2017-서울시-컴퓨터일반-A형-해설-곽후근

1. 다음은 컴퓨터 언어처리에 관련된 시스템 S/W의 기능을 설명한 것이다. 옳지 않은 것은?
① 컴파일러 : 고급언어를 이진목적모듈로 변환기능
② 어셈블러 : 객체지향언어를 이진목적모듈로 변환기능
③ 링커 : 여러 목적모듈을 통합하여 실행 가능한 하나의 모듈로 변환기능
④ 로더 : 실행 가능한 모듈을 주기억장치에 탑재기능
정답 체크 :
(2) 어셈블러 : 해당 설명은 컴파일러이고, 어셈블러는 어셈블리어를 이진파일(실행파일)로 변환한다.
오답 체크 :
(1) 컴파일러 : 고급언어(소스코드)를 이진파일(실행파일) 또는 어셈블리어로 변환한다.
(3) 링커 : 소스코드를 컴파일 한 이진파일과 미리 컴파일 한 라이브러리 이진 파일을 통합하여 실행 가능한 하나의 파일로 만든다.
(4) 로더 : 링커를 통해 만들어진 이진파일(실행파일)을 실행하기 위해 주기억장치에 올린다.

2. 컴퓨터에서 사건이 발생하면 이를 처리하기 위해 인터럽트 기술을 사용한다. 사건의 발생지에 따라 동기와 비동기 인터럽트로 분류된다. 다음 중 비동기 인터럽트는?
① 프로세스가 실행 중에 0으로 나누기를 할 때 발생하는 인터럽트
② 키보드 혹은 마우스를 사용할 때 발생하는 인터럽트
③ 프로세스 내 명령어 실행 때문에 발생하는 인터럽트
④ 프로세스 내 명령어가 보호 메모리영역을 참조할 때 발생 하는 인터럽트
정답 체크 :
동기 인터럽트는 프로세서의 명령어를 실행한 결과로 발생하는 인터럽트이다.
비동기 인터럽트는 명령어와 관련 없는 이벤트에 의해서도 발생하는 인터럽트이다.
(2) 키보드 혹은 마우스는 명령어와 관련 없는 이벤트이므로 비동기 인터럽트이다.
오답 체크 :
(1) 실행 중에 0으로 나누는 것은 명령어 실행의 결과이므로 동기 인터럽트이다.
(3) 명령어 실행 때문에 발생하는 인터럽트는 동기 인터럽트이다.
(4) 보호 메모리 영역을 참조하는 것은 명령어 실행의 결과이므로 동기 인터럽트이다.

3. 데이터통신에서 에러 복구를 위해 사용되는 Go-back-N ARQ에 대한 설명으로 옳지 않은 것은?
① Go-back-N ARQ는 여러 개의 프레임들을 순서번호를 붙여서 송신하고, 수신 측은 이 순서번호에 따라 ACK 또는 NAK를 보낸다.
② Go-back-N ARQ는 송신 측은 확인응답이 올 때까지 전송된 모든 프레임의 사본을 갖고 있어야 한다.
③ Go-back-N ARQ는 재전송 시 불필요한 재전송 프레임 들이 존재하지 않는다.
④ Go-back-N ARQ는 송신 측은 n개의 Sliding Window를 가지고 있어야 한다.
정답 체크 :

(3) 오류가 발생한 프레임과 그 이후에 전송한 모든 프레임을 다시 재전송을 하기 때문에 불필요한 재전송 프레임들이 존재한다.

오답 체크 :

(1) ACK에는 수신이 예상되는 다음 프레임의 번호를 전달하고, NAK에는 손상된 프레임 번호를 전달한다.

(2) 송신측은 확인응답이 올 때까지 전송된 모든 프레임의 사본을 갖는다.

(4) Go-back-N ARQ는 stop-and-wait 방식이 아니라 sliding window 방식이다.

4. 데이터베이스에서 뷰(View)에 대한 설명으로 옳은 것은?

① 뷰는 테이블을 기반으로 만들어지는 가상 테이블이며, 뷰를 기반으로 새로운 뷰를 생성할 수 없다.

② 뷰 삭제는 SQL 명령어 중 DELETE 구문을 사용하며, 뷰 생성에 기반이 된 기존 테이블들은 영향을 미치지 않는다.

③ 뷰 생성에 사용된 테이블의 기본키를 구성하는 속성이 포함되어 있지 않은 뷰도 데이터의 변경이 가능하다.

④ 뷰 생성 시 사용되는 SELECT문에서 GROUP BY 구문은 사용 가능하지만, ORDER BY 구문은 사용할 수 없다.

정답 체크 :

(4) SELECT 문은 생성하려는 뷰의 정의를 표현하며 ORDER BY는 사용이 불가하다.

오답 체크 :

(1) 뷰는 물리적인 테이블을 기반으로 만들어지는 논리적인 테이블이며, 뷰를 기반으로 새로운 뷰를 생성할 수 있다.

(2) 뷰 삭제는 DELETE 구문(뷰의 내용 삭제 시) 또는 DROP 구문(뷰 자체 삭제 시)을 사용하며, 뷰 생성에 기반이 된 기존 테이블들에 영향을 미친다.

(3) 뷰 생성에 사용된 테이블의 기본키를 구성하는 속성이 포함되어 있는 뷰만 데이터의 변경이 가능하다.

5. 초기에 빈 Binary Search Tree를 생성하고, 입력되는 수는 다음과 같은 순서로 된다고 가정한다. 입력되는 값을 이용하여 Binary Search Tree를 만들고 난 후 Inorder Traversal을 했을 때의 방문하는 순서는?

7, 5, 1, 8, 3, 6, 0, 2

① 01235678

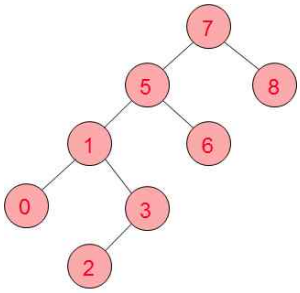
② 02316587

③ 75103268

④ 86230157

정답 체크 :

이진 탐색 트리에 원소를 삽입하기 위해서는 먼저 탐색을 수행하는 것이 필요하다. 탐색에 실패한 위치가 바로 새로운 노드를 삽입하는 위치이다. 주어진 조건으로 이진 탐색 트리를 구성하면 다음과 같다.



(1) inorder traversal(중위 순회)이다.

오답 체크 :

(2) postorder traversal(후위 순회)이다.

(3) preorder traversal(전위 순회)이다.

(4) 어떠한 순회도 아니다.

Tip! : 이진 탐색 트리를 중위 순회하면 오름차순으로 정렬된 값을 얻을 수 있다. 해당 원리를 알고 있었다면 이진 탐색 트리를 구성해서 중위 순회를 할 필요가 없이 바로 오름차순으로 정렬하면 원하는 답을 찾을 수 있다.

6. 통신 S/W 구조에서 제2계층인 데이터링크 계층의 주기능이 아닌 것은?

① 데이터링크 계층에서 전송할 프레임(Frame) 제작 기능

② 점대점(Point to Point) 링크 간의 오류제어 기능

③ 종단(End to End) 간 경로설정 기능

④ 점대점(Point to Point) 링크 간의 흐름제어 기능

정답 체크 :

(3) E2E 경로설정은 네트워크 계층에서 수행한다.

오답 체크 :

(1) 데이터링크 계층의 프로토콜 유닛은 프레임이다.

(2) P2P 오류제어 : ARQ(automatic repeat request)를 사용한다.

(4) P2P 흐름제어 : stop-and-wait 또는 sliding window를 사용한다.

7. 다음 중 집적도가 가장 높은 회로와 가장 큰 저장용량 단위를 나타낸 것은?

GB, PB, MB, TB VLSI, MSI, ULSI, SSI

① ULSI, PB

② VLSI, TB

③ MSI, GB

④ SSI, PB

정답 체크 :

(1)

저장용량을 순서대로 나열하면 다음과 같다.

bit 〈 Byte 〈 KB 〈 MB 〈 GB 〈 TB 〈 PB

집적도를 순서대로 나열하면 다음과 같다.

SSI 〈 MSI 〈 LSI 〈 VLSI 〈 ULSI

이를 기준으로 집적도가 가장 높은 회로는 ULSI이고, 가장 큰 저장용량 단위는 PB이다.
Tip! : 저장용량의 경우 더 작은 단위와 더 큰 단위로 시험에 자주 출제되므로 기억해두는 것이 좋다.

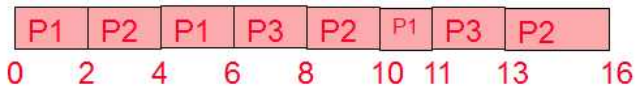
10^n	접두어	기호	배수	십진수
10^{24}	요타 (yotta)	Y	자	1 000 000 000 000 000 000 000 000
10^{21}	제타 (zetta)	Z	십해	1 000 000 000 000 000 000 000
10^{18}	엑사 (exa)	E	백경	1 000 000 000 000 000 000
10^{15}	페타 (peta)	P	천조	1 000 000 000 000 000
10^{12}	테라 (tera)	T	조	1 000 000 000 000
10^{9}	기가 (giga)	G	십억	1 000 000 000
10^{6}	메가 (mega)	M	백만	1 000 000
10^{3}	킬로 (kilo)	k	천	1 000
10^{2}	헥토 (hecto)	h	백	100
10^{1}	데카 (deca)	da	십	10
10^{0}			일	1
10^{-1}	데시 (deci)	d	십분의 일	0.1
10^{-2}	센티 (centi)	c	백분의 일	0.01
10^{-3}	밀리 (milli)	m	천분의 일	0.001
10^{-6}	마이크로 (micro)	μ	백만분의 일	0.000 001
10^{-9}	나노 (nano)	n	십억분의 일	0.000 000 001
10^{-12}	피코 (pico)	p	일조분의 일	0.000 000 000 001
10^{-15}	펨토 (femto)	f	천조분의 일	0.000 000 000 000 001
10^{-18}	아토 (atto)	a	백경분의 일	0.000 000 000 000 000 001
10^{-21}	젭토 (zepto)	z	십해분의 일	0.000 000 000 000 000 000 001
10^{-24}	욕토 (yocto)	y	일자분의 일	0.000 000 000 000 000 000 000 001

8. 다음은 다중스레드(Multi-Thread)에 관련된 설명이다. 옳지 않은 것은?
① 하나의 프로세스에 2개 이상의 스레드들을 생성하여 수행한다.
② 스레드별로 각각의 프로세스를 생성하여 실행하는 것보다 효율적이다.
③ 스레드들 간은 IPC(InterProcess Communication)방식 으로 통신한다.
④ 각각의 스레드는 프로세스에 할당된 자원을 공유한다.
정답 체크 :
(3) IPC는 프로세스 간 통신에 사용되고, 스레드 간 통신은 미리 공유한 자원를 통해 수행된다.
오답 체크 :
(1) 싱글스레디는 하나의 프로세스에 1개의 스레드를 생성하고, 다중(멀티)스레드는 하나의 프로세스에 2개 이상의 스레드들을 생성한다.
(2) 프로세스의 직접 실행 정보를 제외한 나머지 프로세스 관리 정보 공유하기 때문에 싱글스레드보다 효율적이다.
(4) 스레드는 자원을 공유하고, 실행(제어)를 분리한다.

9. 다음과 같이 3개의 프로세스가 있다고 가정한다. 각 프로 세스의 도착 시간과 프로세스의 실행에 필요한 시간은 아래 표와 같다. CPU 스케줄링 알고리즘으로 RR(Round Robin)을 사용한다고 가정한다. 3개의 프로세스가 CPU에서 작업을 하고 마치는 순서는? (단, CPU를 사용하는 타임 슬라이스(time slice)는 2이다.)

프로세스	도착시간	프로세스의 실행에 필요한 시간
P1	0	5
P2	1	7
P3	3	4

① P2, P1, P3
② P2, P3, P1
③ P1, P2, P3
④ P1, P3, P2
정답 체크 :
(4)
주어진 조건으로 간트 차트를 그리면 다음과 같다.



P2 다음에 P3가 아닌 P1이 온 이유는 스케줄링 시점 2에서 P3(3에 도착함)가 아직 도착하지 않아서 기존의 P1이 스케줄링 큐에 들어가게 되는 것이다. 마찬가지로 스케줄링 시점이 4에서 P3가 도착했으므로 P3를 스케줄링 큐에 넣는다. 이러한 RR 스케줄링을 기준으로 작업을 마치는 순서는 P1, P3, P2가 된다.

10. 다음의 C프로그램을 실행한 결과로 옳은 것은?

```
#include 〈stdio.h〉
void main()
{
  int num[4] = {1, 2, 3, 4};
  int *pt = num;
  pt++;
  *pt++ = 5;
  *pt ++= 10;
  pt--;
  *pt +++= 20;
  printf("%d %d %d %d", num[0], num[1], num[2], num[3]);
}
```

① 1 5 10 20
② 1 5 20 4
③ 1 5 30 4
④ 에러 발생
정답 체크 :
(3)
int *pt = num; // pt는 num[0]가 운명을 같이한다.
pt++; // pt의 주소가 증가되어 num[1]을 가리키게 된다.

*pt++ = 5; // 일단 num[1] = 5를 넣고, pt의 주소가 증가되어 num[2]를 가리키게 된다.
*pt ++=10; // 일단 num[2] = 10을 넣고, pt의 주소가 증가되어 num[3]를 가리키게 된다. pt와 ++가 분리된 것에 현혹되지 않는다. pt++과 pt ++은 같은 것이다.
pt--; // pt의 주소가 감소되어 num[2]를 가리키게 된다.
*pt +++=20; // num[2]의 기존 값 10에 20을 더하기 때문에 num[2]는 30이 되고, pt의 주소가 증가되어 num[3]를 가리키게 된다.
그러므로 num[0]은 바뀌지 않아서 1이고, num[1]은 5, num[2]는 30, 마지막으로 num[3]은 바꾸지 않아서 4이다.

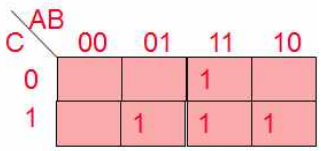
11. Flynn의 병렬컴퓨터 분류방식에 대한 설명으로 옳지 않은 것은?
① SISD - 명령어와 데이터를 순서대로 처리하는 단일프로세서 시스템이다.
② SIMD - 단일 명령어 스트림을 처리하고 배열프로세서라고도 한다.
③ MISD - 여러 개의 프로세서를 갖는 구조로 밀결합 시스템 (tightly - coupled system)과 소결합 시스템(loosely - coupled system)으로 분류한다.
④ MIMD - 여러 개의 프로세서들이 서로 다른 명령어와 데이터를 처리하는 진정한 의미의 병렬프로세서이다.
정답 체크 :
(3) MISD : 해당 설명은 MIMD이고, MISD는 처리장치들에서 수행되는 명령어는 다르지만, 전체적으로는 하나의 데이터 스트림을 가지게 되는 형태다.
오답 체크 :
(1) SISD : 한번에 한 개씩의 명령어와 데이터를 순서대로 처리하는 단일 프로세서 시스템이다.
(2) SIMD : 하나의 명령어 스트림(IS)이 다수의 처리장치들에서 동시 처리되는 기술이다. 벡터 프로세서(vector processor)와 배열 프로세서(array processor)가 대표적인 SIMD 분류에 속한다.
(4) MIMD : 다수의 처리장치가 서로 다른 명령어들을 동시에 병렬로 실행하는 형태로, 통상적인 일반 목적(general-purpose)의 다중 프로세서 구조다.

12. 네트워크 토폴로지(Topology) 중 버스(Bus) 방식에 대한 설명으로 옳지 않은 것은?
① 버스 방식은 네트워크 구성이 간단하고 작은 네트워크에 유용하며 사용이 용이하다.
② 버스 방식은 네트워크 트래픽이 많을 경우 네트워크 효율이 떨어진다.
③ 버스 방식은 통신 채널이 단 한 개이므로 버스 고장이 발생 하면 네트워크 전체가 동작하지 않으므로 여분의 채널이 필요하다.
④ 버스 방식은 노드의 추가·삭제가 어렵다.
정답 체크 :
(4) 버스는 노드를 공유 케이블에 연결하기만 하면 되기 때문에 노드의 추가 및 삭제가 쉽다. 다만, 서로 배치해야 하는 거리가 있어 재구성이 어렵고 결함이 어디에서 발생했는지 찾기가 어렵다(결함 분리).
오답 체크 :
(1) 구조가 간단하고 작은 네트워크에 유용하며 사용이 용이하다.
(2) 버스는 공유 케이블을 사용하는 구조이기 때문에 네트워크 트래픽이 많을 경우 충돌이 발생하여 네트워크 효율이 떨어진다.
(3) 버스는 공유 케이블을 사용하는 구조이기 때문에 버스 고장이 발생하면 네트워크 전체가 동작하

지 않는다.

13. 정보은닉(information hiding)에 대한 설명으로 옳지 않은 것은?
① 필요하지 않은 정보는 접근을 제한하는 것이다.
② 모듈 사이의 독립성을 유지시킨다.
③ 설계전략을 지역화하여 전략의 변경에 따른 영향을 최소화 한다.
④ 모듈 사이의 결합도를 높여 신뢰성을 향상시킨다.
정답 체크 :
(4) 정보은닉은 다른 모듈을 변경하지 못하기 때문에 모듈 사이의 결합도를 낮춘다.
오답 체크 :
(1) 정보은닉은 다른 객체가 한 객체 내의 데이터 값을 직접 참조하거나 접근할 수 없는 구조이다.
(2) 정보은닉은 다른 모듈을 변경하지 못하기 때문에 독립성을 유지시킨다.
(3) 정보은닉은 인터페이스와 구현의 명확한 분리하여 설계전략을 지역화하였다.

14. 다음 전가산기 논리회로에 대한 설명으로 옳지 않은 것은?
① 전가산기는 캐리를 포함하여 연산처리하기 위해 설계되었다.
② $S = (A \oplus B) \oplus C_i$
③ $C_0 = AB + AC_i + BC_i$
④ 전가산기는 두 개의 반가산기만으로 구성할 수 있다.
정답 체크 :
(4) 전가산기는 반가산기 2개와 OR 게이트로 구성된다.
오답 체크 :
(1) 반가산기는 전단계의 캐리를 포함하지 않고, 전가산기는 전단계의 캐리를 포함하여 덧셈을 수행한다.
(2) 그림에서 S를 계산하면 (A xor B) xor C_i가 된다.
(3) 그림에서 C_0를 계산하면 $(A \text{ xor } B)C_i + AB = (A'B + AB')C_i + AB = A'BC_i + AB'C_i + AB$가 된다. 부울식으로 전개해도 되지만 시간이 오래 걸리므로 바로 아래와 같이 카노맵으로 바꾼다. 해당 카노맵을 간략화하면 $C_0 = BC_i + AC_i + AB$가 된다.



Tip! : 부울식을 간략화 하는 것은 수식으로 전개해도 되지만 시간이 오래 걸릴 경우 바로 카노맵을 이용하면 빠르게 간략화를 수행할 수 있다.

15. 다음은 그래프 순회에서 깊이 우선 탐색 방법에 대한 수행 순서를 설명한 것이다. (ㄱ)~(ㄹ)에 알맞은 내용으로 짝지어진 것은?

(1) 시작 정점 v를 결정하고 방문한다. (2) 정점 v에 인접한 정점 중에서 (2-1) 방문하지 않은 정점 w가 있으면 정점 v를 (ㄱ)에 (ㄴ)하고 w를 방문한다. 그리고 w

를 v로 하여 (2)를 수행한다.
(2-2) 방문하지 않은 정점이 없으면 (ㄱ)을/를 (ㄷ)하여 받은 가장 마지막 방문 정점을 v로 설정한 뒤 다시 (2)를 수행한다.
(3) (ㄹ)이/가 공백이 될 때까지 (2)를 반복한다.

	(ㄱ)	(ㄴ)	(ㄷ)	(ㄹ)
①	Stack	push	pop	Stack
②	Stack	pop	push	Queue
③	Queue	enQueue	deQueue	Queue
④	Queue	enQueue	deQueue	Stack

정답 체크 :
(1)
DFS(깊이 우선 탐색)의 구현 코드는 다음과 같다.

```
void dfs_mat(GraphType *g, int v)
{
    int w;
    visited[v] = TRUE;              // (1) 시작 정점 v를 결정하고 방문한다.
    printf("%d ", v);               //
    for(w=0; w<g->n; w++)           // (2) 정점 v에 인접한 정점 중에서
        if(g->adj_mat[v][w] && !visited[w])
            dfs_mat(g, w);          // (2-1) 방문하지 않은 정점 w가 있으면 정점 v를 스택에 push하고 w를 방문한다. 그리고 w를 v로 하여 (1)과 (2)를 다시 수행한다. w로 호출하는 순간 정점 v를 스택에 자동 push한다.
    // (2-2) 방문하지 않은 정점이 없으면 스택을 pop하여 가장 마지막 방문 정점을 v로 설정한 뒤 다시 (2)를 수행한다.
    // (3) 스택이 공백이 될 때까지 (2)를 반복한다.
}
```

Tip! : DFS(깊이 우선 탐색)은 스택을 사용하고, BFS(넓이 우선 탐색)은 큐를 사용한다.

16. 소프트웨어 개발 생명 주기(Software Development Life Cycle)의 순서로 옳은 것은?
① 계획 → 분석 → 설계 → 구현 → 테스트 → 유지보수
② 분석 → 계획 → 설계 → 구현 → 테스트 → 유지보수
③ 분석 → 계획 → 설계 → 테스트→ 구현 → 유지보수
④ 계획 → 설계 → 분석 → 구현 → 테스트 → 유지보수
정답 체크 :
(1)
SDLC(소프트웨어 개발 생명 주기)의 순서는 다음과 같다.
계획 : 사용자는 소프트웨어의 필요성을 파악하고, 이를 개발하기 위한 타당성을 먼저 검토한다. 소프트웨어 개발이 타당하다면, 사용자는 이를 제안 요청서에 의해 개발자에게 요청한다.
분석 : 요구 분석의 목적은 최종 시스템의 기능, 성능, 사용의 용이성, 이식성 등을 파악하는 것이다.
설계 : 설계에는 시스템 구조 설계, 프로그램 설계 그리고 인터페이스 설계 등이 있다. 시스템 구조 설계는 시스템을 이루는 각 모듈과의 관계와 전체적인 구조를 설계하는 것이고, 프로그램 설계는 각

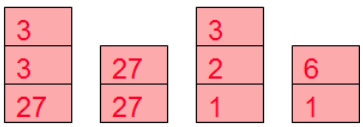
모듈 안에서의 처리 절차나 알고리즘을 설계하는 것을 말한다.
구현 : 프로그래밍을 하는 단계로 각 모듈의 코딩과 디버깅이 이루어지고, 그 결과를 검증하는 단위(모듈) 시험(unit test - test case)을 실시한다.
테스트 : 개발된 각 모듈들을 통합시키며 시험하는 통합 시험(integration test), 사용자의 요구사항(성능)을 만족하는지 알아보는 시스템 시험(system test), 그리고 사용자가 직접 자신의 사용 현장에서 시스템을 검증해 보는 인수 시험(acceptance test) 등이 있다.
유지보수 : 사용자가 개발된 소프트웨어를 인수(acceptance)하여 이용하면서 문제점을 발견하였을 경우, 이를 수정하거나 새로운 기능을 추가해 더욱 유용한 소프트웨어로 발전시키는 단계이다.

17. 다음은 postfix 수식이다. 이 postfix 수식은 스택을 이용 하여 연산을 수행한다. 그리고 ^는 지수함수 연산자이다. 처음 *(곱하기 연산) 계산이 되고 난 후 스택의 top과 top-1에 있는 두 원소는 무엇인가? (단, 보기의 (top) 은 스택의 top 위치를 나타낸다.)

```
27 3 3 ^ / 2 3 * -
```

① (top) 6, 1
② (top) 9, 1
③ (top) 2, 3
④ (top) 2, 7

정답 체크 :
(1)
주어진 조건을 스택을 이용해서 계산하면 다음과 같다. 27, 3, 3 오퍼랜드를 순서대로 스택에 쌓는다(push). ^ 연산자를 만나면 스택의 최상위 오퍼랜드 2개(3, 3)를 꺼내(pop) 계산한 후 다시 스택에 집어넣는다(push). / 연산자를 만나면 스택의 최상위 오퍼랜드 2개(27, 27)를 꺼내(pop) 계산한 후 다시 스택에 집어넣는다(push). 2, 3 오퍼랜드를 순서대로 스택에 쌓는다(push). * 연산자를 만나면 스택의 최상위 오퍼랜드 2개(2, 3)를 꺼내(pop) 계산한 후 다시 스택에 집어넣는다(push). * 계산이 되고 난 후 스택의 top에는 6이 있고, top-1에는 1이 있다.



18. 빅데이터에 대한 설명으로 옳은 것은?
① 빅데이터는 정형데이터로만 구성되며, 소셜 미디어 데이터는 해당되지 않는다.
② 빅데이터를 구현하기 위한 대표적인 프레임워크는 하둡이 있으며, 하둡의 필수 핵심 구성 요소는 맵리듀스(MapReduce) 와 하둡분산파일시스템(Hadoop Distributed File System)이다.
③ 빅데이터 처리과정은 크게 수집 → 저장 → 처리 → 시각화(표현) → 분석 순서대로 수행된다.
④ NoSQL은 관계 데이터 모델을 사용하는 RDBMS 중 하나이다.
정답 체크 :
(2) 하둡은 여러 개의 저렴한 컴퓨터를 마치 하나인 것처럼 묶어 대용량 데이터를 처리하는 기술이다. 하둡은 수천대의 분산된 x86 장비에 대용량 파일을 저장할 수 있는 기능을 제공하는 분산파일시스템과, 저장된 파일 데이터를 분산된 서버의 CPU와 메모리 자원을 이용해 쉽고 빠르게 분석할 수 있는 컴퓨팅 플랫폼인 맵리듀스로 구성돼 있다.

오답 체크 :
(1) 빅데이터는 정형데이터(단어)와 비정형데이터(소셜 미디어 데이터, 이미지, 동영상)로 구성되다.
(3) 수집 → 저장 → 처리 → 분석 → 시각화(표현)의 순서로 처리된다.
(4) NoSQL은 전통적인 관계형 데이터베이스 관리 시스템(RDBMS)과는 다르게 설계된 비관계형(non-relational) DBMS로, 대규모의 데이터를 유연하게 처리할 수 있는 것이 강점이다.

19. 최근 컴퓨팅 환경이 클라우드 환경으로 진화됨에 따라 가상화 기술이 중요한 기술로 부각되고 있다. 이에 대한 설명으로 옳지 않은 것은?
① 하나의 컴퓨터에 2개 이상의 운영체제 운용이 가능하다.
② VM(Virtual Machine)하에서 동작되는 운영체제(Guest OS)는 실 머신에서 동작되는 운영체제보다 효율적이다.
③ 특정 S/W를 여러 OS플랫폼에서 실행할 수 있어 S/W 이식성이 제고된다.
④ VM하에서 동작되는 운영체제(Guest OS)의 명령어는 VM명령어로 시뮬레이션되어 실행된다.
정답 체크 :
(2) VM에서 동작하는 운영체제는 실제 운영체제 위에서 동작하므로(시뮬레이션) 실제 머신에서 동작되는 운영체제보다 비효율적이다.
오답 체크 :
(1) 하나의 컴퓨터에 여러 개의 가상 머신(virtual machine)을 설치하면 2개 이상의 운영체제 운용이 가능하다.
(3) 가상 머신이 없을 때는 특정 S/W의 이식성을 체크하기 위해 여러 대의 컴퓨터에서 실행해야 했는데 가상 머신이 하나의 컴퓨터에서 여러 OS 플랫폼을 만들 수 있기 때문에 기존 방식에 비해 이식성 체크가 수월하다.
(4) VM은 실제 운영체제 위에서 동작하기 때문에 VM에서 실행하는 운영체제 명령어는 VM 명령어로 시뮬레이션 된다.

20. 다음 C프로그램의 실행 결과는?

```
#include <stdio.h>
void change(int *px, int *py, int pc, int pd);
void main(void) {
   int a=10, b=20, c=30, d=40;
   change(&a, &b, c, d);
   printf( a=%d b=%d c=%d d=%d , a, b, c, d);
}
void change(int *px, int *py, int pc, int pd) {
   *px = *py + pd;
   *py = pc + pd;
   pc = *px + pd;
   pd = *px + *py;
}
```

① a=60 b=70 c=50 d=30
② a=60 b=70 c=30 d=40

③ a=10 b=20 c=50 d=30
④ a=10 b=20 c=30 d=40
정답 체크 :
(2)
a = px = 20 + 40 = 60; // a는 인수의 주소가 함수로 복사되므로 참조에 의한 호출(call-by-reference)이다. 그러므로 a는 함수 내에서의 연산에 따라 값이 60으로 수정된다.
b = py = 30 + 40 = 70; // b는 인수의 주소가 함수로 복사되므로 참조에 의한 호출(call-by-reference)이다. 그러므로 b는 함수 내에서의 연산에 따라 값이 70으로 수정된다.
c = pc; // c는 인수의 값만이 함수로 복사되므로 값에 의한 호출(call-by-value)이다. 그러므로 c의 값은 변하지 않는다.
d = pd; // d는 인수의 값만이 함수로 복사되므로 값에 의한 호출(call-by-value)이다. 그러므로 d의 값은 변하지 않는다.
Tip! : 비슷한 문제가 나오면 call-by-value는 함수 호출 후에도 값의 변화가 없으므로 call-by-reference가 변하는지만 보면 된다.