기출문제

2021년 3회 정보처리기사 실기

정보처리기사 실기 시험은 한국산업인력공단에서 문제를 공개하지 않아 문제 복원에 많은 어려움이 있습니다. 다음에 제시된 문제는 시험을 치룬 학생들의 기억을 토대로 복원한 것이므로, 일부 내용이나 문제별 배점이 실제 시험과 다를 수 있음을 알립니다.

***** 수험자 유의사항 *****

1. 시험 문제지를 받는 즉시 응시하고자 하는 종목의 문제지가 맞는지를 확인하여야 합니다.
2. 시험 문제지 총면수·문제번호 순서·인쇄상태 등을 확인하고, 수험번호 및 성명을 답안지에 기재하여야 합니다.
3. 문제 및 답안(지), 채점기준은 일절 공개하지 않으며 자신이 작성한 답안, 문제 내용 등을 수험표 등에 이기(옮겨 적는 행위) 등은 관련 법 등에 의거 불이익 조치 될 수 있으니 유의하시기 바랍니다.
4. 수험자 인적사항 및 답안작성(계산식 포함)은 흑색 기구만 사용하되, 동일한 한 가지 색의 필기구만 사용하여야 하며 흑색을 제외한 유색 필기구 또는 연필류를 사용하거나 2가지 이상의 색을 혼합 사용하였을 경우 그 문항은 0점 처리됩니다.
5. 답란(답안 기재란)에는 문제와 관련 없는 불필요한 낙서나 특이한 기록사항 등을 기재하여서는 안되며 부정의 목적으로 특이한 표식을 하였다고 판단될 경우에는 모든 문항이 0점 처리됩니다.
6. 답안을 정정할 때에는 반드시 정정부분을 두 줄(=)로 그어 표시하여야 하며, 두 줄로 긋지 않은 답안은 정정하지 않은 것으로 간주합니다.
7. 답안의 한글 또는 영문의 오탈자는 오답으로 처리됩니다. 단, 답안에서 영문의 대·소문자 구분, 띄어쓰기는 여부에 관계 없이 채점합니다.
8. 계산 또는 디버깅 등 계산 연습이 필요한 경우는 <문 제> 아래의 연습란을 사용하시기 바라며, 연습란은 채점대상이 아닙니다.
9. 문제에서 요구한 가지 수(항수) 이상을 답란에 표기한 경우에는 답안기재 순으로 요구한 가지 수(항수)만 채점하고 한 항에 여러 가지를 기재하더라도 한 가지로 보며 그 중 정답과 오답이 함께 기재란에 있을 경우 오답으로 처리됩니다.
10. 한 문제에서 소문제로 파생되는 문제나, 가지수를 요구하는 문제는 대부분의 경우 부분채점을 적용합니다. 그러나 소문제로 파생되는 문제 내에서의 부분 배점은 적용하지 않습니다.
11. 답안은 문제의 마지막에 있는 답란에 작성하여야 합니다.
12. 부정 또는 불공정한 방법(시험문제 내용과 관련된 메모지사용 등)으로 시험을 치른 자는 부정행위자로 처리되어 당해 시험을 중지 또는 무효로 하고, 2년간 국가기술자격검정의 응시자격이 정지됩니다.
13. 시험위원이 시험 중 신분확인을 위하여 신분증과 수험표를 요구할 경우 반드시 제시하여야 합니다.
14. 시험 중에는 통신기기 및 전자기기(휴대용 전화기 등)를 지참하거나 사용할 수 없습니다.
15. 국가기술자격 시험문제는 일부 또는 전부가 저작권법상 보호되는 저작물이고, 저작권자는 한국산업인력공단입니다. 문제의 일부 또는 전부를 무단 복제, 배포, 출판, 전자출판 하는 등 저작권을 침해하는 일체의 행위를 금합니다.

※ 수험자 유의사항 미준수로 인한 채점상의 불이익은 수험자 본인에게 전적으로 책임이 있음

문제 1 다음 Java로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
class Connection {
        private static Connection _inst = null;
        private int count = 0;
        static public Connection get() {
                if(_inst == null) {
                        _inst = new Connection();
                        return _inst;
                }
                return _inst;
        }
        public void count() { count++; }
        public int getCount() { return count; }
}

public class Test {
        public static void main(String[] args) {
                Connection conn1 = Connection.get();
                conn1.count();
                Connection conn2 = Connection.get();
                conn2.count();
                Connection conn3 = Connection.get();
                conn3.count();
                System.out.print(conn1.getCount());
        }
}
```

답 :

문제 2 보안 위협에 관한 다음 설명에서 괄호에 공통으로 들어갈 알맞은 답을 쓰시오. (5점)

( ) 스푸핑은 로컬 네트워크(LAN)에서 사용하는 ( ) 프로토콜의 취약점을 이용한 공격 기법으로, 자신의 물리적 주소(MAC)를 변조하여 다른 PC에게 도달해야 하는 데이터 패킷을 가로채거나 방해한다.

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 3 데이터를 제어하는 DCL의 하나인 GRANT의 기능에 대해 간략히 서술하시오. (5점)

답 :

문제 4 AAA 서버에 관한 다음 설명에서 각 번호(①~③)에 들어갈 알맞는 용어를 <보기>에서 찾아 쓰시오. (5점)

AAA 서버는 사용자의 컴퓨터 자원 접근 처리와 서비스 제공에 있어서의 다음 3가지 기능을 제공하는 서버이다. ① - 접근하는 사용자의 신원을 검증하는 기능 ② - 신원이 검증된 사용자에게 특정된 권한과 서비스를 허용하는 기능 ③ - 사용자가 어떤 종류의 서비스를 이용했고, 얼마만큼의 자원을 사용했는지 기록 및 보관하는 기능

<보기>

Application	Authentication	Avalanche	Authorization
Accounting	Ascii		

답

- ①
- ②
- ③

문제 5 디자인 패턴에 관한 다음 설명에서 괄호에 들어갈 알맞은 답을 <보기>에서 찾아 쓰시오. (5점)

( ) 패턴은 객체 생성을 서브 클래스에서 처리하도록 분리하여 캡슐화한 패턴으로, 상위 클래스에서 인터페이스만 정의하고 실제 생성은 서브 클래스가 담당한다. 다른 이름으로 가상 생성자(Virtual Constructor) 패턴이라고도 불린다.

<보기>

Singleton	Abstract Factory	Factory Method	Prototype
Facade	Composite	Template Method	Builder

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 6 결합도(Coupling)의 종류 중 단순 처리 대상인 데이터만 전달되는 것이 아니라 어떻게 처리해야 하는지를 결정하는 제어 요소가 전달되는 경우의 결합도를 영문으로 쓰시오. (5점)

답 :

문제 7 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
#include <stdio.h>
struct jsu {
    char nae[12];
    int os, db, hab, hhab;
};

int main() {
    struct jsu st[3] = { {"데이터1", 95, 88}, {"데이터2", 84, 91}, {"데이터3", 86, 75} };
    struct jsu* p;
    p = &st[0];
    (p + 1)->hab = (p + 1)->os + (p + 2)->db;
    (p + 1)->hhab = (p + 1)->hab + p->os + p->db;
    printf("%d", (p + 1)->hab + (p + 1)->hhab);
}
```

답 :

문제 8 애플리케이션 테스트에 관한 다음 설명에서 괄호(①, ②)에 들어갈 알맞은 답을 쓰시오. (5점)

- (①)는 소프트웨어의 하위 모듈에서 상위 모듈 방향으로 통합하면서 테스트하는 기법이다.
- 하나의 주요 제어 모듈과 관련된 종속 모듈의 그룹인 클러스터(Cluster)가 필요하다.
- 데이터의 입·출력을 확인하기 위해 더미 모듈인 (②)를 생성한다.

답

- ①
- ②

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 9 다음 Python으로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
x, y = 100, 200
print(x==y)
```

답 :

문제 10 <A> 테이블과 <B> 테이블을 참고하여 <SQL문>의 실행 결과를 쓰시오. (5점)

<A>

NAME
Smith
Allen
Scott

<B>

RULE
S%
%T%

<SQL문>

```
SELECT COUNT(*) CNT FROM A CROSS JOIN B WHERE A.NAME LIKE B.RULE;
```

답 :

문제 11 다음 설명에서 괄호에 공통으로 들어갈 알맞은 답을 쓰시오. (5점)

파일의 구조는 파일을 구성하는 레코드들이 보조기억장치에 편성되는 방식을 의미하는 것으로, 크게 순차, ( ), 해싱으로 구분한다. ( ) 파일 구조는 <값, 주소> 쌍으로 구성되는 데이터 구조를 활용하여 데이터에 접근하는 방식으로, 자기 디스크에서 주로 활용된다.

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 12 **다음 테스트 케이스를 참조하여 괄호에 들어갈 테스트 케이스의 구성 요소를 <보기>에서 찾아 쓰시오. (5점)**

식별자_ID	테스트 항목	(①)	(②)	(③)
LS_W10_35	로그인 기능	사용자 초기 화면	아이디(test_a01) 비밀번호(203a!d5%ffa1)	로그인 성공
LS_W10_36	로그인 기능	사용자 초기 화면	아이디(test_a01) 비밀번호(1234)	로그인 실패(1) - 비밀번호 비일치
LS_W10_37	로그인 기능	사용자 초기 화면	아이디("") 비밀번호("")	로그인 실패(2) - 미입력

<보기>

요구 절차	의존성 여부	테스트 데이터	테스트 조건
하드웨어 환경	예상 결과	소프트웨어 환경	성공/실패 기준

답

- ①
- ②
- ③

문제 13 **UML(Unified Modeling Language)에 관한 다음 설명에서 괄호에 공통으로 들어갈 알맞은 답을 쓰시오. (5점)**

( ) 다이어그램은 UML 다이어그램 중 객체(Object)들을 ( )로 추상화하여 표현하는 다이어그램으로 대표적인 구조적 다이어그램이다. ( )는 각각의 객체들이 갖는 속성과 메소드를 표현한 것으로 3개의 구획으로 나눠 이름, 속성, 메소드를 표기한다.

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 14 OSI 7 Layer에 대한 다음 설명에서 각 번호(①~③)에 들어갈 알맞은 계층(Layer)을 쓰시오. (5점)

OSI 7 Layer는 다른 시스템 간의 원활한 통신을 위해 ISO(국제표준화기구)에서 제안한 통신 규약(Protocol)이다. ① – 물리적으로 연결된 두 개의 인접한 개방 시스템들 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 연결 설정, 데이터 전송, 오류 제어 등의 기능을 수행한다. ② – 개방 시스템들 간의 네트워크 연결을 관리하며, 경로 제어, 패킷 교환, 트래픽 제어 등의 기능을 수행한다. ③ – 서로 다른 데이터 표현 형태를 갖는 시스템 간의 상호 접속을 위해 필요한 계층으로, 코드 변환, 데이터 암호화, 데이터 압축, 구문 검색 등의 기능을 수행한다.

답

- ①
- ②
- ③

문제 15 1974년 IBM이 개발하고 1975년 NBS에 의해 미국의 국가 표준으로 발표된 암호화 알고리즘으로, 64비트의 블록 크기와, 56비트의 키 길이, 16회의 라운드를 수행한다. 컴퓨터 기술이 발달함에 따라 해독이 쉬워지면서 미국의 국가 표준이 2001년 AES로 대체되었다. (5점)

답 :

문제 16 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
#include <stdio.h>
int main() {
    int* array[3];
    int a = 12, b = 24, c = 36;
    array[0] = &a;
    array[1] = &b;
    array[2] = &c;
    printf("%d", *array[1] + **array + 1);
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 17 다음 Java로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
public class Test {
	public static void main(String[] args) {
		int w = 3, x = 4, y = 3, z = 5;
		if((w == 2 | w == y) & !(y > z) & (1 == x ^ y != z)) {
			w = x + y;
			if(7 == x ^ y != w)
				System.out.println(w);
			else
				System.out.println(x);
		}
		else {
			w = y + z;
			if(7 == y ^ z != w)
				System.out.println(w);
			else
				System.out.println(z);
		}
	}
}
```

답 :

문제 18 테스트 기법 중 그래프를 활용하여 입력 데이터 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석한 다음 효용성이 높은 테스트 케이스를 선정하여 검사하는 기법을 <보기>에서 찾아 쓰시오. (5점)

<보기>

Equivalence Partition	Boundary Value Analysis	Condition Test
Cause-Effect Graph	Error Guess	Comparison Test
Base Path Test	Loop Test	Data Flow Test

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 19 Windows, MacOS 등에서 사용하는 인터페이스로, 사용자가 명령어를 직접 입력하지 않고 키보드와 마우스 등을 이용하여 아이콘이나 메뉴를 선택하여 모든 작업을 수행하는 사용자 인터페이스를 쓰시오. (5점)

답 :

문제 20 UML의 관계(Relationships)에 관한 다음 설명에서 각 번호(①, ②)에 들어갈 알맞는 용어를 <보기>에서 찾아 쓰시오. (5점)

관계(Relationships)는 사물과 사물 사이의 연관성을 표현하는 것이다. ① - 하나의 사물이 다른 사물에 포함되어 있는 관계로, 전체와 부분으로 구분되어지며 서로 독립적이다. ② - 상위 모듈이 하위 모듈보다 더 일반적인 개념을 가지고 있으며, 하위 모듈이 상위 모듈보다 더 구체적인 개념을 가진다.

<보기>

Association	Aggregation	Composition	Generalization
Dependency	Realization		

답

- ①
- ②

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

기출문제 정답 및 해설

[문제 1]

3

[해설]

이 문제는 객체 변수 _inst가 사용하는 메모리 공간을 객체 변수 conn1, conn2, conn3이 공유함으로써 메모리 낭비를 방지하는 싱글톤(Singleton) 개념을 Java로 구현한 문제입니다.

```
class Connection {                클래스 Connection을 정의한다.
Ⓐ       private static Connection _inst = null;
Ⓑ       private int count = 0;
❷❿⓱    public static Connection get() {
❸⓫⓲            if(_inst == null) {
❹                       _inst = new Connection();
❺                       return _inst;
                }
⓬⓳             return _inst;
        }
❽⓯㉒   public void count() { count++; }
㉔      public int getCount() { return count; }
}

public class Test {
        public static void main(String[] args) {
❶❻              Connection conn1 = Connection.get();
❼               conn1.count();
❾⓭              Connection conn2 = Connection.get();
⓮               conn2.count();
⓰⓴              Connection conn3 = Connection.get();
㉑               conn3.count();
㉓㉕            System.out.print(conn1.getCount());
        }
}
```

Ⓐ Connection 클래스의 객체 변수 _inst를 선언하고 null로 초기화한다.

※ 객체 변수를 생성한다는 것은 **Connection _inst = new Connection();**과 같이 객체 생성 예약어인 new를 통해 heap 영역에 공간을 확보하여 Connection 클래스의 내용을 저장한 후 그 주소를 객체 변수에 저장하는 것인데, Ⓐ에서는 객체 생성 예약어인 new가 생략되었으므로 생성이 아닌 선언만 합니다. 객체 변수를 선언만 하게 되면 heap이 아닌 stack 영역에 내용 없이 저장되어 사용이 불가능합니다. 이후 ❹번과 같이 객체 생성 예약어인 new가 사용되어야만 heap 영역에 내용이 저장되고 그 주소도 객체 변수에 전달되면서 사용 가능한 객체 변수가 됩니다.

Ⓑ 정수형 변수 count를 선언하고, 0으로 초기화한다.

stack 영역	
변수	값
_inst	null
count	0

heap 영역	
주소	내용

모든 Java 프로그램은 반드시 main() 메소드에서 시작한다.

❶ Connection 클래스의 객체 변수 conn1을 선언하고, get() 메소드를 호출한 결과를 저장한다.

※ Ⓐ에서와 같이 객체 변수를 선언만 하였으므로 객체 변수 conn1은 stack 영역에 생성됩니다.

stack 영역	
변수	값
_inst	null
count	0
conn1	

heap 영역	
주소	내용

❷ Connection 형을 반환하는 get() 메소드의 시작점이다.

❸ _inst가 null이면 ❹, ❺번을 수행하고, 아니면 ⓬번으로 이동한다. _inst가 null이므로 ❹번으로 이동한다.

❹ Connection 클래스의 내용을 heap 영역에 저장하고 그 주소를 _inst에 저장한다.

※ Ⓐ에서 객체 변수 _inst는 이미 선언되었으므로, **Connection _inst = new Connection();**과 같이 작성하지 않고 앞쪽의 클래스명을 생략하여 **_inst = new Connection();**과 같이 작성합니다. 생성 예약어인 new를 통해 heap 영역에 공간을 확보하고 Connection 클래스의 내용을 저장한 후 그 주소를 객체 변수 _inst에 저장합니다. 이제 객체 변수 _inst는 Connection() 클래스의 내용이 저장된 heap 영역을 가리키게 됩니다.

stack 영역	
변수	값
_inst	100
count	0
conn1	

heap 영역	
주소	내용
0	
100	private static Connection _inst private int count = 0 static public Connection get() { ... } public void count() { ... } public int getCount() { ... }
200	
300	

❺ _inst에 저장된 값을 메소드를 호출했던 ❻번으로 반환한다.

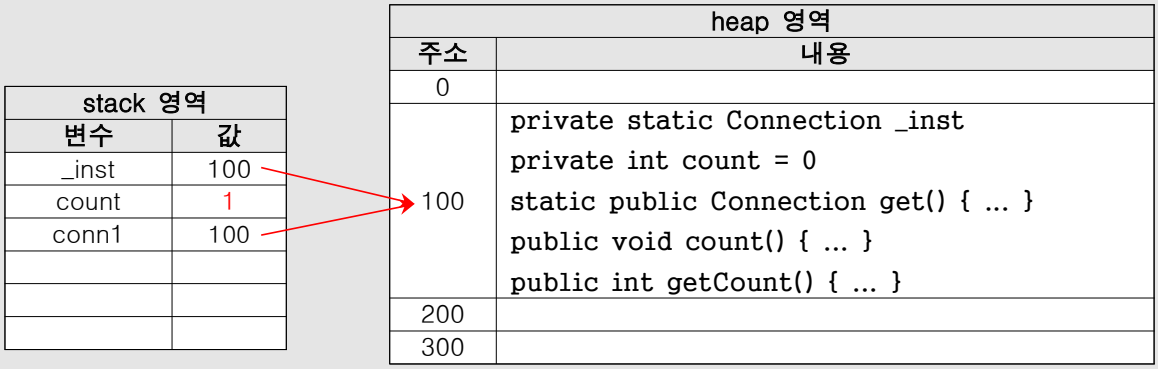
❻ ❺번에서 돌려받은 _inst의 값을 conn1에 저장한다. _inst에는 Connection() 클래스의 내용이 저장된 heap 영역의 주소가 저장되어 있으며, conn1에도 동일한 주소가 저장되므로 이후 _inst와 conn1은 같은 heap 영역의 주소를 가리키게 된다.

stack 영역	
변수	값
_inst	100
count	0
conn1	100

heap 영역	
주소	내용
0	
100	private static Connection _inst private int count = 0 static public Connection get() { ... } public void count() { ... } public int getCount() { ... }
200	
300	

❼ conn1의 count() 메소드를 호출한다. conn1은 Connection() 클래스의 객체 변수이므로 Connection 클래스의 count() 메소드를 호출한다는 의미이다.

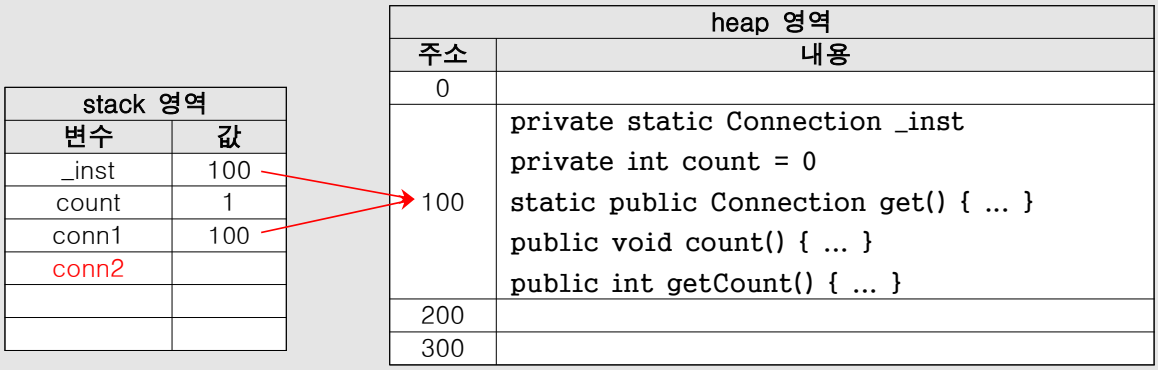
❽ 반환값이 없는 count() 메소드의 시작점이다. count의 값에 1을 더한다.



stack 영역	
변수	값
_inst	100
count	1
conn1	100

heap 영역	
주소	내용
0	
100	private static Connection _inst private int count = 0 static public Connection get() { ... } public void count() { ... } public int getCount() { ... }
200	
300	

❾ Connection 클래스의 객체 변수 conn2를 선언하고, get() 메소드를 호출한 결과를 저장한다.



stack 영역	
변수	값
_inst	100
count	1
conn1	100
conn2	

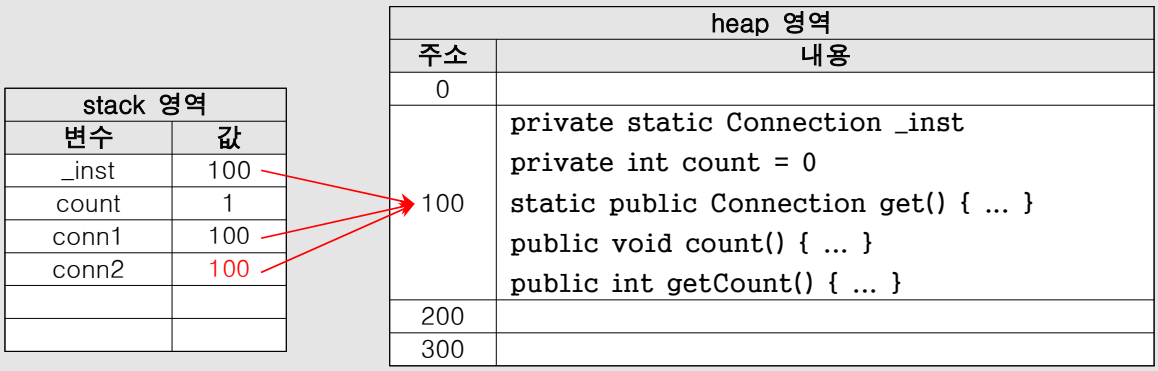
heap 영역	
주소	내용
0	
100	private static Connection _inst private int count = 0 static public Connection get() { ... } public void count() { ... } public int getCount() { ... }
200	
300	

❿ Connection 형을 반환하는 get() 메소드의 시작점이다.

⓫ _inst가 null이면 ❹, ❺번을 수행하고, 아니면 ⓬번으로 이동한다. _inst에는 ❹번에서 저장한 heap 영역의 주소가 저장되어 있어 null이 아니므로 ⓬번으로 이동한다.

⓬ _inst에 저장된 값을 메소드를 호출했던 ⓭번으로 반환한다.

⓭ ⓬번에서 돌려받은 _inst의 값을 conn2에 저장한다.

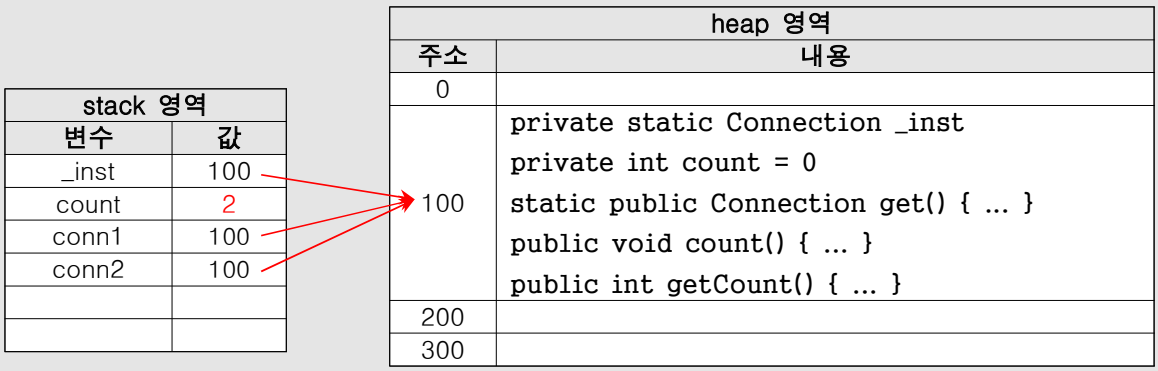


stack 영역	
변수	값
_inst	100
count	1
conn1	100
conn2	100

heap 영역	
주소	내용
0	
100	private static Connection _inst private int count = 0 static public Connection get() { ... } public void count() { ... } public int getCount() { ... }
200	
300	

⓮ conn2의 count() 메소드를 호출한다.

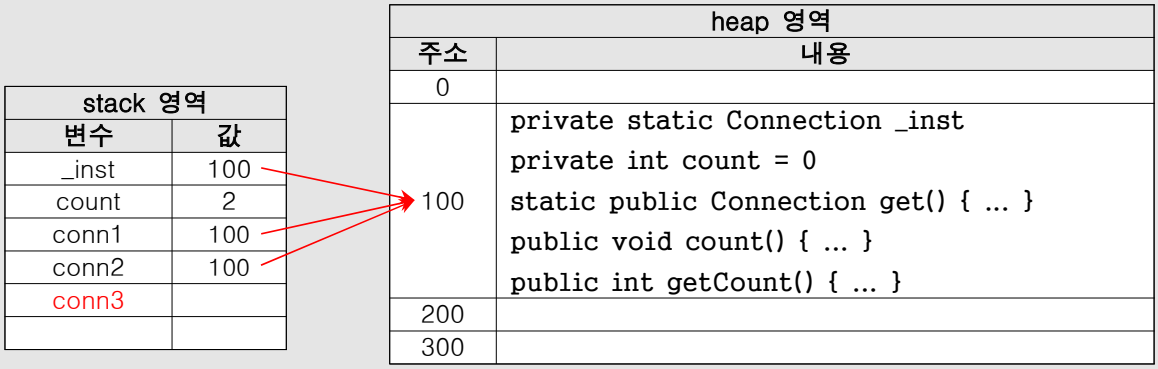
⓯ 반환값이 없는 count() 메소드의 시작점이다. count의 값에 1을 더한다.



stack 영역	
변수	값
_inst	100
count	2
conn1	100
conn2	100

heap 영역	
주소	내용
0	
100	private static Connection _inst private int count = 0 static public Connection get() { ... } public void count() { ... } public int getCount() { ... }
200	
300	

⓰ Connection 클래스의 객체 변수 conn3을 선언하고, get() 메소드를 호출한 결과를 저장한다.



stack 영역	
변수	값
_inst	100
count	2
conn1	100
conn2	100
conn3	

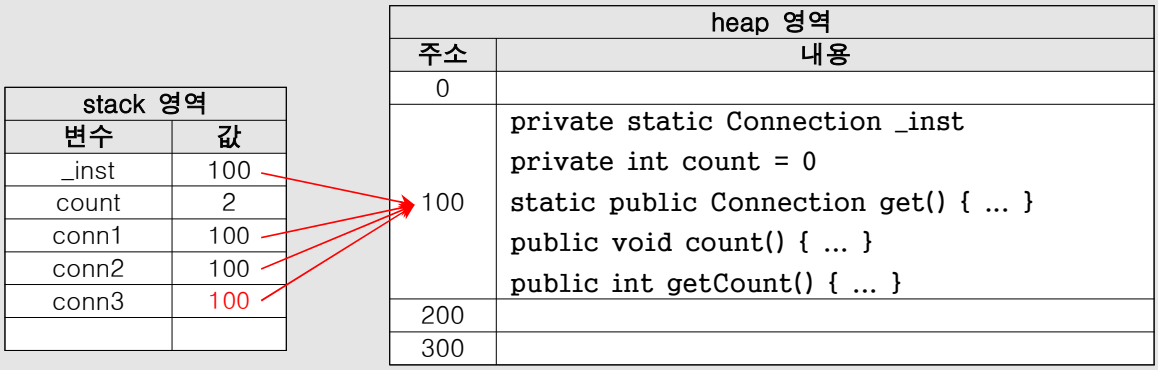
heap 영역	
주소	내용
0	
100	private static Connection _inst private int count = 0 static public Connection get() { ... } public void count() { ... } public int getCount() { ... }
200	
300	

⓱ Connection 형을 반환하는 get() 메소드의 시작점이다.

⓲ _inst가 null이면 ❹, ❺번을 수행하고, 아니면 ⓳번으로 이동한다. _inst가 null이 아니므로 ⓳번으로 이동한다.

⓳ _inst에 저장된 값을 메소드를 호출했던 ⓴번으로 반환한다.

⓴ ⓳번에서 돌려받은 _inst의 값을 conn3에 저장한다.

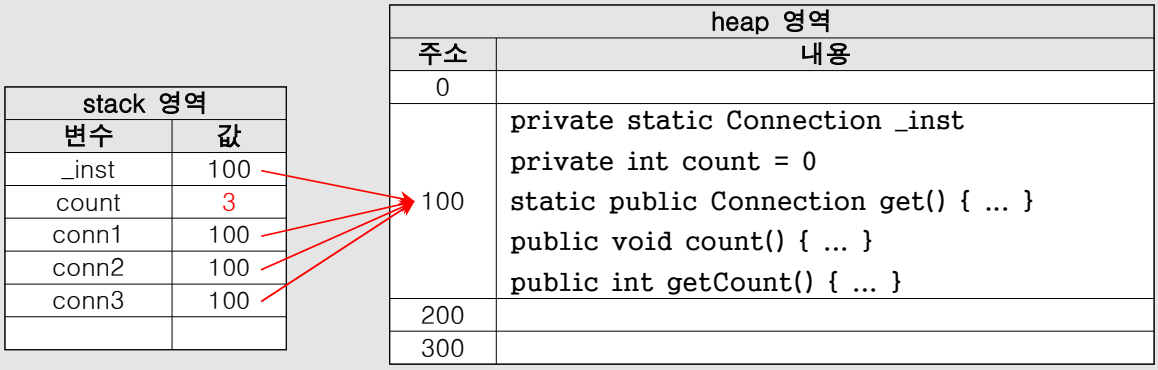


stack 영역	
변수	값
_inst	100
count	2
conn1	100
conn2	100
conn3	100

heap 영역	
주소	내용
0	
100	private static Connection _inst private int count = 0 static public Connection get() { ... } public void count() { ... } public int getCount() { ... }
200	
300	

㉑ conn3 객체 변수의 count() 메소드를 호출한다.

㉒ 반환값이 없는 count() 메소드의 시작점이다. count의 값에 1을 더한다.



stack 영역	
변수	값
_inst	100
count	3
conn1	100
conn2	100
conn3	100

heap 영역	
주소	내용
0	
100	private static Connection _inst private int count = 0 static public Connection get() { ... } public void count() { ... } public int getCount() { ... }
200	
300	

㉓ conn1의 getCount() 메소드를 호출하고 돌려받은 값을 출력한다.

㉔ 정수를 반환하는 getCount() 메소드의 시작점이다. count의 값 3을 메소드를 호출했던 ㉕번으로 반환한다.

※ 객체 변수 _inst, conn1, conn2, conn3은 모두 같은 heap 영역의 주소를 가리키고 있으므로 해당 heap 영역에 저장된 내용을 공유하게 됩니다.

㉕ 화면에 3을 출력한다.

결과 3

[문제 2]

ARP(Address Resolution Protocol)

[문제 3]

※ 다음 중 밑줄이 표시된 내용은 반드시 포함되어야 합니다.

데이터베이스 관리자가 데이터베이스 사용자에게 권한을 부여하는 데 사용하는 명령어

[문제 4]

① Authentication ② Authorization ③ Accounting

[문제 5]

Factory Method

[문제 6]

※ 다음 중 하나를 쓰면 됩니다.

Control, Control Coupling

[문제 7]

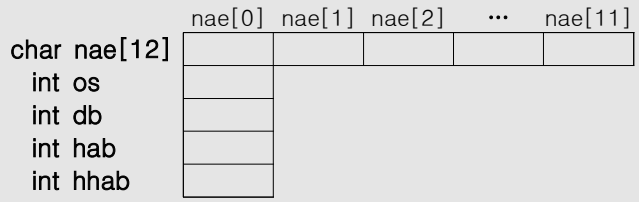
501

[해설]

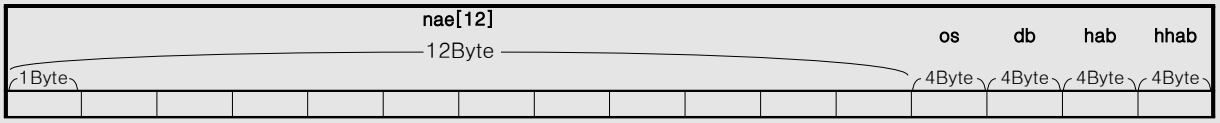
```
#include <stdio.h>
Ⓐ struct jsu {                  구조체 jsu를 정의한다. 구조체를 정의한다는 것은 int나 char 같은 자료형을
                                하나 만든다는 의미다. 구조체의 멤버를 지정할 때는 [변수명].[멤버이름]으로
                                지정하지만, 포인터 변수를 이용해 구조체의 멤버를 지정할 때는 [변수명]->
                                [멤버이름]으로 지정한다.
                                ㆍ구조체(struct) : 배열이 자료의 형과 크기가 동일한 변수의 모임이라면, 구
                                  조체는 자료의 종류가 다른 변수의 모임임
                                ㆍ멤버(member) : 일반 변수를 선언하는 것과 동일하게 필요한 변수들을 임의
                                  로 선언하면 됨
    char nae[12];               12개의 요소를 갖는 문자 배열 nae를 선언한다.
    int os, db, hab, hhab;      정수형 변수 os, db, hab, hhab를 선언한다.
};

int main() {
❶   struct jsu st[3] = { {"데이터1", 95, 88}, {"데이터2", 84, 91}, {"데이터3", 86, 75} };
❷   struct jsu* p;
❸   p = &st[0];
❹   (p + 1)->hab = (p + 1)->os + (p + 2)->db;
❺   (p + 1)->hhab = (p + 1)->hab + p->os + p->db;
❻   printf("%d", (p + 1)->hab + (p + 1)->hhab);
}
```

Ⓐ 구조체 jsu의 구조



※ 위의 구조체는 다음과 같이 메모리의 연속된 공간에 저장된 후 사용됩니다.



모든 C 프로그램은 반드시 main() 함수에서 시작한다

❶ 구조체 jsu 자료형으로 3개짜리 배열 st를 선언하고 초기화한다.

	char nae[12]	int os	int db	int hab	int hhab
st[0]	st[0].nae[0]~st[0].nae[11]	st[0].os	st[0].db	st[0].hab	st[0].hhab
st[1]	st[1].nae[0]~st[1].nae[11]	st[1].os	st[1].db	st[1].hab	st[1].hhab
st[2]	st[2].nae[0]~st[2].nae[11]	st[2].os	st[2].db	st[2].hab	st[2].hhab

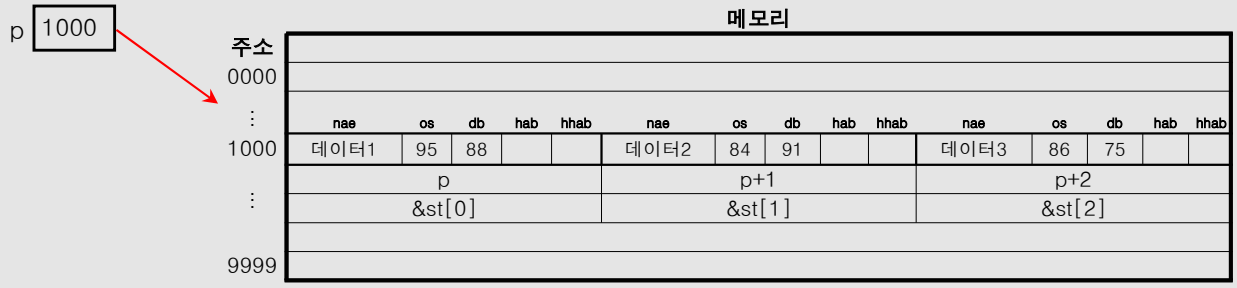
↓

	char nae[12]	int os	int db	int hab	int hhab
st[0]	데 이 터 1 \0	95	88		
st[1]	데 이 터 2 \0	84	91		
st[2]	데 이 터 3 \0	86	75		

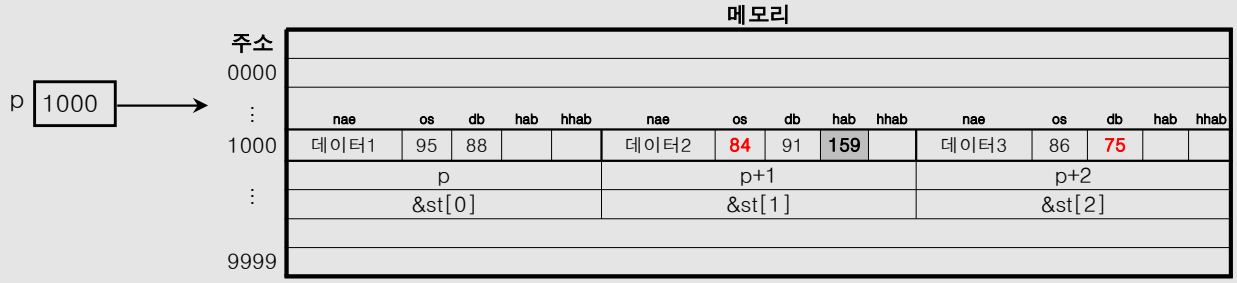
※ 문자열을 저장하는 경우 문자열의 끝을 의미하는 널 문자(\0)가 추가로 저장되며, 출력 시 널 문자는 표시되지 않습니다. 또한 영문, 숫자는 1Byte, 한글은 2Byte를 차지합니다.

❷ 구조체 jsu의 포인터 변수 p를 선언한다.

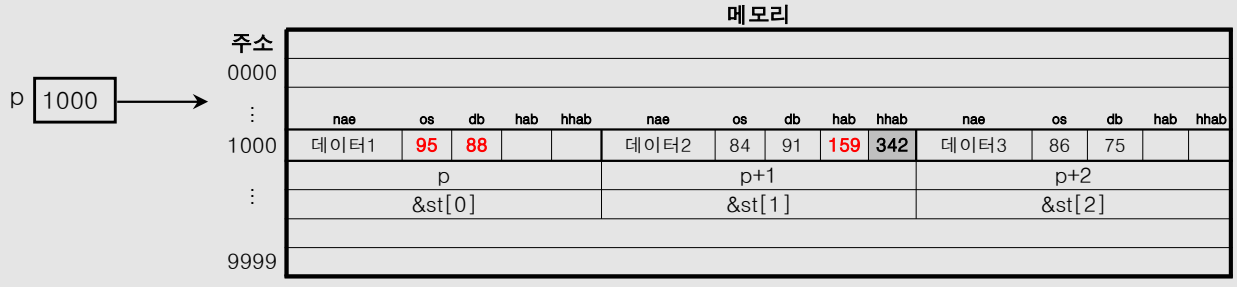
❸ p에 st 배열의 첫 번째 요소의 주소를 저장한다. 주소는 임의로 정한 것이다.



❹ p+1이 가리키는 곳의 hab에 p+1이 가리키는 곳의 os 값과 p+2가 가리키는 곳의 db 값을 더한 후 저장한다. p가 st[0]을 가리키므로 p+1은 st[1]을 p+2는 st[2]를 가리킨다. 따라서 st[1]의 os 값 84와 st[2]의 db 값 75를 더한 값 159를 st[1]의 hab에 저장한다.



❺ p+1이 가리키는 곳의 hhab에 p+1이 가리키는 곳의 hab 값과 p가 가리키는 곳의 os와 db 값을 모두 더한 후 저장한다. st[1]의 hab 값 159, st[0]의 os와 db 값 95와 88을 모두 더한 값 342를 st[1]의 hhab에 저장한다.



❻ p+1이 가리키는 곳의 hab와 hhab의 값을 더한 후 정수로 출력한다. 159와 342를 더한 501가 출력된다.

결과 501

[문제 8]

① 상향식 통합 테스트

※ 다음 중 하나를 쓰면 됩니다.

② 드라이버, 테스트 드라이버, Driver, Test Driver

[문제 9]

False

[답안 작성 시 주의 사항]

C, Java, Python 등의 프로그래밍 언어에서는 대소문자를 구분하기 때문에 출력 결과도 대소문자를 구분하여 정확하게 작성해야 합니다. 예를 들어, 소문자로 **false**로 썼을 경우 부분 점수 없이 완전히 틀린 것으로 간주됩니다.

[해설]

```
❶ x, y = 100, 200
❷ print(x==y)
```

❶ 변수 x, y를 선언하고 각각 100, 200으로 초기화한다.

❷ x의 값 100과 y의 값 200이 같으면 참(True)을, 같지 않으면 거짓(False)을 출력한다.

결과 False

[문제 10]

4

[해설]

```
SELECT COUNT(*) CNT
FROM A CROSS JOIN B
WHERE A.NAME LIKE B.RULE;
```

질의문은 각 절을 분리하여 이해하면 쉽습니다.

- **SELECT COUNT(*) CNT** : 튜플의 개수를 표시하되, 필드명은 'CNT'로 표시합니다.
  ※ 'SELECT COUNT(*) AS CNT'에서 AS가 생략된 형태입니다.
- **FROM A CROSS JOIN B** : <A>와 <B>를 교차 조인(CROSS JOIN)한 결과를 대상으로 검색합니다.

A.NAME	B.RULE
Smith	S%
Smith	%T%
Allen	S%
Allen	%T%
Scott	S%
Scott	%T%

- **WHERE A.NAME LIKE B.RULE** : <A> 테이블의 'NAME' 필드 값이 <B> 테이블의 'RULE' 필드에 저장된 문자열 패턴과 일치하는 튜플만을 대상으로 합니다.
  ※ <B> 테이블의 'RULE' 필드에 저장된 값은 'S%'와 '%T%'와 같이 문자 패턴인 '%' 기호가 포함되어 있으므로, 조건문의 LIKE 연산자와 결합되면 다음과 같이 적용됩니다.

- A.NAME LIKE S% : 'A.NAME'이 "S"로 시작하는 레코드를 검색

NAME	RULE
Smith	S%
Smith	%T%
Allen	S%
Allen	%T%
Scott	S%
Scott	%T%

- A.NAME LIKE %T% : 'A.NAME'이 "T"를 포함하는 레코드를 검색

NAME	RULE
Smith	S%
Smith	%T%
Allen	S%
Allen	%T%
Scott	S%
Scott	%T%

※ CROSS JOIN된 결과에서 조건을 만족하는 튜플은 다음과 같습니다. 그러므로 검색된 튜플의 개수는 4입니다.

NAME	RULE
Smith	S%
Smith	%T%
Scott	S%
Scott	%T%

[문제 11]

※ 다음 중 하나를 쓰면 됩니다.

색인, Index

[문제 12]

① 테스트 조건 ② 테스트 데이터 ③ 예상 결과

[문제 13]

※ 다음 중 하나를 쓰면 됩니다.

클래스, Class

[문제 14]

※ 각 문항별로 다음 중 하나를 쓰면 됩니다.

① 데이터 링크 계층, Data Link Layer

② 네트워크 계층, 망 계층, Network Layer

③ 표현 계층, Presentation Layer

[문제 15]

※ 다음 중 하나를 쓰면 됩니다.

DES(Data Encryption Standard)

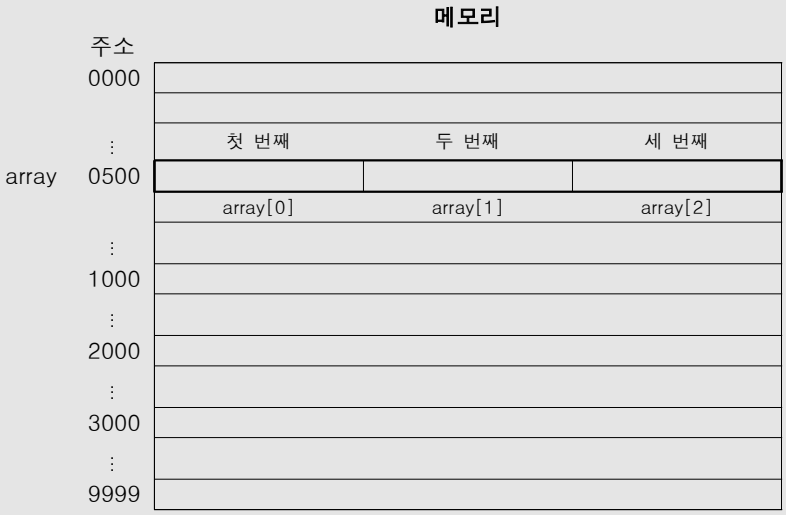
[문제 16]

37

[해설]

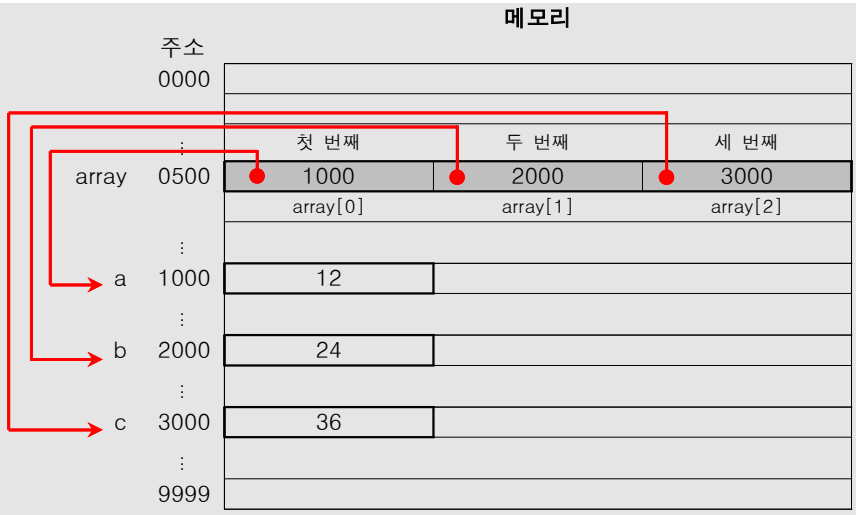
```
#include <stdio.h>
int main() {
❶    int* array[3];
❷    int a = 12, b = 24, c = 36;
❸    array[0] = &a;
❹    array[1] = &b;
❺    array[2] = &c;
❻    printf("%d", *array[1] + **array + 1);
}
```

❶ 3개의 요소를 갖는 정수형 포인터 배열 array를 선언한다. 주소는 임의로 정한 것이다.



❷ 정수형 변수 a, b, c에 각각 12, 24, 36을 저장한다.

❸ array[0]에 a의 주소를 저장한다.
❹ array[1]에 b의 주소를 저장한다.
❺ array[2]에 c의 주소를 저장한다.



❻ array[1]이 가리키는 곳의 값과 *array가 가리키는 곳의 값과 1을 더한 후 정수로 출력한다.

- *array[1] : array[1]에는 2000이 저장되어 있고 2000이 가리키는 곳의 값은 24이다.
- **array
  - array : 배열의 이름만 지정하면 배열의 첫 번째 요소의 주소인 &array[0], 즉 500을 의미한다.
  - *array : array는 500이고 500이 가리키는 곳의 값은 1000이다.
  - **array : *array는 1000이고 1000이 가리키는 곳의 값은 12이다.

∴ 24+12+1 = 37

결과 37

[문제 17]

7

[해설]

```
public class Test {
	public static void main(String[] args) {
❶		int w = 3, x = 4, y = 3, z = 5;
❷		if((w == 2 | w == y) & !(y > z) & (1 == x ^ y != z)) {
❸			w = x + y;
❹			if(7 == x ^ y != w)
❺				System.out.println(w);
			else
				System.out.println(x);
		} ❻
		else {
			w = y + z;
			if(7 == y ^ z != w)
				System.out.println(w);
			else
				System.out.println(z);
		} ❼
	}
}
```

❶ 정수형 변수 w, x, y, z를 선언하고 각각 3, 4, 3, 5로 초기화한다.

❷ 조건이 참이면 ❸번부터 ❻번 이전까지의 문장을, 거짓이면 ❻번 아래 else의 다음 문장부터 ❼번 이전까지의 문장을 수행한다. 연산자 우선순위에 따라 다음의 순서로 조건의 참/거짓을 확인한다.

- (w == 2 ① | w == y ②) & !(y > z ③) & (1 == x ^ y ④ != z ⑤) ⑦
  ⑥
  ⑧
  ⑨
- ① : w의 값 3과 2는 같지 않으므로 거짓(0)이다.
- ② : w의 값 3과 y의 값 3은 같으므로 참(1)이다.
- ③ : y의 값 3은 z의 값 5보다 크지 않으므로 거짓(0)이지만, 앞에 !(논리 not)가 있으므로 참(1)이다.
- ④ : x의 값 4와 y의 값 3을 ^(비트 xor) 연산하면
  0100(4)
  ^ 0011(3)
  0111(7)
  이므로 결과는 7이다.
- ⑤ 1 == ④ : 1과 ④의 결과 7은 같지 않으므로 거짓(0)이다.
- ⑥ ① | ② : ①의 결과 0과 ②의 결과 1을 |(비트 or) 연산하면
  0000(0)
  | 0001(1)
  0001(1)
  이므로 결과는 1이다.
- ⑦ ⑤ != z : ⑤의 결과 0과 z의 값 5는 같지 않으므로 참(1)이다.
- ⑧ ⑥ & ③ : ⑥의 결과 1과 ③의 결과 1을 &(비트 and) 연산하면
  0001(1)
  & 0001(1)
  0001(1)
  이므로 결과는 1이다.
- ⑨ ⑧ & ⑦ : ⑧의 결과 1과 ⑦의 결과 1을 &(비트 and) 연산하면 결과는 1이다.

∴ 최종 결과는 1이며, 1은 조건에서 참을 의미하므로 ❸번으로 이동한다.

❸ w에 x와 y의 합을 저장한다. (w=7)

❹ 조건이 참이면 ❺번 문장을, 거짓이면 ❺번 아래 else 다음 문장을 수행한다. 연산자 우선순위에 따라 다음

의 순서로 조건의 참/거짓을 확인한다.

• 7 == x ^ y != w
①
②
③

• ① : x의 값 4와 y의 값 3을 ^(비트 xor) 연산하면 결과는 7이다.

• ② **7 == ①** : 7과 ①의 결과 7은 같으므로 결과는 참(1)이다.

• ③ **② != w** : ②의 결과 1과 w의 값 7은 같지 않으므로 결과는 참(1)이다.

∴ 최종 결과는 1이며, 1은 조건에서 참을 의미하므로 ❺번 문장을 수행한다.

❺ w의 값 **7**을 출력하고 커서를 다음 줄의 처음으로 옮긴다. 모든 if문이 종료되었으므로 ❼번으로 이동하여 프로그램을 종료한다.

결과 7

[문제 18]

Cause Effect Graph

[문제 19]

※ 다음 중 하나를 쓰면 됩니다.

GUI, Graphical User Interface, Graphic User, Interface, 그래픽 사용자 인터페이스

[문제 20]

① Aggregation ② Generalization