분 구조를 가지고 있는 알고리즘을 일반적인 방법에 비해 더욱 적은 시간 내에 풀 때 사용한다.

<오답 체크> ② **탐욕적 알고리즘**(Greedy Algorithm)
여러 경우 중 하나를 결정할 때마다 그 순간에 최적이라고 생각되는 것을 선택해 나가는 방식으로 진행하여 최종적인 해답에 도달하는 알고리즘이다. 매 순간마다의 선택은 최적이지만, 그것을 모두 통합한 전체가 최적이라는 보장은 없다.

③ **재귀 프로그래밍**(Recursive Programming)
자기 자신을 호출하는 형태의 함수 알고리즘을 말하며, 특정 기능을 반복적으로 수행해야 할 필요가 있을 때 사용한다.

④ **근사 알고리즘**(Approximation Algorithm)
어떤 최적화 문제에 대한 근사값을 구하는 알고리즘이다.
정확히 최적화값을 구할 수는 없지만, 비교적 빠른 시간에 계산이 가능하며 어느 정도 보장된 근사해를 구할 수 있다. NP-완전 문제 등 최적화 알고리즘 알려지지 않은 문제에 적용하여 근사값을 구할 수 있다.