2012년 계리직 컴퓨터일반 풀이

by 호이호이꿀떡

정답 체크

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

문 1. 컴퓨터 용어에 대한 설명으로 옳지 않은 것은?

① MIPS는 1초당 백만개 명령어를 처리한다는 뜻으로 컴퓨터의 연산 속도를 나타내는 단위이다.
② SRAM은 전원이 꺼져도 저장된 자료를 계속 보존할 수 있는 기억장치이다.
③ KB, MB, GB, TB 등은 기억 용량을 나타내는 단위로서 이중 TB가 가장 큰 단위이다.
④ SSI, MSI, LSI, VLSI 등은 칩에 포함되는 게이트의 집적도에 따라 구분된 용어이다.

답 ②

② SRAM(Static RAM, 정적램)은 RAM의 한 종류로, DRAM보다 빠르며 전원이 공급되는 한 그 내용이 계속 보존된다. 하지만 RAM의 특성상 전원이 꺼지면 저장된 자료는 지워진다.

<오답 체크> ① MIPS는 Million Instructions Per Second의 약자로, 단어 그대로 1초당 백만 개의 명령어를 처리하는 것을 뜻한다.

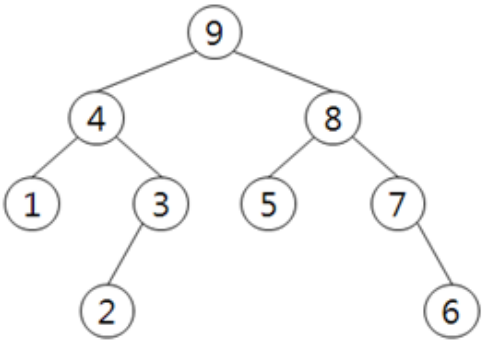
▶ 데이터 용량

10^24 요타 (yotta) 1YB
10^21 제타 (zetta) 1ZB
10^18 엑사 (exa) 1EB
10^15 페타 (peta) 1PB
10^12 테라 (tera) 1TB
10^9 기가 (giga) 1GB
10^6 메가 (mega) 1MB
10^3 킬로 (kilo) 1KB
1바이트 1Byte

▶ 집적도

SSI(small scale integration)	100개 이하
MSI(middle scale integration)	100~1,000개
LSI(large scale integration)	1,000~10,000개
VLSI(very large scale integration)	10,000~1,000,000개
ULSI(ultra large scale integration)	1,000,000개 이상

문 2. 이진트리의 순회(traversal) 경로를 나타낸 그림이다. 이와 같은 이진트리 순회방식은 무엇인가? 단, 노드의 숫자는 순회순서를 의미한다.



① 병렬 순회(parallel traversal)
② 전위 순회(pre-order traversal)
③ 중위 순회(in-order traversal)
④ 후위 순회(post-order traversal)

답 ④

④ 루트 노드를 가장 나중에 순회하는 것을 통해 후위 순회임을 알 수 있다.

<오답 체크> ① 병렬 순회는 없고, 레벨 순회가 있다.
레벨순회(level traversal)은 모든 노드를 낮은 레벨부터 차례대로 순회하는 것이다.
9 - 4 - 8 - 1 - 3 - 5 - 7 - 2 - 6
너비 우선 순회(breadth-first traversal)라고도 한다.

② 전위 순회
9 - 4 - 1 - 3 - 2 - 8 - 5 - 7 - 6

③ 중위 순회
1 - 4 - 2 - 3 - 9 - 5 - 8 - 7 - 6

문 3. 엑셀에서는 서로 다른 시트 사이에 셀 참조가 가능하다. 아래 그림에서 Sheet2의 시금치 가격을 VLOOKUP 함수를 사용하여 Sheet1에서 가져오고자 한다. 이를 위해 Sheet2의 B3 셀에 입력할 수식으로 알맞은 것은?

Sheet1

	A	B	C	D
1	상품명	산지	생산자	가격
2	오이	청주	김철수	500
3	배추	울산	황인용	2000
4	무우	김제	김영운	1500
5	시금치	평창	나윤로	1000
6	상추	대전	김윤철	700

Sheet2

	A	B
1	상품명	가격
2	무우	
3	시금치	
4		
5		
6		

① =VLOOKUP(시금치,Sheet1!A2:D6,4,0)
② =VLOOKUP(시금치,A2:A6,5,0)
③ =VLOOKUP(A3,Sheet1!A2:D6,4,0)
④ =VLOOKUP(A3,Sheet1!A2:A6,5,0)

답 ③

VLOOKUP 함수는 세로로 검색하여 원하는 데이터를 찾는 함수로 4개의 인수를 가지고 있다.
=VLOOKUP (찾고자 하는 값, 검색할 테이블 범위,
반환할 값이 들어있는 자리(열), 유사한 값 여부)

먼저 찾고자 하는 값은 시금치이며, 시트2의 A3셀의 값이다.
=VLOOKUP (A3, , ,)

값을 검색할 테이블 범위는 시트1의 A2:D6 이다.
=VLOOKUP (A3, Sheet1!A2:D6, ,)

반환받고자 하는 값은 가격이므로 검색할 테이블 범위의 4번째 열에 들어있는 값이다.
1은 상품명을, 2는 산지를, 3은 생산자를, 4는 가격을 반환한다.
=VLOOKUP (A3, Sheet1!A2:D6, 4,)

마지막 인수 '유사한 값 여부'는 0(FALSE)를 입력하면 검색값이 정확하게 일치해야만, 1(TRUE)를 입력하면 검색값이 유사한 것도 반환한다.
문제에서는 선택지 전부 0으로 쓰여 있으므로 답과는 상관없다.
=VLOOKUP(A3,Sheet1!A2:D6,4,0)

<오답 체크> ① 찾고자 하는 값을 셀 번호가 아닌 텍스트로 직접 입력해도 된다. 단, 텍스트를 입력할 때는 큰따옴표(" ")로 묶어주어야 텍스트로 인식한다.
=VLOOKUP("시금치",Sheet1!A2:D6,4,0)

문 4. <보기>는 모듈화를 중심으로 한 소프트웨어 설계방법에 대한 설명이다. 빈칸의 내용을 올바르게 나열한 것은?

〈보기〉

◦ 결합도(coupling)와 응집도(cohesion)는 모듈의 (㉠)을 판단하는 기준이다.
◦ 결합도란 모듈 (㉡)의 관련성을 의미하며, 응집도란 모듈 (㉢)의 관련성을 의미한다.
◦ 좋은 설계를 위해서는 결합도는 (㉣), 응집도는 (㉤) 방향으로 설계해야 한다.

	㉠	㉡	㉢	㉣	㉤
①	독립성	사이	내부	작게	큰
②	독립성	내부	사이	크게	작은
③	추상성	사이	내부	작게	큰
④	추상성	내부	사이	크게	작은

답 ①

▷ 응집도(cohesion)는 모듈 내부의 기능들이 관련되어 있는 정도, 결합도(coupling)는 모듈과 모듈 간의 상호 의존 정도를 나타낸 것이다. 응집도는 높을수록, 결합도는 낮을수록 좋다.

㉠ 독립성
㉡ 사이
㉢ 내부
㉣ 작게
㉤ 큰

◈ **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

◈ **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

문 5. 다음 중 데이터 값의 대소를 비교하여 정렬하는 문제에 대한 가장 빠른 알고리즘의 시간복잡도는? 단, n은 정렬 대상의 입력 데이터 수이다.

① $O(n)$ ② $O(\log_2 n)$
③ $O(n\log_2 n)$ ④ $O(n^2)$

답 ③

③ 정렬 알고리즘 중 가장 빠른 것은 힙 정렬과 합병 정렬이다.(또는 최악이 아닌 경우의 퀵 정렬)
이러한 알고리즘들의 시간복잡도는 $O(n\log_2 n)$이다.
삽입 정렬의 경우, 이미 정렬되어 있는 상황에서는 시간복잡도가 O(n)이다. 하지만 이는 특수한 상황이므로 별개로 두고 문제를 푼다.

◈ 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)

문 6. 여덟 개의 페이지(0 ~ 7페이지)로 구성된 프로세스에 네 개의 페이지 프레임이 할당되어 있고, 이 프로세스의 페이지 참조 순서는 <보기>와 같다. 이 경우 LRU 페이지 교체 알고리즘을 적용할 때 페이지 적중률(hit ratio)은 얼마인가? 단, <보기>의 숫자는 참조하는 페이지번호를 나타내고, 최초의 페이지 프레임은 모두 비어있다고 가정한다.

〈보기〉
1, 0, 2, 2, 2, 1, 7, 6, 7, 0, 1, 2

① $\frac{5}{12}$ ② $\frac{6}{12}$
③ $\frac{7}{12}$ ④ $\frac{8}{12}$

답 ①

▶ LRU(Least Recently Used)
가장 오랫동안 사용하지 않은 페이지를 교체

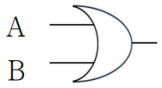
프레임	1	0	2	2	2	1	7	6	7	0	1	2
1	**1**	1	1	1	1	**1**	1	1	1	1	**1**	1
2		**0**	0	0	0	0	0	**6**	6	6	6	**2**
3			**2**	**2**	**2**	2	2	2	2	**0**	0	0
4							**7**	7	**7**	7	7	7
부재	◎	◎	◎				◎	◎		◎		◎

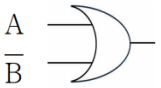
총 12번 중, 7번의 페이지 부재(fault)가 발생하였으며 5번이 페이지 적중(hit)하였다.
따라서 적중률은 **5/12** 이다.

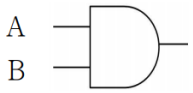
문 7. <보기>의 논리 연산식을 간략화한 논리회로는?

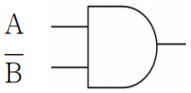
〈보기〉

$(A + B)(A + \overline{B})(\overline{A} + B)$

① 

② 

③ 

④ 

답 ③

(A + B) (A + B` ) ( A` + B)
= (A + B·B` ) ( A` + B) <----- 분배 법칙
= (A + 0) (A` + B)
= A · (A` + B)
= A·A` + AB <----- 분배 법칙
= 0 + AB
= **AB**

③번 회로도가 A와 B의 AND 회로도, A+B 이다.

<오답 체크> ① A + B
② A + B`
④ A · B`

문 8. <보기>의 설명에 해당하는 네트워크 장비는?

〈보기〉

◦ OSI 계층 모델의 네트워크 계층에서 동작하는 장비이다.
◦ 송신측과 수신측 간의 가장 빠르고 신뢰성 있는 경로를 설정·관리하며, 데이터를 전달하는 역할을 한다.
◦ 주로 같은 프로토콜을 사용하는 네트워크간의 최적 경로 설정을 위해 패킷이 지나가야 할 정보를 테이블에 저장하여 지정된 경로를 통해 전송한다.

① 게이트웨이(gateway)
② 브리지(bridge)
③ 리피터(repeater)
④ 라우터(router)

답 ④

④ 네트워크 계층에서 작동하며 경로를 설정하는 장비는 라우터(router)이다.

▷ 1계층 **리피터** (Repeater)
물리계층 상에서 세그먼트를 단순 연결
신호 연장, 증폭 장치
▷ 1계층 **허브** (Hub)
한꺼번에 여러 대의 컴퓨터를 케이블로 연결하는 장치
각 회선을 통합적으로 관리
▷ 2계층 **브리지** (Bridge)
리피터의 기능을 포함하며, 신호 증폭뿐만 아니라 네트워크 분할을 통해 트래픽을 감소, 물리적으로 다른 네트워크를 연결
▷ 2계층 **스위치** (Switch)
브리지의 발전된 형태
훨씬 향상된 네트워크 속도
데이터를 필요로 하는 컴퓨터에만 전송
▷ 3계층 **라우터** (Router)
경로 설정
데이터의 흐름 제어
▷ 4~7계층 **게이트웨이** (Gateway)
프로토콜이 다른 네트워크를 연결, 프로토콜 변환
응용계층을 연결하여 데이터 형식의 변환

문 9. 다음 C 프로그램의 실행 결과로 옳은 것은?

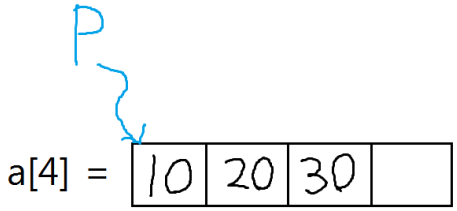
```
void main()
{
    int a[4]={10, 20, 30};
    int *p = a;

    p++;
    *p++ = 100;
    *++p = 200;
    printf("a[0]=%d a[1]=%d a[2]=%d\n",
        a[0], a[1], a[2]);
}
```

① a[0]=10 a[1]=20 a[2]=30
② a[0]=10 a[1]=20 a[2]=200
③ a[0]=10 a[1]=100 a[2]=30
④ a[0]=10 a[1]=100 a[2]=200

답 ③

int a[4]={10, 20, 30};
먼저 4열 짜리 배열 a를 생성한다.
int *p = a;
그리고 그 배열의 주소를 p로 복사한다. 그러면 아래와 같은 자료 메모리 구조가 된다.
p에는 a[0]의 값이 아닌, a[0]의 주소가 들어간다.



p++; 를 통해서 p값을 1 증가시킨다.
p는 주소값이므로 1을 증가시키면 a[1]의 주소값이 된다.

*p++ = 100; 에서 ++이 뒤에 붙어있기 때문에 먼저 *p = 100 의 연산을 먼저 실행한 뒤 p값을 증가시킨다.
현재 p가 가리키는 곳은 a[1]이므로 a[1]에 100을 넣는다.

a[4] = [10 | 100 | 30 |]

그 다음 p값을 1 증가시켜, 이제 p가 가리키는 곳은 a[2]가 된다.

*++p = 200; 에서는 ++이 앞에 붙어있기 때문에 p값을 증가시키는 연산을 먼저 수행한다.
p가 1 증가하여 이제 p가 가리키는 곳은 a[3]이 된다.
그러고 나서 p가 가리키는 a[3]에 200을 집어넣는다.

a[4] = [10 | 100 | 30 | 200]

printf 문을 통해 a[0], a[1], a[2] 값을 출력하면,
각각 **10, 100, 30 이 출력**된다.

문 10. 인터럽트 처리를 위한 <보기>의 작업이 올바로 나열된 것은?

〈보기〉
ㄱ. 인터럽트 서비스 루틴을 수행한다.
ㄴ. 보관한 프로그램 상태를 복구한다.
ㄷ. 현재 수행 중인 명령을 완료하고 상태를 저장한다.
ㄹ. 인터럽트 발생 원인을 찾는다.

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

답 ①

✠ 인터럽트 처리 과정
1. 인터럽트 요청 신호 발생
2. 현재 수행중인 명령을 완료하고 상태를 저장 -- (ㄷ)
3. 어느 장치가 인터럽트를 요청하였는지 확인 --- (ㄹ)
4. 인터럽트 서비스(취급) 루틴을 수행 ---------- (ㄱ)
5. 보존한 프로그램 상태를 복귀 ---------- (ㄴ)

문 11. <표>의 CPM(Critical Path Method) 소작업 리스트에서 작업 C의 가장 빠른 착수일(earliest start time), 가장 늦은 착수일(latest start time), 여유 기간(slack time)을 순서대로 나열한 것은?

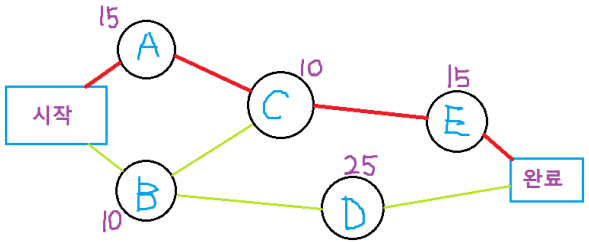
〈표〉 CPM 소작업 리스트

소작업	선행 작업	소요 기간(일)
A	없음	15
B	없음	10
C	A, B	10
D	B	25
E	C	15

① 15일, 15일, 0일
② 10일, 15일, 5일
③ 10일, 25일, 5일
④ 15일, 25일, 0일

답 ①

CPM 경로를 그림으로 그려보면 다음과 같다.



빨간 경로가 임계 경로(Critical Path)이며, 이 작업 전체 소요 기간은 임계 경로 상의 작업들의 기간을 더한 15 + 10 + 15 = 40일이다.
작업 C는 선행 작업인 A가 완료되어야 작업을 시작할 수 있으므로, C의 가장 빠른 착수일은 아무리 **빨라도 15일**이다.
전체 작업 기간이 40일이며, 작업 C 다음에 수행할 작업 E는 15일이 걸린다. 그러므로 작업 C는 늦어도 40 - 15 = 25일 까지는 완료해서 E로 넘겨줘야 하며, 작업 C의 소요 기간은 10일이므로 아무리 **늦어도 15일**에는 작업을 시작해야 한다.
작업 C의 가장 빠른 착수일도 15일, 가장 늦은 착수일도 15일이므로, **여유 기간은 없다.(0일)**

cf) 원래 임계 경로 상의 작업들은 빠른 착수일과 늦은 착수일이 같으며 여유 시간이 없다. 그러므로 여기에서 C의 빠른 착수일만 구한 뒤, 늦은 착수일과 같은 것, 여유 기간이 0인 것을 고르면 빨리 풀 수 있다.

문 12. <보기>는 스택을 이용한 0-주소 명령어 프로그램이다. 이 프로그램이 수행하는 계산으로 옳은 것은?

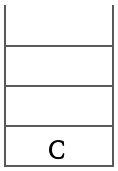
〈보기〉
PUSH C
PUSH A
PUSH B
ADD
MUL
POP Z

① Z = C + A * B ② Z = (A + B) * C
③ Z = B + C * A ④ Z = (C + B) * A

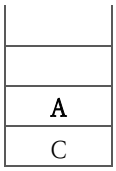
답 ②

스택에서 연산자가 삽입되면, 연산자 아래 있는 두 개를 그 연산자로 계산한다.

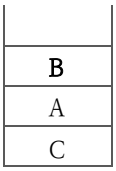
1) PUSH C



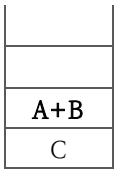
2) PUSH A



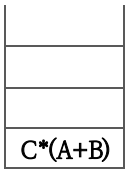
3) PUSH B



4) ADD -> A+B



5) MUL -> C * (A+B)



6) POP Z -> 현재 스택 값을 Z로 출력한다는 의미이다.
Z = C * (A + B)

solve 2) 또한 스택은 후위 표기법 연산과 같다.
스택에 들어온 변수와 연산자를 후위 표기법대로 표시하면,
C A B + * => C (A + B) *
C (A + B) * => C * (A + B)

따라서, **Z = C * (A + B)** 가 된다.

문 13. 트랜잭션의 특성과 이에 대한 설명으로 옳지 않은 것은?

① 원자성(atomicity) : 트랜잭션은 완전히 수행되거나 전혀 수행되지 않아야 한다.
② 일관성(consistency) : 트랜잭션을 완전히 실행하면 데이터베이스를 하나의 일관된 상태에서 다른 일관된 상태로 바꿔야 한다.
③ 고립성(isolation) : 하나의 트랜잭션의 실행은 동시에 실행 중인 다른 트랜잭션의 간섭을 받아서는 안 된다.
④ 종속성(dependency) : 완료한 트랜잭션에 의해 데이터베이스에 가해진 변경은 어떠한 고장에도 손실되지 않아야 한다.

답 ④

④ 트랜잭션의 특성 중 종속성은 없으며, 지속성이 들어가야 한다.

◈ **트랜잭션의 특성**(ACID)
▷ **원자성**(Automicity)
트랜잭션에 포함된 오퍼레이션(작업)들은 모두 수행되거나, 아니면 전혀 수행되지 않아야 한다.
▷ **일관성**(Consistency)
트랜잭션이 성공적으로 완료되면, 일관성있는 상태로 있어야 한다.
▷ **고립성, 독립성**(Isolation)
각 트랜잭션은 독립적으로 수행되고, 실행 중 다른 트랜잭션이 끼어들지 않아야 한다.
▷ **지속성**(Durability) : 성공적으로 수행된 트랜잭션의 결과는 계속해서 유지되어야 한다.

문 14. <보기>의 다양한 진법으로 표현한 숫자들을 큰 숫자부터 나열한 것은?

〈보기〉

ㄱ. $F9_{16}$ ㄴ. 256_{10}

ㄷ. 11111111_{2} ㄹ. 370_{8}

① ㄱ, ㄴ, ㄷ, ㄹ ② ㄴ, ㄷ, ㄱ, ㄹ

③ ㄷ, ㄹ, ㄱ, ㄴ ④ ㄹ, ㄱ, ㄴ, ㄷ

답 ②

2진수, 8진수, 16진수을 비교할 때는 모두 2진법으로 변환한 뒤 비교하는 것이 편하다.

8진수 한 자리는 2진수 3자리로, 16진수 한 자리는 2진수 4자리로 변환이 가능하다.

ㄱ. $F9_{16}$ = 1111 1001

ㄴ. 256은 2^8이므로,

256_{10} = 1 0000 0000

ㄷ. 11111111_{2} = 1111 1111

ㄹ. 370_{8} = 1111 1000

따라서 큰 숫자 순으로 나열하면, **ㄴ - ㄷ - ㄱ - ㄹ** 이다.

문 15. 공개키(public key) 암호화 방식에 대한 설명으로 옳지 <u>않은</u> 것은?

① 공개키와 개인키로 이루어진다.

② 대표적 활용 예로는 전자서명이 있다.

③ 송수신자는 서로 다른 키를 사용한다.

④ 개인키는 메시지를 전송할 때 사용한다.

답 ④

④ 공개키 암호화 방식에서는 메시지를 전송하기 위해 암호화할 때 공개키를 사용하고, 메시지를 전송 받아 복호화할 때 개인키를 사용한다. 이렇게 함으로써 개인키를 가진 수신자만 메시지를 복호화해서 내용을 볼 수 있다.

<오답 체크> ② 메시지를 암호화해서 전송할 때와는 다르게, 전자서명에서는 메시지에 전자 서명을 해서 보낼 때 개인키로 전자서명을 하고, 메시지를 받아 전자 서명을 확인할 때 공개키로 서명을 검증한다.

문 16. 주기억장치와 캐시 기억장치만으로 구성된 시스템에서 <보기>와 같이 기억장치 접근시간이 주어질 때 캐시 적중률(hit ratio)은?

〈보기〉
◦ 평균 기억장치 접근시간 : Ta = 1.9ms
◦ 주기억장치 접근시간 : Tm = 10ms
◦ 캐시 기억장치 접근시간 : Tc = 1ms

① 80% ② 85%
③ 90% ④ 95%

답 ③

평균 기억장치 접근시간은
= 캐시 적중률 × 캐시 접근시간
+ 캐시 적중 실패율 × (캐시 + 주기억장치 접근시간)

캐시 적중률을 h라고 가정하면, 위의 식은
1.9 = h × 1 + (1 - h) × (1 + 10)
-> 1.9 = h + (1 - h) × 11
-> 1.9 = h + 11 - 11h
-> 1.9 = 11 - 10h
-> 10h = 11 - 1.9 = 9.1
∴ h = 0.91
따라서 **캐시 적중률은 91%** 이다.

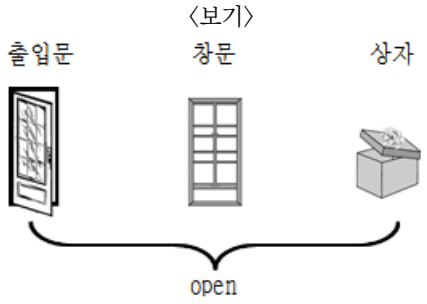
그런데 선지에 답이 없는데, 그 이유는 간혹 어떤 문제들은 캐시 적중 실패했을 때의 접근시간을 (캐시+주기억장치) 접근시간이 아닌, **주기억장치 접근시간만 쓰기** 때문이다.
(사실 캐시 적중 실패했을 때도 먼저 캐시에 접근하여 데이터가 없는 것을 확인하고 나서 주기억장치로 가서 데이터를 찾아오기 때문에, (캐시+주기억장치) 접근시간으로 쓰는 것이 맞다.)

평균 기억장치 접근시간은
= 캐시 적중률 × 캐시 접근시간
+ 캐시 적중 실패율 × 주기억장치 접근시간

1.9 = h × 1 + (1 - h) × 10
-> 1.9 = h + 10 - 10h
-> 1.9 = 10 - 9h
-> 9h = 10 - 1.9 = 8.1
∴ h = 0.9
따라서 **캐시 적중률은 90%** 이다.

문 17. <보기>에서 설명하는 객체지향 개념은?

〈보기〉



◦ 그림에서 'open'이라는 오퍼레이션(operation)은 객체마다 다르게 기능한다.
◦ Java 언어에서 오버로딩(overloading), 오버라이딩(overriding)으로 구현되는 개념이다.

① 캡슐화(encapsulation)
② 인스턴스(instance)
③ 다형성(polymorphism)
④ 상속(inheritance)

답 ③

③ 다형성(polymorphism)
하나의 명령을 자료에 따라 다르게 동작하는 것
- 오버로딩(Overloading): 같은 이름의 함수에 매개변수를 다르게 사용하여 매개 변수에 따라 다른 함수가 실행되는 것.
- 오버라이딩(Overriding): 부모클래스의 함수를 자식클래스에서 같은 이름, 같은 매개변수로 재정의해서 사용하는 것.

<오답 체크> ① 캡슐화(encapsulation)
데이터 구조와 데이터를 다루는 방법을 결합시켜 묶는 것
패스워드에 사용자의 도청(가로채기)

② 인스턴스(instance)
정확히 객체와 인스턴스가 구별되는 것은 아니지만, 객체는 클래스로 정의한 개념이라고 한다면, 인스턴스는 그 클래스의 속성을 받는 객체를 실제로 생성한 것이라 할 수 있다.
예를 들어, 삼성전자에서 갤럭시8을 생산한다면 그 생산품 전체를 갤럭시8이라는 객체라고 볼 수 있지만, 실제 생산되어 소비자가 구입하여 사용하는 각각의 갤럭시8 실물은 미세하게나마 조금씩 다를 것이다. 이렇게 실체화된 객체를 인스턴스라고 한다.

④ 상속성(inheritance)
상위개념의 특징을 하위 개념이 물려받는 것

문 18. <보기>의 연산을 2의 보수를 이용한 연산으로 변환한 것은?

〈보기〉
$6_{10} - 13_{10}$

① $00000110_2 + 11110011_2$
② $00000110_2 - 11110011_2$
③ $11111010_2 + 11110011_2$
④ $11111010_2 - 11110011_2$

답 ①

2의 보수에 의한 연산에서는 양수는 부호화 절대값 표현방법과 같이 표현하고, 음수(뺄셈의 다음 부분)은 2의 보수 표현방법을 이용해 표현한다.

6의 부호화 절대값 -> 0000 0110

13의 부호화 절대값 -> 0000 1101
-13의 1의 보수 방식 -> 1111 0010
-13의 2의 보수 방식 -> 1111 0011

따라서 6 - 13을 2의 보수 표현방법으로 바꾸면,
0000 0110 + 1111 0011 이 된다.

문 19. <보기>는 Windows XP의 실행창(시작 ⇒ 실행)에 입력할 수 있는 명령어들을 나열한 것이다. 명령어별로 수행할 수 있는 기능을 순서대로 나열한 것은?

〈보기〉
dxdiag - msconfig - regedit - mstsc

① 컴퓨터사양 확인 - 시작프로그램 편집 - 레지스트리 편집 - 원격데스크탑 실행
② 원격데스크탑 실행 - 작업관리자 편집 - 서비스 편집 - 시스템 셧다운 설정
③ 컴퓨터사양 확인 - 작업관리자 편집 - 레지스트리 편집 - 원격데스크탑 실행
④ 원격데스크탑 실행 - 시작프로그램 편집 - 서비스 편집 - 시스템 셧다운 설정

답 ①

▷ **dxdiag** 명령어: DirectX 진단 도구를 실행하여, 시스템, 디스플레이, 오디오, 입력 장치 등 컴퓨터 사양을 확인
▷ **msconfig** 명령어: 시스템 구성 도구를 실행하여, 컴퓨터 시작 프로그램을 편집하여 최적화를 수행
▷ **regedit** 명령어: 레지스트리 편집기를 실행하여, 레지스트리를 편집하고 윈도우의 여러 설정값을 변경
▷ **mstsc** 명령어: 원격 데스크톱 연결 도구를 실행하여, 시스템 원격 접속에 대한 설정을 수행

문 20. <보기>는 0 ~ 199번의 200개 트랙으로 이루어진 디스크 시스템에서, 큐에 저장된 일련의 입출력 요청들과 어떤 디스크 스케줄링(disk scheduling) 방식에 의해 처리된 서비스 순서이다. 이 디스크 스케줄링 방식은 무엇인가? 단, <보기>의 숫자는 입출력할 디스크 블록들이 위치한 트랙 번호를 의미하며, 현재 디스크 헤드의 위치는 트랙 50번이라고 가정한다.

〈보기〉
◦ 요청 큐 : 99, 182, 35, 121, 12, 125, 64, 66
◦ 서비스 순서 : 64, 66, 99, 121, 125, 182, 12, 35

① FCFS ② C-SCAN
③ SSTF ④ SCAN

답 ②

서비스 순서를 보면 현재 50번에서 64 - 66 - 99 - 121 - 125 - 182 등 한쪽 방향으로 움직이면서 서비스를 다 처리한 뒤, 제일 앞으로 돌아와 12 - 35를 처리하는 것을 볼 수 있다.
이렇게 한쪽 방향으로 움직이며 가는 길에 있는 요청을 처리하고, 한쪽 끝에 다다르면 처음 시작했던 자리로 다시 되돌아가서 다시 한쪽 방향으로 이동하며 요청을 처리하는 것은 **C-SCAN 스케줄링** (C-SCAN Scheduling) 기법이다.
C-SCAN은 요청이 없어도 끝까지 이동하였다가 다시 처음 부분으로 돌아오는 기법이고, 진행방향에 더 이상 요청이 없으면 그 자리에서 바로 앞으로 되돌아오는 기법은 C-LOOK 스케줄링이다.
문제에서 199번과 0번 트랙으로 이동한 것이 안 보이므로, C-LOOK이라고 생각할 수도 있으나, 문제에 나온 것은 헤더의 이동 경로가 아닌, 서비스 처리 순서이다. 따라서 헤더가 199번, 0번으로 이동했는지 아닌지는 나와있지 않으며, 선택지에 C-LOOK도 없으므로, C-SCAN이 답이 된다.

<오답 체크> ① ▷ **FIFO**(first-come-first-served, 선입 선처리)
요청이 들어온 순서대로 서비스 하는 방식
순서 그대로 99 - 182 - 35 - 121 - 12 - 125 - 64 - 66

③ **SSTF**(Shortest Seek Time First, 최소 탐색 시간 우선)
현재 위치에서 탐색거리가 가장 짧은 요청을 먼저 서비스하는 방식
현재 50에서부터
64 - 66 - 35 - 12 - 99 - 121 - 125 - 182

④ ▷ **SCAN** Scheduling
디스크의 한 끝에서 시작하여 다른 끝으로 왕복 이동하며 가는 길에 있는 모든 요청을 처리
현재 50에서부터(0에서 199방향으로 이동 중이라고 가정)
64 - 66 - 99 - 121 - 125 - 182 - 35 - 12

◈ **디스크 스케줄링 기법**

▷ 선입 선처리(first-come-first-served)
요청이 들어온 순서대로 서비스 하는 방식
빠른 서비스를 제공하지는 못하고 비효율적

▷ 최소 탐색 시간 우선 스케줄링(SSTF Scheduling)
현재 위치에서 탐색거리가 가장 짧은 요청을 먼저 서비스하는 방식
일괄처리 시스템에 유용하며, 기아상태 발생 가능성이 있다.

▷ SCAN 스케줄링(SCAN Scheduling)
디스크의 한 끝에서 시작하여 다른 끝으로 왕복 이동하며 가는 길에 있는 모든 요청을 처리
다른 한쪽 끝에 도달하면 역 방향으로 이동하면서 오는 길에 있는 요청을 처리
SSTF의 응답시간 편차를 줄일 수 있음

▷ C-SCAN 스케줄링 (C-SCAN Scheduling)
항상 바깥쪽에서 안쪽으로 움직이며 가는 길에 있는 요청을 처리
한쪽 끝에 다다르면 처음 시작했던 자리로 다시 되돌아가서 서비스를 시작

▷ LOOK 스케줄링 (LOOK Scheduling)
SCAN 기법처럼 디스크를 왕복 이동하며 요청을 처리하나, 진행방향에 더 이상의 요청이 없으면 끝까지 이동하지 않고 역으로 스캔

▷ C-LOOK 스케줄링 (C-LOOK Schelduling)
항상 바깥쪽에서 안쪽으로 움직이며 가는 길에 있는 요청을 처리하되, 진행방향에 더 이상 요청이 없으면 바로 처음으로 돌아가 다시 서비스를 시작

▷ N-STEP SCAN
SCAN 스케줄링과 유사하나 디스크 헤드가 이동 중에 새로 들어오는 요청은 무시하고 미리 대기 중인 요청만 처리
SSTF나 SCAN에 비해 응답시간의 분산이 적음

▷ SLTF 스케줄링
회전지연시간 최적화를 위한 알고리즘
디스크 헤드가 특정 실린더에 도착하면 그 실린더 내의 여러 트랙에 대한 요청들을 검사 후, 회전지연시간이 가장 짧은 요청부터 서비스