2019년 서울시 9급 컴퓨터일반 풀이

by 호이호이꿀떡

정답 체크

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

1. C 프로그램을 컴파일하면 <보기>와 같은 것들이 실행된다. 이 중 3번째로 실행되는 것은?

〈 보 기 〉
링커(linker), 어셈블러(assembler),
전처리기(preprocessor), 컴파일러(compiler)

① 링커(linker) ② 어셈블러(assembler)
③ 전처리기(preprocessor) ④ 컴파일러(compiler)

③ 전처리기(preprocessor)
본격적인 컴파일을 하기 전에, 컴파일러 과정에서 사용할 여러 입력값들을 우선 처리하여 준비해놓는 프로그램이다.

④ 컴파일러(compiler)
특정 프로그래밍 언어(고급 언어)로 작성된 원시 코드를 한번에 번역하여 목적 프로그램으로 생성해주는 프로그램이다.
다만, 컴파일 과정을 세분화해서 나누면 소스코드를 1차 컴파일 과정을 거치면 어셈블리어로 구성된 프로그램이 생성되고, 이것을 2차 어셈블리어로 해야 기계어 코드로 된 파일이 생성된다.

② 어셈블러(assembler)
어셈블리어로 작성된 코드를 해석해 기계어로 변환해주는 컴퓨터 언어 번역 프로그램이다.

① 링커(linker)
컴파일 과정을 거쳐 생성된 목적 프로그램과 라이브러리, 실행 프로그램 등을 연결·병합하여 실행 가능한 로드 모듈로 만드는 프로그램이다. 링킹 과정까지 끝나야 우리가 실행하는 실행파일이 만들어진다.

답 ②

전처리기 ⇨ 컴파일러 ⇨ 어셈블러 ⇨ 링커

2. 유닉스 파일 시스템에 대한 설명으로 가장 옳지 않은 것은?

① 슈퍼블록은 전체 블록의 수, 블록의 크기, 사용 중인 블록의 수 등 파일 시스템의 정보를 가지고 있다.
② 아이노드는 파일의 종류, 크기, 소유자, 접근 권한 등 각종 속성 정보를 가지고 있다.
③ 파일마다 데이터 블록, 아이노드 외에 직접 블록 포인터와 단일·이중·삼중 간접 블록 포인터로 구성된 인덱스 정보를 가진 인덱스 블록을 별도로 가지고 있다.
④ 디렉터리는 하위 파일들의 이름과 아이노드 포인터(또는 아이노드 번호)를 포함하는 디렉터리 엔트리들로 구성된다.

③ 직접 블록 포인터와 간접 블록 포인터는 아이노드 안에 포함되어 있는 정보이다. 따라서 아이노드 외에 별도로 가지고 있다는 표현이 틀렸다.

<오답 체크> ① 슈퍼 블록(super block)은 자료 구조, 파일 시스템의 크기, 블록의 수, 인덱스, 파일 시스템의 이름, 디스크 이름 등 유닉스 파일 시스템에 관한 정보가 저장된다.

② 아이노드(i-node)는 소유자 그룹, 접근 모드, 파일 형태, 아이노드 숫자 등 해당 파일에 관한 정보가 저장된다.

④ 디렉터리 엔트리(Directory entry)란 디렉터리를 표현하는 데 쓰이는 자료구조를 의미한다.
각각의 파일 시스템에 따라 디렉터리 엔트리를 구성하는 항목이 다른데, 유닉스의 디렉터리 엔트리에는 파일 이름과 아이노드 번호만 저장된다.

답 ③

3. <보기>는 8비트에 부호 있는 2의 보수 표현법으로 작성한 이진수이다. 이에 해당하는 십진 정수는?

〈 보 기 〉
1 0 1 1 1 1 0 0

① -60 ② -68
③ 94 ④ 188

먼저 첫 번째 비트가 1이므로 부호는 마이너스(-)이다.

2의 보수 1011 1100
⇨ 1의 보수 1011 1011
⇨ 각 비트를 반전시켜 부호화 절대치로 바꾸면 0100 0100
1 × 64 + 1 × 4 = 68

따라서, 2위 보수 1011 1100은 **-68** 이다.

답 ②

4. <보기>가 설명하는 것은?

〈 보 기 〉
다음에 실행할 명령어의 주소를 보관하는 레지스터이다. 계수기로 되어 있어 실행할 명령어를 메모리에서 읽으면 명령어의 길이만큼 증가하여 다음 명령어를 가리키며, 분기 명령어는 목적 주소로 갱신할 수 있다.

① 명령어 레지스터 **② 프로그램 카운터**
③ 데이터 레지스터 **④ 주소 레지스터**

다음에 실행할 명령어의 주소를 보관하는 레지스터는 프로그램 카운터(PC, program counter)이다.
주소 레지스터는 처리할 명령어나 데이터가 들어있는 메모리의 주소를 기억하는 레지스터로, 프로그램 카운터도 주소 레지스터의 일종이기는 하나, 보통 별개로 취급한다.

답 ②

5. 운영체제에서 가상 메모리의 페이지 교체 기법에 대한 설명으로 가장 옳지 않은 것은?

① FIFO 기법에서는 아무리 참조가 많이 된 페이지라도 교체될 수 있다.
② LRU 기법을 위해서는 적재된 페이지들의 참조된 시간 또는 순서에 대한 정보가 필요하다.
③ Second-chance 기법에서는 참조 비트가 0인 페이지는 교체되지 않는다.
④ LFU 기법은 많이 참조된 페이지는 앞으로도 참조될 확률이 높을 것이란 판단에 근거한 기법이다.

③ SCR(2차 기회) 기법은 페이지를 교체하기까지 두 번의 기회를 부여하는 방식이다. 먼저 각 페이지마다 참조 비트를 두는데, 최근에 참조되었던 페이지들은 참조 비트가 1이 되고, 오랫동안 참조되지 않은 페이지들은 참조 비트가 0이 된다.
그리고 이 페이지를 교체할 순서가 되면, 참조 비트가 1인 경우는 참조 비트를 0으로 바꾼 뒤 교체는 하지 않는다. 그리고 참조 비트가 0인 상태에서 또 한 번 페이지 교체 순서가 되면 그 때 페이지를 교체한다.

<오답 체크> ① FIFO 기법은 가장 먼저 들어온, 오래된 페이지를 교체하는 기법이기 때문에, 바로 직전에 연속적으로 참조됐던 페이지라도 교체될 수 있다.
② LRU 기법은 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법이기 때문에, 페이지들이 언제 쓰였는지에 대한 시간이나 순서 정보가 필요하다.
④ LFU 기법은 사용 빈도가 적은 페이지를 교체하는 기법인데, 이것은 사요오 빈도가 많았던 페이지는 앞으로도 참조될 확률이 높다는 예측을 따른 것이다.

답 ③

◈ 페이지 교체 알고리즘

▶ **OPT**(OPTimal replacement, 최적 교체)
- 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법
- 각 페이지의 호출 순서와 참조 상황을 미리 예측해야 하므로 실현 가능성이 희박

▶ **FIFO**(Firstt In First Out)
- 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법
- 벨레이디의 모순현상(Belady's Anomaly)이 발생
[일반적으로 페이지 프레임 수가 많으면 페이지 부재의 발생 횟수가 줄어들지만, 페이지 프레임 수를 증가시켰는데도 불구하고 페이지 부재 횟수가 더 많이 일어나는 현상]

▶ **LRU**(Least Recently Used)
- 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법
- 계수기나 스택과 같은 별도의 하드웨어가 필요하며, 시간적인 오버헤드가 발생

▶ **LFU**(Least Frequently Used)
- 사용 빈도가 가장 적은 페이지를 교체하는 기법
- 프로그램 실행 초기에 많이 사용된 페이지가 그 후로 계속 페이지 프레임을 점유

▶ **NUR**(Not Used Recently)
- 최근에 사용하지 않은 페이지를 교체하는 기법
- 최근 사용 여부를 확인하기 위해서 각 페이지마다 참조 비트(Reference Bit)와 변형 비트(Modified Bit, Dirty Bit)를 사용
- 참조하지 않은 페이지를 우선 교체

▶ **MRU**(Most Recently Used)
- 가장 최근에 사용했던 페이지를 교체하는 기법(LRU의 반대)
- 특정한 반복 순환 구조 효율적

▶ **MFU**(Most Frequently Used)
- 사용 빈도가 가장 많은 페이지를 교체하는 기법
- 적게 사용된 페이지가 방금 들어온 것이고 아직 덜 사용되었으므로, 앞으로 사용될 확률이 높다는 계산에 근거

▶ **SCR**(Second Chance Replacement, 2차 기회 교체)
FIFO 기법의 단점을 보완하는 기법으로, 오랫동안 주기억장치에 있었지만 자주 사용되는 페이지의 교체를 방지하기 위한 기법이다.
각 페이지마다 참조 비트를 두어, 교체 대상이 되기 전에 참조 비트를 검사하여 1일 경우, 교체는 하지 않고 참조 비트만 0으로 바꾸어 한번의 기회를 더 부여한다. 그 다음에 참조되지 않아 참조 비트가 0으로 남아있는 상태에서 또 교체해야 할 상황이 오면 교체한다.

6. 네트워킹 장비에 대한 설명으로 가장 옳지 않은 것은?

① 라우터(router)는 데이터 전송을 위한 최선의 경로를 결정한다.
② 허브(hub)는 전달받은 신호를 그와 케이블로 연결된 모든 노드들에 전달한다.
③ 스위치(switch)는 보안(security) 및 트래픽(traffic) 관리 기능도 제공할 수 있다.
④ 브리지(bridge)는 한 네트워크 세그먼트에서 들어온 데이터를 그의 물리적 주소에 관계없이 무조건 다른 세그먼트로 전달한다.

④ 브리지는 주로 OSI 2계층 데이터링크 계층에서 작동하며, 네트워크를 연결·중재하는 역할을 수행한다. 이 때 브리지는 들어오는 프레임을 분석하여 필요한 곳으로만 전달한다. 전체 네트워크 세그먼트를 분할하는 하위 세그먼트로 분할하는 역할을 수행한다고 볼 수 있다.

답 ④

7. 다음의 정렬된 데이터에서 2진탐색을 수행하여 C를 찾으려고 한다. 몇 번의 비교를 거쳐야 C를 찾을 수 있는가? (단, 비교는 '크다', '작다', '같다' 중의 하나로 수행되고, '같다'가 도출될 때까지 반복된다.)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

① 1번 ② 2번 ③ 3번 ④ 4번

현재 전체 데이터의 개수는 15개이다. 이진탐색에서는 정렬된 데이터열에서 검색하고자 하는 범위의 중간값과 대소(大小)를 비교해나가면서 데이터를 찾는 방식이다.

1. 먼저 전체 데이터 배열은 1번부터 15번까지이므로,
   비교할 데이터는 (1 + 15) / 2 = 8번째 데이터인 H이다.
   찾고자 하는 데이터 C와 8번째 데이터인 H를 비교한 뒤, C가 H보다 작으므로 H 이후의 데이터는 검색 범위에서 제외한다.
2. 이번에 검색할 범위는 1번 A부터 7번 G까지이며,
   비교할 데이터는 (1 + 7) / 2 = 4번째 데이터인 D이다.
   C가 D보다 작으므로, D 이후의 데이터들은 검색 범위에서 제외한다.
3. 검색 범위는 1번 A부터 3번 C까지이며,
   비교할 데이터는 (1 + 3) / 2 = 2번째 데이터인 B이다.
   C가 B보다 크므로, B 이전의 데이터들은 검색 범위에서 제외한다.
4. 이로써 남은 검색 범위는 3번 C 하나이며, C와 비교하여 같은 값을 찾는다.
   이렇게 H -> D -> B -> C 총 4번의 비교를 거쳐 C를 찾을 수 있다.

답 ④

8. 인터넷 서비스 관련 용어들에 대한 설명으로 가장 옳지 않은 것은?

① ASP는 동적 맞춤형 웹페이지의 구현을 위해 사용된다.
② URL은 인터넷상에서 문서나 파일의 위치를 나타낸다.
③ HTML은 웹문서의 전달을 위한 통신 규약이다.
④ SSL은 안전한 웹통신을 위한 암호화를 위해 사용된다.

③ HTML(HyperText Markup Language)
웹 페이지를 표시하기 위한 마크업 언어이다. 여기서 마크업 언어(markup language)란 태그 등을 이용해 문서가 화면에 표시되는 형태나 데이터의 구조를 명기하는 언어를 의미한다.
따라서 HTML은 웹 문서의 전달을 위한 통신 규약이 아니고, 데이터를 분석하여 웹 문서를 화면에 표시하기 위한 규약에 해당한다.
가장 옳지 않은 것을 골라야 하므로 ③번이 답이 된다.

<오답 체크> ① ASP(Active Server Page)
마이크로소프트사에서 동적으로 웹 페이지들을 생성하기 위해 개발한 서버 측 스크립트 엔진이다.

② URL(Uniform Resource Locator)
네트워크 상에서 자원이 어디 있는지를 알려주기 위한 규약. 흔히 웹 사이트 주소로 알고 있지만, URL은 웹 사이트 주소뿐만 아니라 컴퓨터 네트워크상의 모든 자원의 위치를 나타낼 수 있다.

③ SSL(Secure Socket Layer) 또는 TLS(Transport Layer Security)
Netscape사에서 웹 서버와 브라우저 사이의 보안을 위해 만든 프로토콜. 전송 계층에서 클라이언트와 서버에 대한 인증 및 데이터 암호화를 수행한다.

답 ③

9. <보기>의 배열 A에 n개의 원소가 있다고 가정하자. 다음 의사코드에 대한 설명으로 가장 옳지 않은 것은?

〈 보 기 〉

```
Function(A[ ], n)
{
    for last ← n downto 2
    // last를 n에서 2까지 1씩 감소
        for i ← 1 to last-1
            if (A[i]>A[i+1]) then A[i]↔A[i+1];
            //A[i]와 A[i+1]를 교환
}
```

① 제일 큰 원소를 끝자리로 옮기는 작업을 반복한다.
② 선택 정렬을 설명하는 의사코드이다.
③ $O(n^2)$의 수행 시간을 가진다.
④ 두 번째 for 루프의 역할은 가장 큰 원소를 맨 오른쪽으로 보내는 것이다.

위의 코드는 last 값을 하나씩 줄여나가면서,
i 를 1번부터 last-1까지 늘려가며 i 번째 원소와 i+1번째 원소를 비교하여 큰 원소를 뒤로 보내는 코드이다.

간략하게 확인을 해보면, 현재 last는 6이라고 가정하자.
i 는 1부터 last-1 = 5까지 하나씩 증가한다.
i 가 1일 때, 1번째 원소와 2번째 원소를 비교하여 큰 것을 2번으로 보내고,
그 다음 i 가 2일 때, 2번째 원소와 3번째 원소를 비교하여 큰 것을 3번으로 보내고,
i 가 3, 4, 5일 때도 계속 반복한다.
이렇게 한 번의 루틴을 끝내면 가장 큰 원소는 6번으로 보내지게 된다.
이렇게 인접한 두 원소를 비교하여 큰 원소를 하나씩 뒤로 보내는 정렬 방식은 **버블 정렬**(bubble sort) 방식이다.

② **선택 정렬**(selection sort)은 아직 정렬되지 않은 원소들 중에서 가장 큰 원소를 찾아 제일 뒤의 원소와 자리를 바꾸면서, 뒤에서부터 큰 원소를 하나씩 채워나가는 정렬 방법이다.
경우에 따라 앞에서부터 작은 원소를 채워나가는 선택 정렬도 있다.

<오답 체크> ③ 버블 정렬의 시간복잡도는 최선일 때 O(n), 평균 또는 최악일 때 $O(n^2)$이다. 특별한 설명이 없으면 일반적으로 평균을 의미하므로 $O(n^2)$이라 표현한 것이다.

④ 첫 번째 for문은 비교해나갈 루틴의 범위를 하나씩 줄여나가는 역할을 하고, 두 번째 for문은 인접한 두 원소를 비교해나가며 가장 큰 원소를 오른쪽으로 보내는 역할을 한다.

답 ②

10. <보기>의 Java 프로그램의 실행 결과는?

〈 보 기 〉

```
class A {
    public void f( ) { System.out.print("1"); }
    public static void g() { System.out.print("2"); }
}
class B extends A {
    public void f( ) { System.out.print("3"); }
}
class C extends B {
    public static void g( ) { System.out.print("4"); }
}
public class D {
    public static void main(String args[ ]) {
        A obj = new C( );
        obj.f( );
        obj.g( );
    }
}
```

① 3 2 ② 3 4 ③ 1 2 ④ 1 4

A 클래스는 최상위 클래스,
B 클래스는 A 클래스를 상속받는 하위 클래스
C 클래스는 B 클래스를 상속받는 하위 클래스이다.

A obj = new C( ); 는 A 클래스를 참조하는 C 클래스 타입의 객체 obj를 생성한다.

obj.f( ); 는 C 클래스에는 f( ) 메소드가 없으므로, 차상위 클래스인 B 클래스를 찾아간다. 그래서 B 클래스의 f( ) 메소드가 실행되어 **3이 출력**된다.

obj.g( );는 A 클래스와 C 클래스에 정의되어 있으므로, 상속을 따른다면 C 클래스의 g( ) 메소드가 실행될 것이다.
하지만, g( ) 메소드는 static 메소드로 설정되어 있다.
static 메소드는 상속이 불가능하며, 따라서 C 클래스에서 재정의한 g( ) 메소드는 무시된다.(hiding 처리)
따라서 A 클래스의 g( ) 메소드가 실행되어 **2가 출력**된다.

답 ①

11. 어떤 시스템은 7비트의 데이터에 홀수 패리티 비트를 최상위 비트에 추가하여 8비트로 표현하여 저장한다. 다음과 같은 데이터를 저장 장치에서 읽어 왔을 때 오류가 발생한 경우는?

① 011010111 ② 101101111
③ 011001110 ④ 101001101

홀수 패리티 방식은 비트열의 1의 개수를 홀수 개수로 맞추는 방식이다.

① 011010111 ⇨ 1이 6개
② 101101111 ⇨ 1이 7개
③ 011001110 ⇨ 1이 5개
④ 101001101 ⇨ 1이 5개

따라서 1의 개수가 짝수인 ①번이 오류가 발생한다.

답 ①

그런데 이 문제에서 7비트의 데이터에 홀수 패리티 비트를 최상위 비트에 추가하여 **8비트로 표현해 저장**한다고 하였다.
하지만, 문제의 보기들은 모두 9비트이다. 9비트이므로 8비트씩 불러와 패리티 검사를 수행하고, 뒤에 남은 1비트는 생략된 뒷부분으로 이어진다.
그렇게 앞의 8비트씩 잘라서 패리티 체크를 해보면,(빨간색으로 칠한 비트는 패리티 비트)

① **0110 1011** ⇨ 1의 개수 5개
② **1011 0111** ⇨ 1의 개수 6개
③ **0110 0111** ⇨ 1의 개수 5개
④ **1010 0110** ⇨ 1의 개수 4개

따라서 오류가 발생하는 것은 ②번과 ④번 두 개이다.
그런데 이 문제가 왜 답이 ①번인지, 이의제기가 받아들여지지 않았는지는 모르겠다. 전공서적과 구글링을 통해서는 원하는 답변을 찾지 못하였다. 혹시 아는 분들은 꼭 답글을 남겨주기 바란다.

12. 고객, 제품, 주문, 배송업체 테이블을 가진 판매 데이터베이스를 SQL을 이용해 구축하고자 한다. 각 테이블이 <보기>와 같은 속성을 가진다고 가정할 때, 다음 중 가장 옳지 않은 SQL문은? (단, 밑줄은 기본키를 의미한다.)

· 고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
· 제품(제품번호, 제품명, 재고량, 단가, 제조업체)
· 주문(주문번호, 주문제품, 주문고객, 수량, 배송지, 주문일자)
· 배송업체(업체번호, 업체명, 주소, 전화번호)

① 고객 테이블에 가입 날짜를 추가한다. →
"ALTER TABLE 고객 ADD 가입 날짜 DATE;"
② 주문 테이블에서 배송지를 삭제한다. →
"ALTER TABLE 주문 DROP COLUMN 배송지;"
③ 고객 테이블에 18세 이상의 고객만 가입 가능하다는 무결성 제약 조건을 추가한다. → "ALTER TABLE 고객 ADD CONSTRAINT CHK_AGE CHECK(나이>=18);"
④ 배송업체 테이블을 삭제한다. →
"ALTER TABLE 배송업체 DROP;"

④ 테이블 자체를 삭제할 때는 ALTER 문이 아닌 DROP 문을 사용한다. 옳게 바꾸면,
"DROP TABLE 배송업체;" 가 된다.

<오답 체크> 속성(컬럼)을 추가하건 제거하건, 제약조건을 명시하건 제외하건, 이러한 것들은 테이블의 구조를 변경하는 것이므로 기본적으로 ALTER 문을 사용한다.

① 테이블에 속성을 추가할 땐 ALTER ~ ADD 문을 사용한다.
② 테이블에서 속성을 제거할 땐 ALTER ~ DROP COLUMN 문을 사용한다.
DROP은 테이블 자체를 삭제할 때도 사용하지만, 속성을 제거할 때도 사용한다. 대신 DROP COLUMN이라고 속성을 삭제한다는 것을 분명히 명시해야 한다.
① 번의 ADD문과 비교해서 꼭 차이점을 확인하고 넘어간다.
③ 속성의 제약조건을 추가하는 것은 ALTER ~ CONSTRAINT 문을 사용한다.
해당 SQL문은 CHK_AGE라는 이름의 제약조건을 추가하는 것이고, 이 제약조건은 나이가 18 이상인지 체크(CHECK)한다는 의미이다.
제약조건의 이름이 필요한 이유는 나중에 삭제할 때 필요하기 때문이다. "DROP CONSTRAINT CHK_AGE;"

답 ④

13. <보기>의 UML 다이어그램 중 시스템의 구조(structure)보다는 주로 동작(behavior)을 묘사하는 다이어그램들만 고른 것은?

〈 보 기 〉
ㄱ. 클래스 다이어그램(class diagram)
ㄴ. 상태 다이어그램(state diagram)
ㄷ. 시퀀스 다이어그램(sequence diagram)
ㄹ. 패키지 다이어그램(package diagram)
ㅁ. 배치 다이어그램(deployment diagram)

① ㄱ, ㄹ ② ㄴ, ㄷ ③ ㄴ, ㅁ ④ ㄷ, ㄹ

ㄴ. 상태 다이어그램(State Diagram)
사건이나 시간에 따라 시스템 객체의 상태 변화(동적인 상황 변화)를 나타낸 그림
ㄷ. 시퀀스(순서) 다이어그램(Sequence Diagram)
사용 사례가 어떻게 수행되는지, 어떤 메시지가 언제 보내지는지 메시지 교환을 시간의 흐름에 따라 나타낸 그림
<오답 체크> ㄱ. 클래스 다이어그램(Class Diagram)
객체, 클래스, 속성, 오퍼레이션 및 연관관계를 이용하여 시스템의 논리적 구조를 표현한 그림(정적인 구조)
ㄹ. 패키지 다이어그램(Package Diagram)
클래스와 같은 여러 모델 요소들을 그룹화하여 패키지를 구성하고 패키지 사이의 관계를 표현하는 그림
ㅁ. 배치 다이어그램(Deployment Diagram)
시스템을 구성하는 하드웨어 간의 연결관계를 표현하고, 하드웨어 자원에 대한 소프트웨어 컴포넌트의 배치 상태를 표현한 그림

답 ②

14. <보기 1>의 테이블 R에 대해 <보기 2>의 SQL을 수행한 결과로 옳은 것은?

〈 보 기 1 〉

A	B
3	1
2	4
3	2
2	5
3	3
1	5

〈 보 기 2 〉
SELECT SUM(B) FROM R GROUP BY A
HAVING COUNT(B) = 2;

① 2 ② 5 ③ 6 ④ 9

GROUP BY ~ HAVING 구문은 다른 SQL문에 비해 복잡하기 때문에, 익숙해지기 전까지는 하나씩 차근차근 분석해나가는 습관이 중요하다.

먼저 GROUPB BY A 이므로, A 값이 같은 것끼리 묶는다. 묶으면 A가 3인 그룹, A가 2인 그룹, A가 1인 그룹으로 나뉜다.

A	B
3	1
3	2
3	3

A	B
2	4
2	5

A	B
1	5

그리고 HAVING COUNT(B) = 2를 통해서, COUNT(B) = 2, 즉, B 값을 가진 튜플이 2개인 그룹을 추려낸다.

A	B
2	4
2	5

마지막으로 SELECT SUM(B)를 통해, 추려낸 그룹의 B 값들을 모두 더한다.
따라서 연산 결과 **4 + 5 = 9** 가 출력된다.

답 ④

15. <보기>는 데이터가 정렬되는 단계를 일부 보여준 것이다. 어떤 정렬 알고리즘을 사용하면 이와 같은 데이터의 자리 교환이 일어나겠는가? (단, 제일 위의 행이 주어진 데이터이고, 아래로 내려갈수록 정렬이 진행되는 것이다.)

〈 보 기 〉

초기 데이터	8 9 4 3 7 1 5 2
	8 9 3 4 1 7 2 5
	3 4 8 9 1 2 5 7
정렬 데이터	1 2 3 4 5 7 8 9

① 삽입 정렬 **② 선택 정렬**
③ 합병 정렬 **④ 퀵 정렬**

▶ 합병 정렬(merge sort)은 분할 정복 알고리즘(divide and conquer algorithm)을 사용하는 정렬 기법 중 하나로, 전체 리스트를 잘게 나누어 여러 개의 부분으로 쪼갠 뒤, 각 부분들을 합쳐가며 정렬하는 기법이다.

▷ 분할(divide) 과정
초기 상태 [8 9 4 3 7 1 5 2]
1차 분할 [8 9 4 3] [7 1 5 2]
2차 분할 [8 9] [4 3] [7 1] [5 2]
분할 과정은 데이터의 순서가 바뀌지 않기 때문에 보기의 1행과 같은 상태이다.

▷ 병합(정복, conqure) 과정
초기 상태 (보기 1행) [8 9] [4 3] [7 1] [5 2]
1차 정렬: 각 부분들을 정렬한다 (보기 2행)
[8 9] [3 4] [1 7] [2 5]
2차 정렬: 각 부분들을 1차로 통합한 뒤 정렬한다 (보기 3행)
(통합) [8 9 3 4] [1 7 2 5]
⇨ (정렬) [3 4 8 9] [1 2 5 7]
3차 정렬: 한 번 더 통합·정렬한다 (보기 4행)
(통합) [8 9 3 4 1 7 2 5]
⇨ (정렬) [1 2 3 4 5 7 8 9]

답 ③

① **삽입 정렬**(insertion sort)

배열을 정렬된 앞 부분과 정렬되지 않은 뒷 부분으로 나눠, 이미 정렬된 앞 부분 배열과 비교하여 뒷 부분의 요소를 하나씩 적당한 위치를 찾아 삽입하는 방법

초기 데이터 [8 9 4 3 7 1 5 2]
1회전 [8 9 4 3 7 1 5 2]
2회전 [4 8 9 3 7 1 5 2]
3회전 [3 4 8 9 7 1 5 2]
4회전 [3 4 7 8 9 1 5 2]
5회전 [1 3 4 7 8 9 5 2]
6회전 [1 3 4 5 7 8 9 2]
7회전 [1 2 3 4 5 7 8 9]

② **선택 정렬**(selection sort)

아직 정렬되지 않은 원소들 중에서 가장 큰 원소를 찾아 제일 뒤의 원소와 자리를 바꾸면서, 뒤에서부터 큰 원소를 하나씩 채워나가는 정렬 방법

초기 데이터 [8 9 4 3 7 1 5 2]
1회전 [8 2 4 3 7 1 5 9]
2회전 [5 2 4 3 7 1 8 9]
3회전 [5 2 4 3 1 7 8 9]
4회전 [1 2 4 3 5 7 8 9]
5회전 [1 2 3 4 5 7 8 9]
6회전 [1 2 3 4 5 7 8 9]
7회전 [1 2 3 4 5 7 8 9]

④ **퀵 정렬**(quick sort)

분할 정복 알고리즘을 사용하는 정렬 기법 중 하나.

퀵 정렬의 작동방식은 설명하기 복잡하므로 이론서를 찾아보는 걸 권한다.(노란색 원소는 pivot 값)

▷ 1차 분할
초기 데이터 [8 9 4 3 7 1 5 2]
1과 8 교환 [1 9 4 3 7 8 5 2]
pivot(2)와 9 교환 [1 2 4 3 7 8 5 9]
[1] [2] [4 3 7 8 5 9]

▷ 2차 분할: 9가 가장 크므로 교환 과정 없이 분할만 수행
[1] [2] [4 3 7 8 5] [9]

▷ 3차 분할
초기 데이터 [1] [2] [4 3 7 8 5] [9]
pivot(5)와 7 교환 [1] [2] [4 3 5 8 7] [9]
[1] [2] [4 3] [5] [8 7] [9]

▷ 최종 정렬 [1] [2] [3 4] [5] [7 8] [9]
[1 2 3 4 5 7 8 9]

16. <보기>의 각 설명과 일치하는 데이터 구조로 바르게 짝지어진 것은?

〈 보 기 〉
㈎ 먼저 추가된 항목이 먼저 제거된다.
㈏ 먼저 추가된 항목이 나중에 제거된다.
㈐ 항목이 추가된 순서에 상관없이 제거된다.

	㈎	㈏	㈐
①	큐	연결 리스트	스택
②	스택	연결 리스트	큐
③	스택	큐	연결 리스트
④	큐	스택	연결 리스트

㈎ 큐(queue)
리스트의 한 방향에서 삽입되어, 반대 방향으로 삭제되는 구조 (rear로 삽입되어 front로 삭제된다) 따라서 추가된 순서대로 제거되는 FIFO 구조이다.

㈏ 스택(stack)
삽입/삭제가 한 방향에서 이루어지는 데이터 구조(LIFO 구조)

㈐ 연결 리스트
노드의 포인터 부분으로 서로 연결시킨 리스트 구조. 노드의 연결값만 바꾸면 되므로, 순서에 상관없이 추가·삭제된다.

답 ④

17. 전화번호의 마지막 네 자리를 3으로 나눈 나머지를 해싱(hashing)하여 데이터베이스에 저장하고자 한다. 나머지 셋과 다른 저장 장소에 저장되는 것은?

① 010-4021-6718 ② 010-9615-4815
③ 010-7290-6027 ④ 010-2851-5232

① 6718 mod 3 = 1
② 4815 mod 3 = 0
③ 6027 mod 3 = 0
④ 5232 mod 3 = 0

따라서 나머지 값이 다른 ①번만 1번 저장소에 들어가고, 나머지는 0번 저장소에 들어간다.

3의 배수는 각 자릿수의 합이 3의 배수가 된다는 점을 이용하면 직접 나눗셈을 할 필요없이 빨리 구할 수 있다.

답 ①

18. 다음 메모리 영역 중 전역 변수가 저장되는 영역은?

① 데이터(Data) ② 스택(Stack)
③ 텍스트(Text) ④ 힙(Heap)

① **Data**(데이터) 영역
전역 변수와 정적(static) 변수가 할당되는 영역
프로그램의 시작과 동시에 할당되고, 프로그램이 종료되어야 메모리에서 소멸됨
쓰기 가능, 크기 고정
(좀 더 세분화하면 데이터 영역에는 초기화된 전역 변수와 정적 변수가 저장되고, 초기화되지 않은 전역 변수와 정적 변수는 BSS (Block Stated Symbol) 영역에 저장된다.)

<오답 체크> ② **Stack**(스택) 영역
함수 호출 시 생성되는 지역 변수와 매개 변수가 저장되는 영역
함수 호출이 완료되면 사라짐
쓰기 가능, 크기 가변

③ **Text**(텍스트) 영역 = Code(코드) 영역
소스 코드 자체를 기계어로 변환하여 저장되는 영역으로, 변수가 아닌 순수 코드만 있는 공간이다.
쓰기 금지, 크기 고정

④ **Heap**(힙) 영역
필요에 의해 동적으로 메모리를 할당 할 때 사용
쓰기 가능, 크기 가변

답 ①

19. UML(Unified Modeling Language)에 대한 설명으로 가장 옳지 않은 것은?

① UML은 방법론으로, 단계별로 어떻게 작업해야 하는지 자세하게 나타낸다.
② UML은 소프트웨어의 구성요소와 그것들의 관계 및 상호작용을 시각화한 것이다.
③ UML은 객체지향 소프트웨어를 모델링하는 표준 그래픽 언어로, 심벌과 그림을 사용해 객체지향 개념을 나타낼 수 있다.
④ UML은 소프트웨어 개발의 중요한 작업인 분석, 설계, 구현의 정확하고 완벽한 모델을 제공한다.

▶ UML(Unified Modeling Language, 통합 모델링 언어)
소프트웨어를 설계에서 객체간의 관계와 구조를 시각적으로 표현하는 모델링 언어. 건축의 설계도의 개념과 유사하다.

① UML도 단순한 모델링 언어일 뿐, 소프트웨어 방법론은 아니다.

답 ①

20. <보기>의 C 프로그램을 실행했을 때, 화면에 출력되는 값은? (단, 프로그램의 첫 번째 열의 숫자는 행 번호이고 프로그램의 일부는 아님.)

〈 보 기 〉

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define N 3
4  int main(void) {
5      int (*in)[N], *out, sum=0;
6
7      in=(int (*)[N])malloc(N * N * sizeof(int));
8     out=(int *)in;
9
10     for (int i=0; i < N * N; i++)
11         out[i]=i;
12
13     for (int i=0; i < N; i++)
14         sum += in[i][i];
15
16     printf("%d", sum);
17     return 0;
18 }
```

① 0 ② 3 ③ 6 ④ 12

line 5. N이 3이므로, int (*in) [N]은 int (*in) [3]으로 바뀐다.
이것은 1행이 3칸으로 이루어진 int형 2차원 배열을 가리키는 포인터 변수 in을 선언한다는 의미이다.

line 7. malloc(N * N * sizeof(int)); 는 메모리를 동적 할당하는 함수로, 3 * 3 * int 크기, 즉 int 크기의 9배에 해당하는 메모리를 할당한다는 의미이다.
그리고 이렇게 할당한 메모리를 2차원 배열 포인터인 in 변수에 연결한다.
연산 결과 in 배열 포인터에는 int 크기의 9배에 해당하는 메모리가 할당되었고, in 배열 포인터는 1행이 int 변수 3칸으로 구성된 2차원 배열을 가리키므로 총 3행이 만들어진다.
즉, in 포인터는 3×3 배열을 가리키는 포인터가 되었다.

line 8. in의 포인터 값을 out 포인터 변수에 넘겨준다.
단, 이 때 out 포인터 변수는 2차원이, 단일 포인터로 형변환되어 넘겨 받는다.
이것을 그림으로 나타내면 in 배열 포인터가 가리키는 형태는 아래와 같은 모양이고,
(여기에서 x00, x04, x08 등은 메모리 주소를 나타낸다)

x00 a[0][0]	x04 a[0][1]	x08 a[0][2]
x12 a[1][0]	x16 a[1][1]	x20 a[1][2]
x24 a[2][0]	x28 a[2][1]	x32 a[2][2]

out 포인터가 가리키는 형태는 아래과 같은 모양이다.

x00 a[0]	x04 a[1]	x08 a[2]	x12 a[3]	x16 a[4]	x20 a[5]	x24 a[6]	x28 a[7]	x32 a[8]

line 10. for문을 통해 위의 배열에는 0부터 8까지 순차적으로 값이 들어간다.

line 13. for문을 통해 in[0][0], in[1][1], in[2][2]의 세 값이 sum에 더해지며, 이것은 out[0], out[4], out[8]의 값과 같으므로
sum은 **0 + 4 + 8 = 12** 가 된다.

답 ④