

컴퓨터일반

문 1. 컴퓨팅 사고(Computational Thinking)에서 주어진 문제의 중요한 특징만으로 문제를 간결하게 재정의함으로써 문제 해결을 쉽게 하는 과정은? 3

- ① 분해 ② 알고리즘
③ 추상화 ④ 패턴 인식

[해설]

※ 컴퓨팅 사고(Computational Thinking)

- 컴퓨터로 처리할 수 있는 형태로 문제와 해결책을 표현하는 사고 과정.
- 컴퓨터에게 뭘 해야 할지를 사람이 설명해 주는 것이라고 볼 수 있다.
- 복잡한 문제를 컴퓨터가 효과적으로 처리할 수 있는 단위로 분해 (decomposition), 문제 간 유사성을 찾는 패턴 인식(pattern recognition), 문제의 핵심만 추려 복잡한 문제를 단순화하는 추상화(abstraction), 일련의 규칙과 절차에 따라 문제를 해결하는 알고리즘 기법 등을 사용하여 사람과 컴퓨터 모두가 문제를 처리할 수 있는 형태로 표현한다.

[저자/제공처] 한국정보통신기술협회

※ 추상화(Abstraction)

- ① 복잡한 문제를 이해하기 위해서 필요없는 세부 사항을 배제하는 것을 의미

- ② 복잡한 구조(문제)를 해결하기 위하여 설계 대상의 상세내용은 배제하고 유사점을 요약해서 표현하는 기법

- ③ 구체적인 데이터의 내부구조를 외부에 알리지 않으면서 데이터를 사용하는 데 필요한 함수만을 알려주는 기법

- #### ④ 추상화의 유형

- ⑦ 기능(functional) 추상화

- 입력자료를 출력자료로 변환하는 과정의 추상화
- 모듈, 서브루틴, 함수에 의해 절차를 추상화
- 절차 지향언어는 함수와 함수간 부프로그램을 정의 시에 유용
- 객체지향언어는 Method를 정의 시에 유용

- ㉔ 자료(Data) 추상화

- 자료와 자료에 적용될 수 있는 기능을 함께 정의함으로써 자료 객체를 구성하는 방법

- 어떤 데이터 개체들에 대한 연산을 정의함으로써 데이터형이나 데이터 대상을 정의하며, 그 표현과 처리내용은 은폐하는 방법

- ㉔ 제어(Control) 추상화

- 제어의 정확한 메커니즘을 정의하지 않고 원하는 효과를 정하는데 이
용

문 2. 소프트웨어에 대한 설명으로 옳지 않은 것은? 4

- ① 하드웨어에 대응하는 개념으로 우리가 원하는 대로 컴퓨터를 작동하게 만드는 논리적인 바탕을 제공한다.
- ② 운영체제 등 컴퓨터 시스템을 가동시키는 데 사용되는 소프트웨어를 시스템 소프트웨어라 한다.
- ③ 문서 작성이나 게임 등 특정 분야의 업무를 처리하는 데 사용되는 소프트웨어를 응용 소프트웨어라 한다.
- ④ 고급 언어로 작성된 프로그램을 한꺼번에 번역한 후 실행하는 것이 인터프리터 방식이다.

[해설]

- ### ① 컴파일러

- 컴파일 언어로 작성된 원시프로그램을 준기계어로 번역한 후 목적 프로그램을 출력하는 번역기이다.
- 원시언어가 고급언어이고, 목적언어가 실제 기계언어에 가까운 저급언어

어인 번역기이다.

- ② 인터프리터(Interpreter)

- 고급언어를 기계어로 하는 컴퓨터를 하드웨어로 구성하는 대신에 이 고급언어를 기계에서 실행되도록 소프트웨어로 시뮬레이션하여 구성하는 방법이다.
- 원시프로그램을 구성하는 각 명령을 기계어로 번역하여 즉시 실행시키는 것으로 별도의 목적프로그램을 만들지는 않는다.

문 3. 4GHz의 클럭 속도를 갖는 CPU에서 CPI(Cycle per Instruction)가 4.0이고 총 10^{10} 개의 명령어로 구성된 프로그램을 수행하려고 할 때, 이 프로그램의 실행 완료를 위해 필요한 시간은? 2

- [illegible]

[해설]

- CPU의 속도 : $4\text{GHz} = 4 * 10^9$
- 수행하려는 프로그램 : $4 * 10^{10}$
- $4 * 10^{10} / 4 * 10^9 = 10$

문 4. -35 를 2의 보수(2's Complement)로 변환하면? 2

- ① 11011100 ② 11011101
③ 11101100 ④ 11101101

[해설]

- ① -35 를 2진수로 변환 : 10100011
- ② 10100011 를 1의 보수로 변환 : 11011100
- ③ 11011100 을 2의 보수로 변환 : 11011101

문 5. OSI 7계층에서 계층별로 사용하는 프로토콜의 데이터 단위는 다음 표와 같다. ㉠ ~ ㉣에 들어갈 내용을 바르게 연결한 것은? 3

계층	데이터 단위
트랜스포트(Transport) 계층	(㉠)
네트워크(Network) 계층	(㉡)
데이터링크(DataLink) 계층	(㉢)
물리(Physical) 계층	비트

- | ㉠ | ㉡ | ㉢ |
|--------|------|------|
| ① 세그먼트 | 프레임 | 패킷 |
| ② 패킷 | 세그먼트 | 프레임 |
| ③ 세그먼트 | 패킷 | 프레임 |
| ④ 패킷 | 프레임 | 세그먼트 |

[해설]

* OSI 7계층과 TCP/IP 프로토콜에서의 캡슐화

OSI 7 Layer	Data		TCP/IP 4 Layer
Application	Message		Application
Presentation			
Session			
Transport	Segment	TCP Header	Transport
Network	Packet (Datagram)	IP Header	Internet
Data Link	Frame	Frame Header	Network Access
Physical	Bit (Signal)		

문 6. 300개의 노드로 이진 트리를 생성하고자 할 때, 생성 가능한 이진 트리의 최대 높이와 최소 높이로 모두 옳은 것은? (단, 1개의 노드로 생성된 이진 트리의 높이는 1이다) 4

	최대 높이	최소 높이
①	299	8
②	299	9
③	300	8
④	300	9

[해설]

- 1개의 노드로 생성된 이진 트리의 높이가 1일 경우에 n개의 노드를 가지는 이진트리의 깊이는 최대 n 이고, 최소 $\lceil \log_2(n+1) \rceil$ 이다.
- 최대 : 300
- 최소 : $\lceil \log_2(300+1) \rceil = \lceil \log_2(301) \rceil = \lceil 8. xxx... \rceil = 9$

문 7. 아래와 같은 순서대로 회의실 사용 요청이 있을 때, 다음 중 가장 많은 회의실 사용 시간을 확보할 수 있는 스케줄링 방법은? (단, 회의실은 하나이고, 사용 요청은 (시작 시각, 종료 시각)으로 구성된다. 회의실에 특정 회의가 할당되면 이 회의 시간과 겹치는 회의 요청에 대해서는 회의실 배정을 할 수 없다) 4

(11:50, 12:30),	(9:00, 12:00),	(13:00, 14:30),
(14:40, 15:00),	(14:50, 16:00),	(15:40, 16:20),
(16:10, 18:00)		

- ① 시작 시각이 빠른 요청부터 회의실 사용이 가능하면 확정한다.
- ② 종료 시각이 빠른 요청부터 회의실 사용이 가능하면 확정한다.
- ③ 사용 요청 순서대로 회의실 사용이 가능하면 확정한다.
- ④ 회의 시간이 긴 요청부터 회의실 사용이 가능하면 확정한다.

[해설]

- 문제의 사용 요청 (시작 시각, 종료 시각)을 확인하면 가장 많은 회의실 사용시간을 확보할 수 있는 스케줄링 방법은 회의 시간이 긴 요청부터 회의실 사용이 가능하면 확정하는 방법이다.
- (9:00, 12:00) ⇨ (13:00, 14:30) ⇨ (14:50, 16:00) ⇨ (16:10, 18:00)

문 8. 제품 테이블에 대하여 SQL 명령을 실행한 결과가 다음과 같을 때, ㉠과 ㉡에 들어갈 내용을 바르게 연결한 것은? 1

<제품 테이블>			
제품ID	제품이름	단가	제조업체
P001	나사못	100	A
P010	망치	1,000	B
P011	드라이버	3,000	B
P020	망치	1,500	C
P021	장갑	800	C
P022	너트	200	C
P030	드라이버	4,000	D
P031	절연테이프	500	D

<SQL 명령>	
SELECT 제조업체, MAX(단가) AS 최고단가	
FROM 제품	
GROUP BY (㉠)	
HAVING COUNT(*) > (㉡) ;	

<실행 결과>	
제조업체	최고단가
B	3,000
C	1,500
D	4,000

㉠	㉡
① 제조업체	1
② 제조업체	2
③ 단가	1
④ 단가	2

[해설]

- 문제의 실행결과를 볼 때, 제조업체별로 그룹을 나누고 그룹으로 묶인 제조업체의 튜플수가 1 초과(2 이상) 인 것만 제조업체와 최고단가를 검색한다.

SELECT [ALL DISTINCT 열_리스트(검색 대상)]
FROM 테이블_리스트
[WHERE 조건]
[GROUP BY 열_이름 [HAVING 조건]]

- GROUP BY : 그룹으로 나누어준다.
- HAVING : 그룹에 대한 조건, GROUP BY에서 사용.

문 9. 스택의 입력으로 4개의 문자 D, C, B, A가 순서대로 들어올 때, 스택 연산 PUSH와 POP에 의해서 출력될 수 없는 결과는? 2

- ① ABCD
- ② BDCA
- ③ CDBA
- ④ DCBA

[해설]

- 스택의 입력으로 4개의 문자 D, C, B, A가 순서대로 들어올 때, 보기 2번의 BDCA 순으로 출력될 수 없다. B를 가장 먼저 출력하기 위해서는 D C B 까지 먼저 PUSH 가 되어야하고, B를 POP 한 상태에서 top에 C가 들어있으므로 D가 먼저 POP 될 수 없다.

문 10. 임계구역에 대한 설명으로 옳은 것은? 4

- ① 임계구역에 진입하고자 하는 프로세스가 무한대기에 빠지지 않도록 하는 조건을 진행의 융통성(Progress Flexibility)이라 한다.
- ② 자원을 공유하는 프로세스들 사이에서 공유자원에 대해 동시에 접근하여 변경할 수 있는 프로그램 코드 부분을 임계영역(Critical Section)이라 한다.
- ③ 한 프로세스가 다른 프로세스의 진행을 방해하지 않도록 하는 조건을 한정 대기(Bounded Waiting)라 한다.
- ④ 한 프로세스가 임계구역에 들어가면 다른 프로세스는 임계구역에 들어갈 수 없도록 하는 조건을 상호 배제(Mutual Exclusion)라 한다.

[해설]

- ① 임계구역에 진입하고자 하는 프로세스가 무한대기에 빠지지 않도록 하는 조건을 한정 대기(Bounded Waiting)라 한다.
- ② 한 프로세스가 임계구역에 들어가면 다른 프로세스는 임계구역에 들어갈 수 없다.

③ 한 프로세스가 다른 프로세스의 진행을 방해하지 않도록 하는 조건을 진행의 융통성(Progress Flexibility)이라 한다.

- ※ 임계구역 해결 조건
- 상호 배제 (mutual exclusion) : 한 프로세스가 임계구역에 들어가면 다른 프로세스는 임계구역에 들어갈 수 없다.
 - 한정 대기 (bounded waiting) : 어떤 프로세스도 무한 대기 (infinite postpone)에 빠지지 않도록 해야 한다.
 - 진행의 융통성 (progress flexibility) : 한 프로세스가 다른 프로세스의 진행을 방해해서는 안된다.

문 11. 통합 테스트 방법에 대한 설명으로 옳지 않은 것은? 1

- ① 연쇄식(Threads) 통합은 초기에 시스템 골격을 파악하기 어렵다.
- ② 빅뱅(Big-bang) 통합은 모든 모듈을 동시에 통합하여 테스트한다.
- ③ 상향식(Bottom-up) 통합은 가장 하부 모듈부터 통합하여 테스트한다.
- ④ 하향식(Top-down) 통합은 프로그램 제어 구조에서 상위 모듈부터 통합하는 것을 말한다.

[해설]

- ① 하향식 통합
- 주프로그램으로부터 그 모듈이 호출하는 다음 레벨의 모듈을 테스트하고, 점차적으로 하위 모듈로 이동하는 방법
 - 드라이버는 필요치 않고 통합이 시도되지 않은 곳에 스텝이 필요, 통합이 진행되면서 스텝은 실제 모듈로 교체
- ② 상향식 통합
- 시스템 하위 레벨의 모듈로부터 점진적으로 상위 모듈로 통합하면서 테스트하는 기법
 - 스텝은 필요치 않고 드라이버가 필요
- ③ 연쇄식(Threads) 통합
- 특수하고 중요한 기능을 수행하는 최소 모듈 집합을 먼저 구현하고 보조적인 기능의 모듈은 나중에 구현하여 테스트한 후 계속 추가한다.
 - 제일 먼저 구현되고 통합될 모듈은 중심을 이루는 기능을 처리하는 모듈의 최소 집합이다. 이렇게 점차적으로 구축된 스레드에 다른 모듈을 추가시켜 나간다.

문 12. 다음 중 파이썬 프로그래밍 언어에 대한 설명으로 옳은 것만을 모두 고르면? 3

- ㄱ. 변수 선언 시 변수명 앞에 데이터형을 지정해야 한다.
- ㄴ. 플랫폼에 독립적인 대화식 언어이다.
- ㄷ. 클래스를 정의하여 객체 인스턴스를 생성할 수 있다.

- ① ㄴ
- ② ㄱ, ㄷ
- ③ ㄴ, ㄷ
- ④ ㄱ, ㄴ, ㄷ

[해설]

- ※ 파이썬(Python)
- ① 파이썬은 1991년 프로그래머인 귀도 반 로섬(Guido van Rossum)이

- 발표한 고급 프로그래밍 언어이다.
- ② 플랫폼에 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어이다.
- ③ 파이썬의 특징
- 문법이 쉽고 간단하며, 배우기가 쉽다.
 - 객체 지향적이다.
 - 다양한 패키지가 제공된다.
 - 오픈 소스이며 무료로 제공된다.

문 13. 해쉬(Hash)에 대한 설명으로 옳지 않은 것은? 3

- ① 연결리스트는 체이닝(Chaining) 구현에 적합하다.
- ② 충돌이 전혀 없다면 해쉬 탐색의 시간 복잡도는 O(1)이다.
- ③ 최악의 경우에도 이진 탐색보다 빠른 성능을 보인다.
- ④ 해쉬 함수는 임의의 길이의 데이터를 입력받을 수 있다.

[해설]

- ※ 해쉬(Hash)
- 해쉬는 다른 레코드의 키 값과 비교할 필요가 없는 탐색방법이다.
 - 탐색 시간의 복잡도는 O(1)이지만, 충돌이 발생하면 O(n)이 된다.
 - 해쉬함수 : 제곱법, 제산법, 폴딩법, 자릿수 분석법
- ※ 이진탐색
- 파일이 정렬되어있어야 하며, 파일의 중앙의 키값과 비교하여 탐색 대상이 반으로 감소된다.
 - 탐색시간이 적게 걸리지만, 삽입과 삭제가 많을 때는 적합하지 않고 고정된 데이터에 적합하다.
 - 수행 시간의 차수는 O(logn)이다.

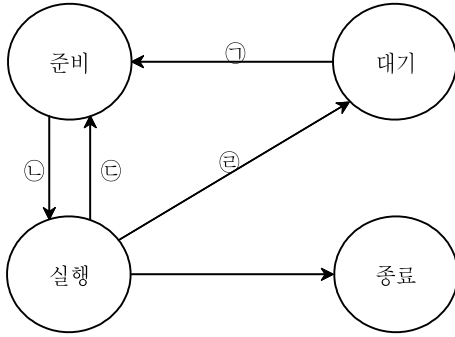
문 14. 프로세스의 메모리는 세그멘테이션에 의해 그 역할이 할당되어 있다. 표준 C언어로 작성된 프로그램이 컴파일 후 실행파일로 변환되어 메모리를 할당받았을 때, 이 프로그램에 할당된 세그먼트에 대한 설명으로 옳은 것은? 4

- ① 데이터 세그먼트는 모든 서브루틴의 지역변수와 서브루틴 종료 후 돌아갈 명령어의 주소값을 저장한다.
- ② 스택은 현재 실행 중인 서브루틴의 매개변수와 프로그램의 지역변수를 저장한다.
- ③ 코드 세그먼트는 CPU가 실행할 명령어와 메인 서브루틴의 지역변수를 저장한다.
- ④ 힙(Heap)은 동적 메모리 할당을 위해 사용되는 공간이고, 주소값이 커지는 방향으로 증가한다.

[해설]

- 코드(Code)영역 : 소스 코드 자체를 기계어로 변환하여 저장되는 영역이다.
- 데이터(Data) 영역: 전역 변수/정적(static) 변수가 할당되는 영역이다. (초기화된 전역 변수와 정적 변수가 저장)
- BSS(Block Stated Symbol) 영역 : 초기화 안된 전역 변수와 정적 변수 저장되는 영역이다.
- 스택(Stack) 영역 : 함수 호출 시 생성되는 지역 변수와 매개 변수가 저장되는 영역이다.
- 힙(Heap) 영역 : 필요에 따라 동적으로 메모리를 할당 할 때 사용한다.

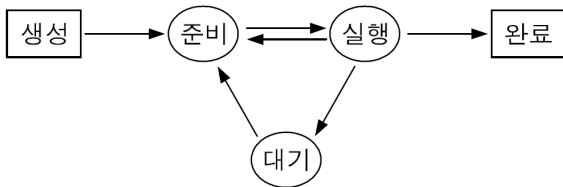
문 15. 다음은 프로세스 상태 전이도이다. 각 상태 전이에 대한 예로 적절하지 않은 것은? 4



- ① ㉠ - 프로세스에 자신이 기다리고 있던 이벤트가 발생하였다.
 ② ㉡ - 실행할 프로세스를 선택할 때가 되면, 운영체제는 프로세스들 중 하나를 선택한다.
 ③ ㉢ - 실행 중인 프로세스가 자신에게 할당된 처리기의 시간을 모두 사용하였다.
 ④ ㉣ - 실행 중인 프로세스가 작업을 완료하거나 실행이 중단되었다.

[해설]

※ 프로세스 상태 전이도



- 생성(New) 상태 : 작업이 제출되어 스펙 공간에 수록
- 준비(Ready) 상태 : 중앙처리장치가 사용 가능한(할당할 수 있는) 상태
- 실행(Running) 상태 : 프로세스가 중앙처리장치를 차지(프로세스를 실행)하고 있는 상태
- 대기(Block) 상태 : I/O와 같은 사건으로 인해 중앙처리장치를 양도하고 I/O 완료 시까지 대기 큐에서 대기하고 있는 상태
- 완료(Exit) 상태 : 중앙처리장치를 할당받아 주어진 시간 내에 수행을 종료한 상태

※ 프로세스의 상태 전환

- Dispatch(준비상태 → 실행상태) : 준비 상태의 프로세스들 중에서 우선순위가 가장 높은 프로세스를 선정하여 중앙처리장치를 할당함으로써 실행상태로 전환
- Timer runout(실행상태 → 준비상태) : 중앙처리장치의 지정된 할당 시간을 모두 사용한 프로세스를 다른 프로세스를 위해 다시 준비 상태로 전환
- Block(실행상태→대기상태) : 실행중인 프로세스가 입출력 명령을 만나면 입출력 전용 프로세서에게 중앙처리장치를 스스로 양도하고 자신은 대기 상태로 전환
- Wake up(대기상태→준비상태) : 입출력 완료를 기다리다가 입출력 완료 신호가 들어오면 대기 중인 프로세스는 준비 상태로 전환

문 16. -30.25×2^{-8} 의 값을 갖는 IEEE 754 단정도(Single Precision) 부동소수점(Floating-point) 수를 16진수로 변환하면? 3

- ① 5DF30000 ② 9ED40000
 ③ BDF20000 ④ C8F40000

[해설]

※ 32비트 부동소수점 표현 - Single Precision(단정도, float)
 - 지수는 8비트, 가수는 23비트이며, 바이어스는 127이다.

	지수부(8비트)	가수부(23비트)
--	----------	-----------

└ 부호비트(1비트) : 0(양수), 1(음수)

- ① -30.25 을 2진수로 변환 : -11110.01
 ② -11110.01×2^{-8}
 ③ 정규화 수행 : -1.111001×2^{-4}
 ④ 부호비트는 1(음수)
 ⑤ 지수부 : $(-4 + 127)_{10} = (123)_{10} = (01111011)_2$
 ⑥ 가수부 : 1110010000000000000000 (가장 왼쪽 1. 은 생략하고, 나머지 가수부 비트는 0으로 채운다)
 ⑦ 10111101111100100000000000000000를 16진수로 변환 : BDF20000

문 17. 다음은 어느 학생이 C 언어로 작성한 학점 계산 프로그램이다. 출력 결과는? 3

```
#include <stdio.h>

int main()
{
    int score = 85;
    char grade;
    if (score >= 90) grade='A';
    if (score >= 80) grade='B';
    if (score >= 70) grade='C';
    if (score < 70) grade='F';
    printf("학점 : %c\n", grade);
    return 0;
}
```

- ① 학점 : A ② 학점 : B
 ③ 학점 : C ④ 학점 : F

[해설]

- 문제의 코드에서 if (score >= 80) grade='B'; 의 조건이 만족하여 변수 grade 의 'B' 가 되지만, 아래에 있는 if (score >= 70) grade='C'; 도 조건이 만족하므로 변수 grade 는 'C' 가 저장되어 출력된다.

문 18. 파이프라인 해저드(Pipeline Hazard)에 대한 다음 설명에서 ㉠과 ㉡에 들어갈 내용을 바르게 연결한 것은? 1

- 하드웨어 자원의 부족 때문에 명령어를 적절한 클록 사이클에 실행할 수 있도록 지원하지 못할 때 (㉠) 해저드가 발생한다.
- 실행할 명령어를 적절한 클록 사이클에 가져오지 못할 때 (㉡) 해저드가 발생한다.

㉠

㉡

- ① 구조적 제어
- ② 구조적 데이터
- ③ 데이터 구조적
- ④ 데이터 제어

[해설]

※ 파이프라인 해저드

파이프라인에서 명령어 실행이 불가하여 지연, 중지가 발생하는 현상을 말하며, 파이프라인 해저드는 구조적 해저드, 데이터 해저드, 제어 해저드로 구분할 수 있다.

① 구조적 해저드

- 포트가 하나인 메모리에 동시에 접근하려고 하거나 ALU 등의 하드웨어를 동시에 사용하려고 할 때 발생할 수 있다.
- 메모리를 명령어 영역과 데이터 영역을 분리하여 사용한다든지, ALU 등의 하드웨어를 여러 개 사용하는 것 등을 통해 해결할 수 있다.

② 데이터 해저드

- 이전 명령어에서 레지스터의 값을 바꾸기 전에 후속 명령어가 그 값을 읽거나 쓰려고 하는 경우와 같이 사용하는 데이터의 의존성이 있는 경우 발생한다.

- 명령어 재배치나 전방전달(Data forwarding), No-operation insertion 등을 통해 해결할 수 있다.

③ 제어 해저드

- 조건 분기로 인해 명령어의 실행 순서가 변경되어 명령어가 무효화되는 것을 말한다.
- 분기 예측이나 Stall 삽입법 등을 통해 해결할 수 있다.

문 19. 합성곱 신경망(CNN, Convolutional Neural Network) 처리 시 다음과 같은 입력과 필터가 주어졌을 때, 합성곱에 의해 생성된 특징 맵(Feature Map)의 ㉠에 들어갈 값은? 2

1	1	0	0
1	1	1	1
0	1	0	0
1	1	0	1

입력

1	0	0
0	1	1
1	1	1

필터

(㉠)	

특징 맵

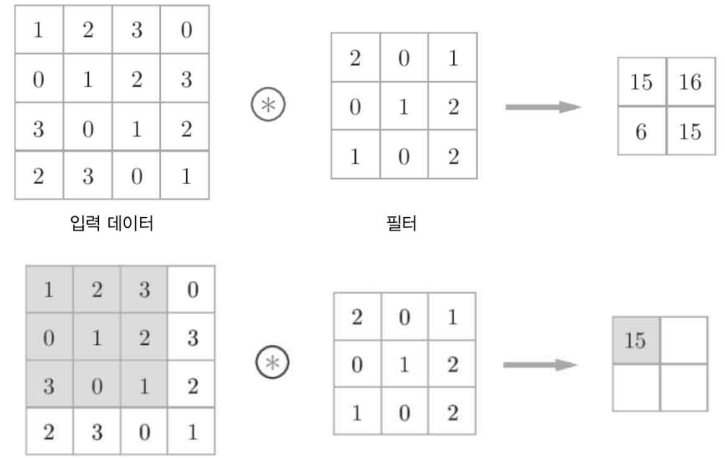
- ① 3
- ② 4
- ③ 5
- ④ 6

[해설]

※ CNN(Convolution Neural Network, 합성곱 신경망)

- 컴퓨터 영상처리에 적합한 합과 곱의 연산이 있는 신경망이다.
- 어떤 특성값들을 합한 뒤 곱하는 신경망이다.
- 합성곱 연산 : 입력데이터를 필터라는 행렬과 연산하여 사용한다.

ex)



문 20. 해밍코드에 대한 패리티 비트 생성 규칙과 인코딩 예가 다음과 같다. 이에 대한 설명으로 옳은 것은? 4

<패리티 비트 생성 규칙>

원본 데이터	d4	d3	d2	d1			
인코딩된 데이터	d4	d3	d2	p4	d1	p2	p1

$$p1 = (d1 + d2 + d4) \bmod 2$$

$$p2 = (d1 + d3 + d4) \bmod 2$$

$$p4 = (d2 + d3 + d4) \bmod 2$$

<인코딩 예>

원본 데이터	0	0	1	1			
인코딩된 데이터	0	0	1	1	1	1	0

- ① 이 방법은 홀수 패리티를 사용하고 있다.
- ② 원본 데이터가 0100이면 010110으로 인코딩된다.
- ③ 패리티 비트에 오류가 발생하면 복구는 불가능하다.
- ④ 수신측이 0010001을 수신하면 한 개의 비트 오류를 수정한 후 최종적으로 0010으로 복호한다.

[해설]

- ① 문제의 방법은 짝수 패리티를 사용하고 있다.
- ② 원본 데이터가 0100이면 0101010으로 인코딩된다.
- ③ 패리티 비트에 오류가 발생하여도 복구는 가능할 수 있다.

※ 해밍코드(hamming code)

- 오류 검출과 교정이 가능하다.
- 2의 거듭제곱 번째 위치에 있는 비트들은 패리티 비트로 사용한다. (1, 2, 4, 8, 16...번째 비트)
- 나머지 비트에는 부호화될 데이터가 들어 간다. (3, 5, 6, 7, 9, 10, 11, 12 ...번째 비트)

P1	P2		P3				P4			...
----	----	--	----	--	--	--	----	--	--	-----

- P1의 패리티 값 : 1, 3, 5, 7, 9, 11, ... 번째 비트들의 짝수(또는 홀수) 패리티 검사 수행
- P2의 패리티 값 : 2, 3, 6, 7, 10, 11, ... 번째 비트들의 짝수(또는 홀수) 패리티 검사 수행
- P3의 패리티 값 : 4, 5, 6, 7, 12, 13, 14, 15, ... 번째 비트들의 짝수(또는 홀수) 패리티 검사 수행
- P4의 패리티 값 : 8, 9, 10, 11, 12, 13, 14, 15, 24, 25, ... 번째 비트들의 짝수(또는 홀수) 패리티 검사 수행