기출문제
2024년 1회 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 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
#include <stdio.h>
#include <stdlib.h>

main(int argc, char *argv[]) {
        int v1 = 0;
        int v2 = 35;
        int v3 = 29;
        if (v1 > v2 ? v2 : v1)
                v2 = v2 << 2;
        else
                v3 = v3 << 2;
        printf("%d", v2 + v3);
        return 0;
}
```

답 :

문제 2 다음 설명에 해당하는 라우팅 프로토콜을 쓰시오. (5점)

- RIP의 단점을 해결하여 새로운 기능을 지원하는 인터넷 프로토콜이다.
- 최단 경로 탐색에 Dijkstra 알고리즘을 사용한다.
- 대규모 네트워크에서 많이 사용된다.
- 링크 상태를 실시간으로 반영하여 최단 경로로 라우팅을 지원한다.

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 3 다음의 정규화(Normalization) 과정은 어떤 단계의 정규화 과정인지 <보기>에서 찾아 쓰시오. (5점)

〈주문〉

주문번호	고객번호	주소
A345	100	서울
D347	200	부산
A210	300	광주
B230	200	부산

⬇

〈주문〉

주문번호	고객번호
A345	100
D347	200
A210	300
B230	200

〈고객〉

고객번호	주소
100	서울
200	부산
300	광주

<보기>

• 제1정규형	• 제2정규형	• 제3정규형
• 보이스/코드 정규형	• 제4정규형	• 제5정규형

답 :

문제 4 다음에 제시된 응집도(Cohesion)를 높은 순에서 낮은 순으로 나열하시오. (5점)

㉠ 기능적 응집도(Functional Cohesion)
㉡ 교환적 응집도(Communication Cohesion)
㉢ 우연적 응집도(Coincidental Cohesion)
㉣ 시간적 응집도(Temporal Cohesion)

답 : ( ) → ( ) → ( ) → ( )

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 5 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;
        public static 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();
                conn1.count();
                System.out.print(conn1.getCount());
        }
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 6 다음 설명에 해당하는 디자인 패턴을 <보기>에서 찾아 쓰시오. (5점)

• 구체적인 클래스에 의존하지 않고, 인터페이스를 통해 서로 연관·의존하는 객체들의 그룹으로 생성하여 추상적으로 표현한다. • 키트(Kit) 패턴이라고도 불린다. • 연관된 서브 클래스를 묶어 한 번에 교체하는 것이 가능하다.

<보기>

생성 패턴	구조 패턴	행위 패턴
Abstract Factory	Adapter	Command
Builder	Bridge	Interpreter
Factory Method	Composite	Iterator
Prototype	Decorator	Mediator
Singleton	Proxy	Observer

답 :

문제 7 3개의 페이지를 수용할 수 있는 주기억장치가 있으며, 초기에는 모두 비어 있다고 가정한다. 다음의 순서로 페이지 참조가 발생할 때, LRU와 LFU 페이지 교체 알고리즘을 사용하면 각각 발생하는 페이지 결함의 횟수를 쓰시오. (5점)

LRU	페이지 참조 순서
	1, 2, 3, 1, 2, 4, 5, 1
LFU	페이지 참조 순서
	1, 2, 3, 1, 2, 4, 1, 2, 3, 4

답

- ① LRU :
- ② LFU :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 8 다음 Python으로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
a = [ 'Seoul', 'Kyeonggi', 'Incheon', 'Daejeon', 'Daegu', 'Pusan'];
str01 = 'S'
for i in a:
        str01 = str01 + i[1]
print(str01)
```

답 :

문제 9 다음은 조인(Join)에 대한 설명이다. 괄호(①~③)에 들어갈 알맞은 조인의 종류를 <보기>에서 찾아 쓰시오. (5점)

- (①)은 조인에 참여하는 두 릴레이션의 속성 값을 비교하여 조건을 만족하는 튜플만 반환하는 조인이다.
- (②)은 (①)에서 = 연산자를 사용한 조인으로, 일반적으로 조인이라고 하면 (②)을 의미한다.
- (②)의 결과 릴레이션의 차수는 첫 번째 릴레이션과 두 번째 릴레이션의 차수를 합한 것이다.
- (③)은 (②)의 결과 릴레이션에서 중복된 속성을 제거하여 수행하는 연산, 즉 (②)에서 중복 속성 중 하나가 제거된 것이다.
- (③)의 핵심은 두 릴레이션의 공통된 속성을 매개체로 하여 두 릴레이션의 정보를 '관계'로 묶어내는 것이다.

<보기>

• 자연 조인	• 외부 조인	• 셀프 조인	• 세타 조인
• 동등 조인	• 교차 조인	• 자연 조인	• 자동 조인

답

- ①
- ②
- ③

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 10 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
#include <stdio.h>
#include <string.h>

void inverse(char *str, int len) {
	for(int i = 0, j = len - 1; i < j; i++, j--) {
		char ch = str[i];
		str[i] = str[j];
		str[j] = ch;
	}
}

int main() {
	char str[100] = "ABCDEFGH";
	int len = strlen(str);
	inverse(str, len);
	for(int i = 1; i < len; i += 2) {
		printf("%c", str[i]);
	}
	return 0;
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 11 다음 Java로 구현된 프로그램의 실행 순서를 나열하시오(단, 같은 번호는 중복해서 작성하지 마시오). (5점)

```
    class Parent {
        int x, y;

①       Parent(int x, int y) {
            this.x = x;
            this.y = y;
        }

②       int getX() {
            return x*y;
        }
    }

    class Child extends Parent {
        int x;

③       Child(int x) {
            super(x+1, x);
            this.x = x;
        }

④       int getX(int n) {
            return super.getX() + n;
        }
    }

    public class Main {
⑤       public static void main(String[] args) {
⑥           Parent parent = new Child(10);
⑦           System.out.println(parent.getX());
        }
    }
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 12 다음 <R1>과 <R2> 테이블을 참조하여 <SQL문>을 실행했을 때 출력되는 결과를 쓰시오. (SQL을 실행하였을 때 출력되는 속성명과 값들을 모두 답안에 적으시오.) (5점)

<R1>

A	B	C
1	a	x
2	b	y
3	c	t

<R2>

C	D	E
x	k	k
y	k	t
z	p	k

<SQL문>

```
SELECT B
FROM R1
WHERE C IN (SELECT C FROM R2 WHERE D='k');
```

답 :

문제 13 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
#include <stdio.h>
#include <ctype.h>

int main( ) {
        char *p = "It is 8";
        char result[100];
        int i;
        for(i = 0; p[i] != '\0'; i++) {
                if(isupper(p[i]))
                        result[i] = (p[i] - 'A'+ 5) % 25 + 'A';
                else if(islower(p[i]))
                        result[i] = (p[i] - 'a'+ 10) % 26 + 'a';
                else if(isdigit(p[i]))
                        result[i] = (p[i] - '0'+ 3) % 10 + '0';
                else if(!(isupper(p[i]) || islower(p[i]) || isdigit(p[i])))
                        result[i] = p[i];
        }
        result[i] = '\0';
        printf("변환된 문자열 : %s₩n", result);
        return 0;
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 14 다음 설명에 해당하는 커버리지(Coverage)를 <보기>에서 찾아 쓰시오. (5점)

• 개별 조건식이 다른 개별 조건식의 영향을 받지 않고 전체 조건식의 결과에 독립적으로 영향을 주는 구조적 테스트 케이스이다. • 해당 개별 조건식이 전체 조건식의 결과에 영향을 주는 조건 조합을 찾아 커버리지를 테스트하는 방법이다. • 프로그램에 있는 모든 결정 포인트 내의 전체 조건식이 적어도 한 번은 참과 거짓을 만족해야 한다. • 프로그램에 있는 결정 포인트 내의 모든 개별 조건식이 적어도 한 번은 참과 거짓을 만족해야 한다.

<보기>

• All Path	• Multiple Condition	• MC/DC	• Condition/Decision
• Condition	• Decision	• Statement	

답 :

문제 15 보안 위협에 대한 다음 설명에 해당하는 용어를 <보기>에서 찾아 쓰시오. (5점)

• 시스템에 침입한 후 침입 사실을 숨긴 채 백도어, 트로이목마를 설치하고, 원격 접근, 내부 사용 흔적 삭제, 관리자 권한 획득 등 주로 불법적인 해킹에 사용되는 기능들을 제공하는 프로그램들의 모음이다. • 자신 또는 다른 소프트웨어의 존재를 감춰줌과 동시에 허가되지 않은 컴퓨터나 소프트웨어의 영역에 접근할 수 있게 하는 용도로 설계되었다. • 이 프로그램이 설치되면 자신이 뚫고 들어온 모든 경로를 바꾸어 놓고, 명령어들을 은폐해 놓기 때문에 해커가 시스템을 원격에서 해킹하고 있어도 이 프로그램이 설치되어 있는 사실조차 감지하기 어렵다. • 공격자가 보안 관리자나 보안 시스템의 탐지를 피하면서 시스템을 제어하기 위해 설치하는 악성 프로그램으로, 운영체제의 합법적인 명령어를 해킹하여 모아놓았다. • 운영체제에서 실행 파일과 실행 중인 프로세스를 숨김으로써 운영체제 검사 및 백신 프로그램의 탐지를 피할 수 있다.

<보기>

• Worm	• Logic Bomb	• Spyware	• Honeypot
• Bug Bounty	• Rootkit	• Bootkit	• Ransomware

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 16 다음 설명에 해당하는 용어를 <보기>에서 찾아 쓰시오. (5점)

다양한 IT 기술과 방식들을 이용해 조직적으로 특정 기업이나 조직 네트워크에 침투해 활동 거점을 마련한 뒤 때를 기다리면서 보안을 무력화시키고 정보를 수집한 다음 외부로 빼돌리는 형태의 공격으로, 일반적으로 공격은 침투, 검색, 수집, 유출의 4단계로 실행된다.

- 침투(Infiltration) : 목표로 하는 시스템을 악성코드로 감염시켜 네트워크에 침투한다.
- 검색(Exploration) : 시스템에 대한 정보를 수집하고 기밀 데이터를 검색한다.
- 수집(Collection) : 보호되지 않은 시스템의 데이터를 수집하고, 시스템 운영을 방해하는 악성코드를 설치한다.
- 유출(Exfiltration) : 수집한 데이터를 외부로 유출한다.

<보기>

• MITM	• ATM	• XDR	• APT
• Key Logger Attack	• 사회공학 기법	• TearDrop	• SMURFING

답 :

문제 17 <EMP_TBL> 테이블을 참고하여 <SQL문>의 실행 결과를 쓰시오. (5점)

<EMP_TBL>

EMPNO	SAL
100	1500
200	3000
300	2000

<처리 조건>

```
SELECT COUNT(*) FROM EMP_TBL WHERE EMPNO > 100 AND SAL >= 3000 OR EMPNO = 200;
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 18 다음 JAVA로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
class firstArea {
        int x, y;
        public firstArea(int x, int y) {
                this.x = x;
                this.y = y;
        }
        public void print() {
                System.out.println(x+y);
        }
}

class secondArea extends firstArea {
        int bb = 3;
        public secondArea(int i) {
                super(i, i+1);
        }
        public void print() {
                System.out.println(bb*bb);
        }
}

public class Main {
        public static void main(String[] args) {
                firstArea st = new secondArea(10);
                st.print();
        }
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

한국산업인력공단에서 시험 문제를 공개하지 않아 수험생의 기억을 토대로 문제 대부분을 재구성하였으나, [문제 19]와 [문제 20]은 수험생의 기억을 토대로 재구성하기에 어려움이 있었습니다. 이점 양해 바랍니다.

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

기출문제 정답 및 해설

[문제 1]

151

[해설]

```
#include <stdio.h>
#include <stdlib.h>

main(int argc, char *argv[]) {
❶       int v1 = 0;
❷       int v2 = 35;
❸       int v3 = 29;
❹       if (v1 > v2 ? v2 : v1)
                v2 = v2 << 2;
        else
❺               v3 = v3 << 2;
❻       printf("%d", v2 + v3);
❼       return 0;
}
```

❶ 정수형 변수 v1을 선언하고 0으로 초기화한다.

❷ 정수형 변수 v2를 선언하고 35로 초기화한다.

❸ 정수형 변수 v3을 선언하고 29로 초기화한다.

❹ v1이 v2보다 크면 v2의 값을 조건으로 사용하고 그렇지 않으면 v1의 값을 조건으로 사용한다. 조건은 결과가 0이면 거짓이고, 나머지는 참이다. 0은 35보다 크지 않으므로 v1의 값 0을 조건으로 사용한다. 조건의 결과가 0, 즉 거짓이므로 ❺번 문장으로 이동한다.

❺ <<는 왼쪽 시프트 연산자로, v3에 저장된 값을 왼쪽으로 2비트 이동시킨 후 v3에 저장한다. 정수는 4Byte를 사용하므로 29를 4Byte 2진수로 변환하여 계산하면 된다.

- 29를 4Byte 2진수로 표현하면 다음과 같다.

	32	31	30	29	…	9	8	7	6	5	4	3	2	1
29	0	0	0	0	…	0	0	0	0	1	1	1	0	1
부호 비트					…		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
					…		128	64	32	16	8	4	2	1

- 부호를 제외한 전체 비트를 왼쪽으로 2비트 이동시킨다. 양수이므로 패딩 비트(빈자리)에는 0이 채워진다.

	32	31	30	29	…	9	8	7	6	5	4	3	2	1	
116	0	0	0	0	…	0	0	1	1	1	0	1	0	0	패딩 비트
부호 비트					…		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
					…		128	64	32	16	8	4	2	1	

- 이동된 값을 10진수로 변환하면 116이다. v3에는 116이 저장된다.

❻ v2+v3의 결과 151을 정수형으로 출력한다.

결과 151

❼ main() 함수에서의 'return 0'은 프로그램의 종료를 의미한다.

[문제 2]

※ 다음 중 하나를 쓰면 됩니다.

OSPF, Open Shortest Path First protocol

[문제 3]

제3정규형

[해설]

- <주문> 테이블에서 '고객번호'가 '주문번호'에 함수적 종속이고, '주소'가 '고객번호'에 함수적 종속이므로 '주소'는 기본키인 '주문번호'에 대해 이행적 함수적 종속을 만족한다. 즉 주문번호 → 고객번호이고, 고객번호 → 주소이므로 주문번호 → 주소는 이행적 함수적 종속이 된다.
- <주문> 테이블에서 이행적 함수적 종속(즉 주문번호 → 주소)을 제거하여 <주문> 테이블과 <고객> 테이블로 무손실 분해함으로써 제3정규형이 되었다.

[문제 4]

㉠, ㉡, ㉣, ㉤

[문제 5]

4

[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;
        }
❽⓯㉒㉔ public void count() { count++; }
㉖      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();
㉓              conn1.count();
㉕㉗            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.

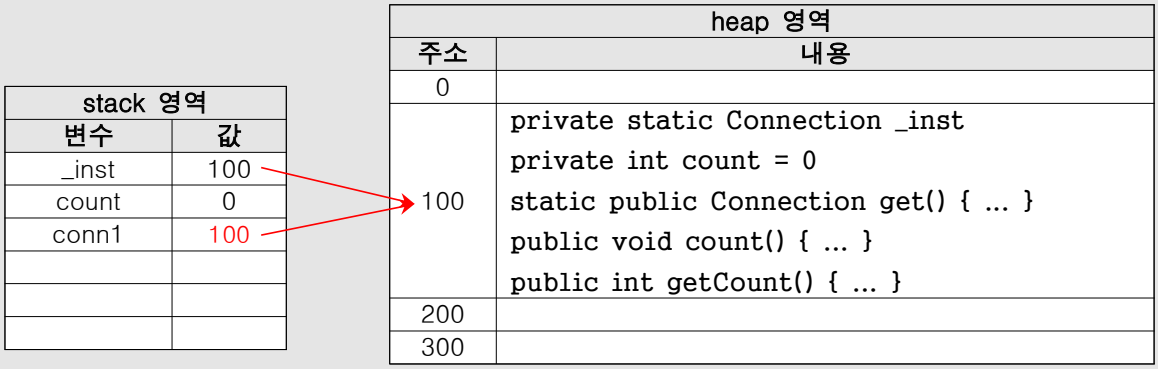
※ Ⓐ에서 객체 변수 _inst는 이미 선언되었으므로, **Connection _inst = new Connection();**과 같이 작성하지 않고 앞쪽의 클래스명을 생략하여 **_inst = new Connection();**과 같이 작성합니다. 생성 예약어인 new를 통해 heap 영역에 공간을 확보하고 Connection 클래스의 내용을 저장한 후 그 주소를 객체 변수 _inst에 저장합니다. 이제 객체 변수 _inst는 Connection() 클래스의 내용이 저장된 heap 영역을 가리키게 됩니다.

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.

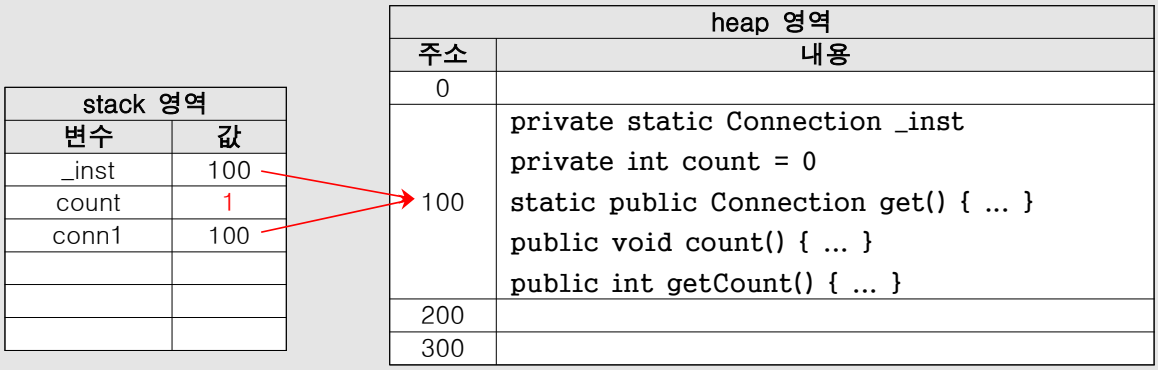


stack 영역	
변수	값
_inst	100
count	0
conn1	100

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.

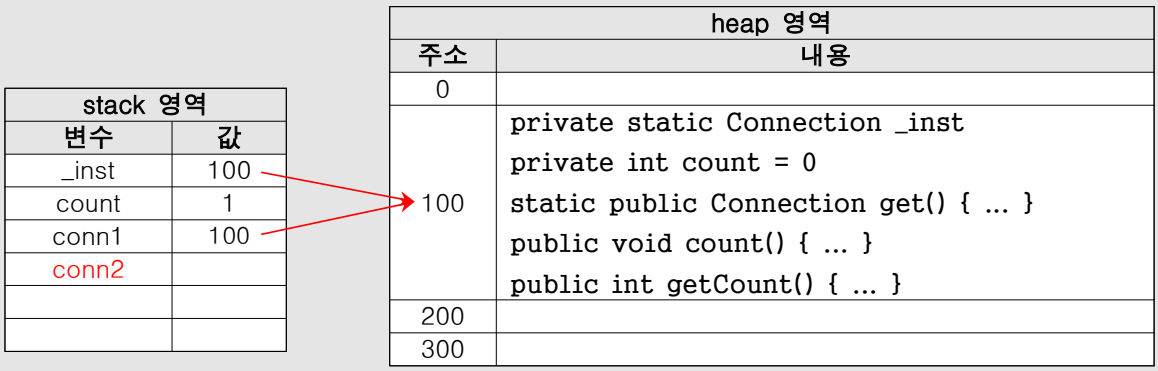
❽ 반환값이 없는 count() 메소드의 시작점이다. count의 값에 1을 더한다.



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	

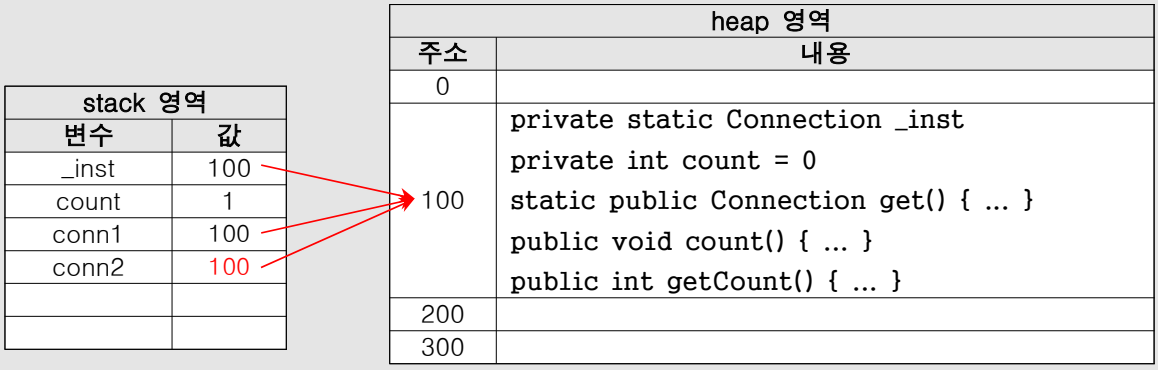
heap 영역	
주소	내용
0	
100	private static Connection _inst private int count = 0 static public Connection get() { ... } public void count() { ... } public int getCount() { ... }
200	
300	

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.

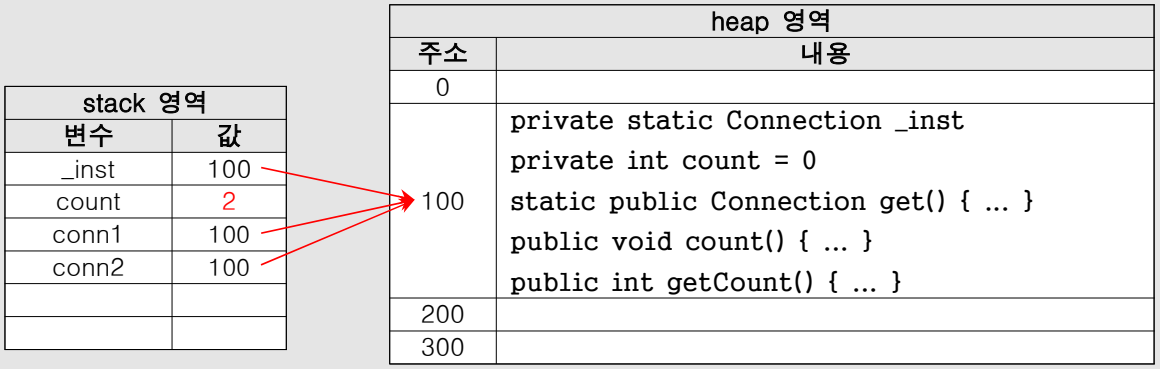


stack 영역	
변수	값
_inst	100
count	1
conn1	100
conn2	100

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.

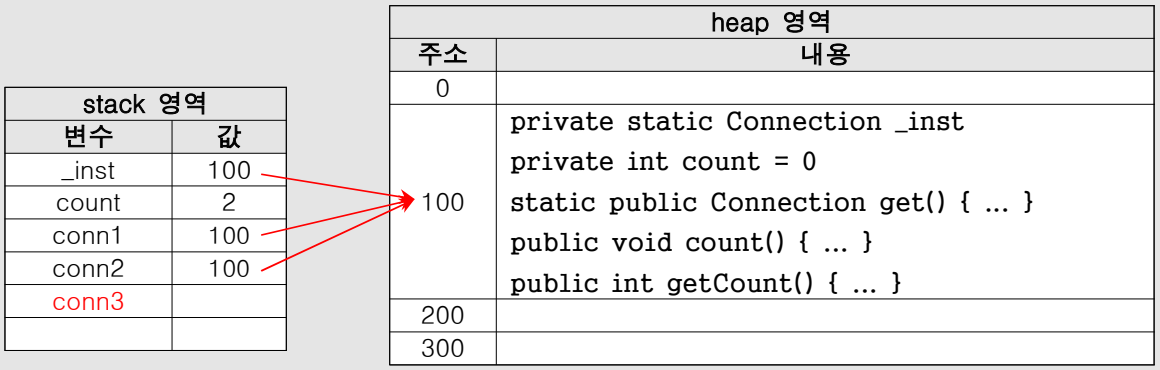
⓯ 반환값이 없는 count() 메소드의 시작점이다. count의 값에 1을 더한다.



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	

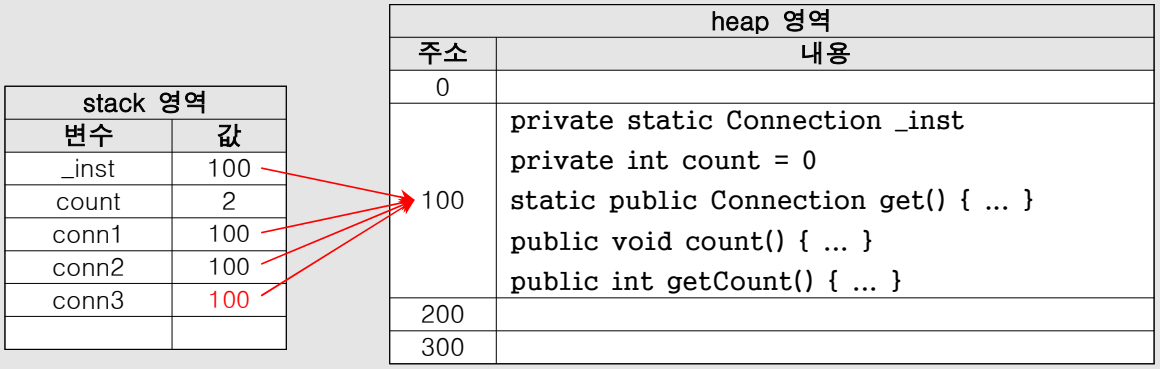
heap 영역	
주소	내용
0	
100	private static Connection _inst private int count = 0 static public Connection get() { ... } public void count() { ... } public int getCount() { ... }
200	
300	

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.

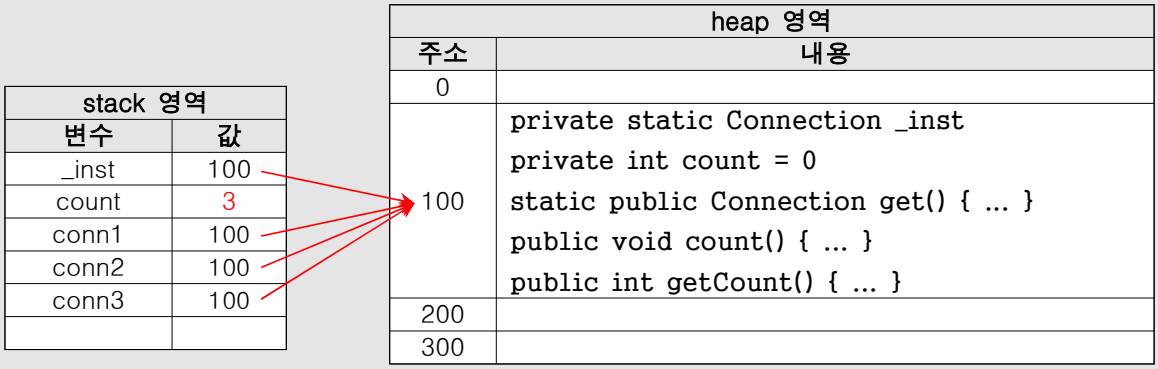


stack 영역	
변수	값
_inst	100
count	2
conn1	100
conn2	100
conn3	100

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	

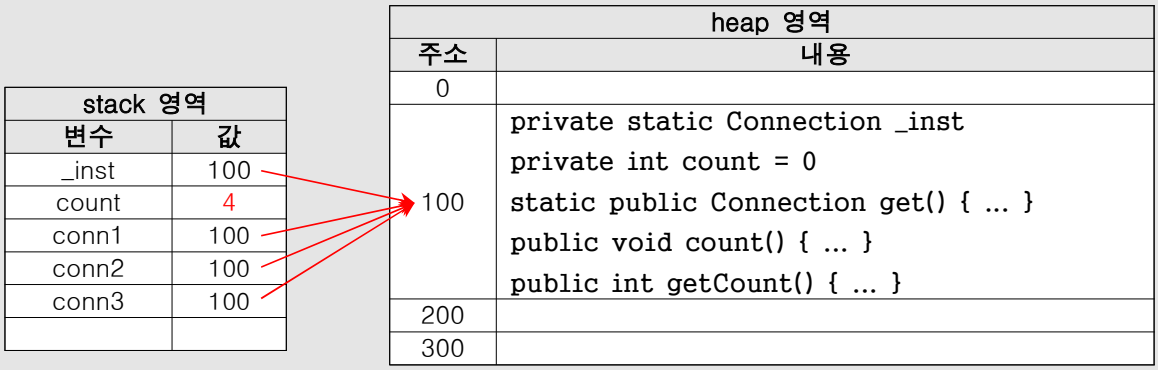
㉑ conn3 객체 변수의 count() 메소드를 호출한다.

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



㉓ conn1의 count() 메소드를 호출한다. conn1은 Connection() 클래스의 객체 변수이므로 Connection 클래스의 count() 메소드를 호출한다는 의미이다.

㉔ 반환값이 없는 count() 메소드의 시작점이다. count의 값에 1을 더한다.



㉕ conn1의 getCount() 메소드를 호출하고 돌려받은 값을 출력한다.

㉖ 정수를 반환하는 getCount() 메소드의 시작점이다. count의 값 4를 메소드를 호출했던 ㉗번으로 반환한다.

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

㉗ 화면에 4를 출력한다.

결과 4

[문제 6]

Abstract Factory

[문제 7]

① 6

② 6

[해설]

① LRU(Least Recently Used) 기법

3개의 페이지를 수용할 수 있는 주기억장치이므로 아래 그림과 같이 3개의 페이지 프레임으로 표현할 수 있습니다.

참조 페이지	1	2	3	1	2	4	5	1
페이지 프레임	**1**	1	1	1	1	1	**5**	5
		2	2	2	2	2	2	**1**
			3	3	3	**4**	4	4
부재 발생	●	●	●			●	●	●

참조 페이지가 페이지 테이블에 없으면 페이지 결함(부재)이 발생합니다. 초기에는 모든 페이지가 비어 있으므로

처음 1, 2, 3페이지 적재 시 페이지 결함이 발생합니다. 다음 참조 페이지 1, 2는 이미 적재되어 있으므로 그냥 참조합니다. LRU 기법은 최근에 가장 오랫동안 사용되지 않은 페이지를 교체하는 기법이므로, 참조 페이지 4를 적재할 때 3을 제거한 후 4를 가져옵니다. 이러한 과정으로 모든 페이지에 대한 요구를 처리하고 나면 총 페이지 결함의 발생 횟수는 6회입니다.

② LFU(Least Frequently Used) 기법

3개의 페이지를 수용할 수 있는 주기억장치이므로 아래 그림과 같이 3개의 페이지 프레임으로 표현할 수 있습니다.

참조 페이지	1	2	3	1	2	4	1	2	3	4
페이지 프레임	1	1	1	1	1	1	1	1	1	1
		2	2	2	2	2	2	2	2	2
			3	3	3	4	4	4	3	4
부재 발생	●	●	●			●			●	●

참조 페이지가 페이지 테이블에 없으면 페이지 결함(부재)이 발생합니다. 초기에는 모든 페이지가 비어 있으므로 처음 1, 2, 3페이지 적재 시 페이지 결함이 발생합니다. 다음 참조 페이지 1, 2는 이미 적재되어 있으므로 그냥 참조합니다. LFU 기법은 사용 빈도가 가장 적은 페이지를 교체하는 기법이므로, 참조 페이지 4를 적재할 때 3을 제거한 후 4를 가져옵니다. 마지막 3, 4페이지를 적재할 때도 사용 빈도가 적은 4와 3을 제거한 후 가져옵니다. 이러한 과정으로 모든 페이지에 대한 요구를 처리하고 나면 총 페이지 결함의 발생 횟수는 6회입니다.

[문제 8]

Seynaau

[답안작성 시 주의 사항]

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

[해설]

```
❶ a = [ 'Seoul', 'Kyeonggi', 'Incheon', 'Daejeon', 'Daegu', 'Pusan'];
❷ str01 = 'S'
❸ for i in a:
❹     str01 = str01 + i[1]
❺ print(str01)
```

❶ 리스트 a를 선언하면서 초기값을 지정한다. 초기값으로 지정된 수만큼 리스트의 요소가 만들어진다.

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
a	Seoul	Kyeonggi	Incheon	Daejeon	Daegu	Pusan

❷ 변수 str01을 선언하면서 초기값으로 문자 'S'를 지정한다.

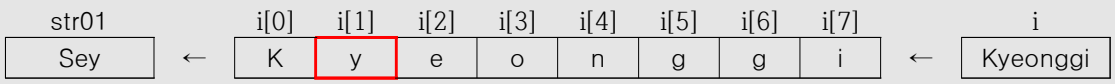
❸ 리스트 a의 요소 수만큼 ❹번 문장을 반복 수행한다. 리스트 a는 6개의 요소를 가지므로 각 요소를 i에 할당하면서 다음 문장을 6회 수행한다.

❹ str01과 i에 저장된 문자열의 두 번째 글자(i[1])를 더하여 str01에 저장한다. 즉 str01에 저장된 문자 뒤에 i에 저장된 문자열의 두 번째 글자가 덧붙여진다.

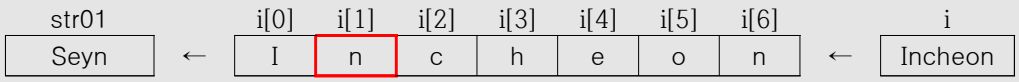
- 1회 : i에 a[0]이 저장되고 i의 1번째 글자 e가 str01에 더해진다.

str01		i[0]	i[1]	i[2]	i[3]	i[4]		i
Se	←	S	e	o	u	l	←	Seoul

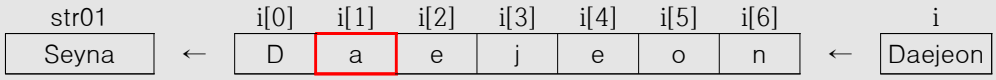
- 2회 : i에 a[1]이 저장되고 i의 1번째 글자 y가 str01에 더해진다.



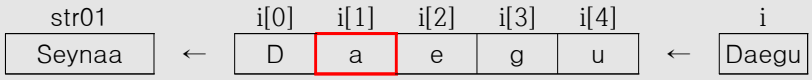
• 3회 : i에 a[2]가 저장되고 i의 1번째 글자 n이 str01에 더해진다.



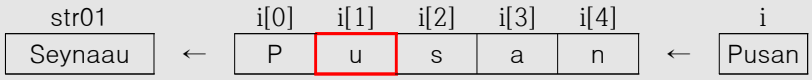
• 4회 : i에 a[3]이 저장되고 i의 1번째 글자 a가 str01에 더해진다.



• 5회 : i에 a[4]가 저장되고 i의 1번째 글자 a가 str01에 더해진다.



• 6회 : i에 a[5]가 저장되고 i의 1번째 글자 u가 str01에 더해진다.



❺ 결과 **Seynaau**

[문제 9]

① 세타 조인

② 동등 조인

③ 자연 조인

[문제 10]

GECA

[답안작성 시 주의 사항]

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

[해설]

```
#include <stdio.h>
#include <string.h>

❹ void inverse(char *str, int len) {
❺       for(int i = 0, j = len - 1; i < j; i++, j--) {
❻               char ch = str[i];
❼               str[i] = str[j];
❽               str[j] = ch;
        }
}❾

int main() {
❶       char str[100] = "ABCDEFGH";
❷       int len = strlen(str);
❸       inverse(str, len);
❿       for(int i = 1; i < len; i += 2) {
⓫               printf("%c", str[i]);
        }
⓬       return 0;
}
```

모든 C 언어 프로그램 반드시 main() 함수에서 시작한다.

❶ 100개의 요소를 갖는 문자형 배열 str을 선언하고, “ABCDEFGH”로 초기화한다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
str	‘A’	‘B’	‘C’	‘D’	‘E’	‘F’	‘G’	‘H’

❷ 정수형 변수 len을 선언하고, str의 길이인 8로 초기화한다.
- **strlen()** : 문자열의 길이를 반환한다.

❸ str과 len을 인수로 하여 inverse() 함수를 호출한다. 인수로 배열의 이름을 지정하면 배열의 시작 주소가 인수로 전달된다.

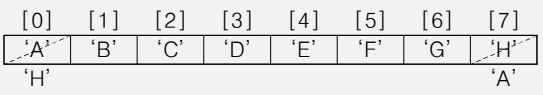
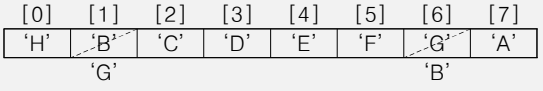
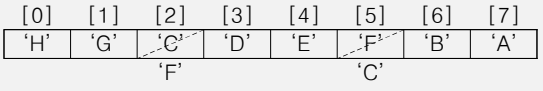
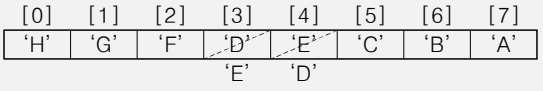
❹ 반환값이 없는 inverse 함수의 시작점이다. 문자형 포인터 변수 str은 str 배열의 시작 주소를 받고, 정수형 변수 len은 8을 받는다.

❺ 반복 변수 i는 0에서 시작하여 1씩 증가하고, 반복 변수 j는 len-1, 즉 7에서 시작하여 -1씩 증가하면서, i가 j보다 작은 동안 ❻~❽번을 반복 수행한다.

❻ 문자형 변수 ch에 str[i]의 값을 저장한다.

❼ str[i]에 str[j]의 값을 저장한다.

❽ str[j]에 ch의 값을 저장한다. ❻~❽은 str[i]와 str[j]의 값을 교환하는 과정이다.
반복문 실행에 따른 변수들의 변화는 다음과 같다.

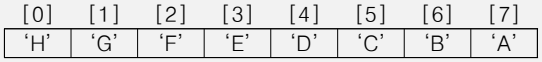
i	j	str[i]	str[j]	ch	배열 str
0	7	'A' 'H'	'H' 'A'	'A'	
1	6	'B' 'G'	'G' 'B'	'B'	
2	5	'C' 'F'	'F' 'C'	'C'	
3	4	'D' 'E'	'E' 'D'	'D'	
4	5				

❾ 함수를 마치고 inverse(str, len) 함수를 호출했던 main() 함수로 제어를 옮긴다. str 배열의 주소를 받아서 처리했으므로 main() 함수에 있는 str 배열에 자리 바꿈의 결과가 그대로 반영되어 있다.

❿ 반복 변수 i가 1에서 시작하여 2씩 증가하면서, len보다 작은 동안, 즉 8보다 작은 동안 ⓫번을 반복 수행한다.

⓫ str[i]의 값을 문자형으로 출력한다.

반복문의 실행에 따른 변수들의 변화는 다음과 같다.

i	str[i]	배열 str	출력
1	'G'		G
3	'E'		GE
5	'C'		GEC
7	'A'		GECA
9			

⓬ main() 함수에서의 'return 0'은 프로그램의 종료를 의미한다.

[문제 11]

⑤, ⑥, ③, ①, ⑦, ②

[해설]

```
    class Parent {
        int x, y;

①   ❺  Parent(int x, int y) {
    ❻      this.x = x;
    ❼      this.y = y;
        }

②   ❿  int getX() {
    ⓫      return x*y;
        }
    }

    class Child extends Parent {
        int x;

③   ❸  Child(int x) {
    ❹      super(x+1, x);
    ❽      this.x = x;
        }

④       int getX(int n) {
            return super.getX() + n;
        }
    }

    public class Main {
⑤   ❶  public static void main(String[] args) {
⑥   ❷      Parent parent = new Child(10);
⑦   ❾⓬     System.out.println(parent.getX());
        }
    }
```

❶ 모든 Java 프로그램은 반드시 main() 메소드에서 시작한다.

❷ Child 클래스의 생성자를 이용하여 Parent 클래스의 객체 변수 parent를 선언한다. 10을 인수로 하여 Child 클래스의 생성자를 호출한다.

- **[부모클래스명] [객체변수명] = new [자식클래스생성자( )]** : 부모 클래스의 객체 변수를 선언하면서 자식 클래스의 생성자를 사용하면 형 변환이 발생한다.
- 이렇게 형 변환이 발생했을 때 부모 클래스와 자식 클래스에 동일한 속성이나 메소드가 있으면 자식 클래스의 속성이나 메소드로 재정의된다.

❸ Child 클래스의 생성자인 Child() 메소드의 시작점이다. ❷번에서 전달한 10을 x가 받는다.

<table>
<tr><td colspan="2">객체 변수 parent</td></tr>
<tr><td colspan="2">Child</td></tr>
<tr><td rowspan="2">x</td><td>Child(int x)</td></tr>
<tr><td>x</td></tr>
<tr><td></td><td>10</td></tr>
</table>

❹ 부모 클래스의 생성자를 호출하며, 인수로 x+1과 x의 값을 전달한다.

- **super** : 상속한 부모 클래스를 가리키는 예약어

❺ 부모 클래스인 Parent 클래스의 생성자 Parent()의 시작점이다. ❹번에서 전달한 x+1의 값 11을 x가 받고,

x의 값 10을 y가 받는다.

객체 변수 parent					
Parent				Child	
x	y	Parent(int x, int y)		x	Child(int x)
		x	y		x
		11	10		10

❻ 'Parent.x = x;'와 동일하다. Parent.x에 x의 값 11을 저장한다.
- **this** : 현재 실행중인 메소드가 속한 클래스를 가리키는 예약어

❼ Parent.y에 y의 값 10을 저장한다.
생성자가 종료되면 생성자를 호출했던 ❹번의 다음 줄인 ❽번으로 이동한다.

객체 변수 parent					
Parent				Child	
x	y	Parent(int x, int y)		x	Child(int x)
		x	y		x
11	10	11	10		10

❽ 'Child.x = x;'와 동일하다. Child.x에 x의 값 10을 저장한다. 생성자가 종료되면 생성자를 호출했던 ❷번의 다음 줄인 ❾번으로 이동한다.

객체 변수 parent					
Parent				Child	
x	y	Parent(int x, int y)		x	Child(int x)
		x	y		x
11	10	11	10	10	10

❾ parent.getX() 메소드를 호출하여 반환받은 값을 출력한다.
❷번에서 형 변환이 발생했으므로 부모 클래스와 자식 클래스에 동일한 속성이나 메소드가 있으면 자식 클래스의 속성이나 메소드로 재정의 되지만 Parent 클래스의 int getX() 메소드와 Child 클래스의 int getX(int n) 메소드는 같은 메소드가 아니다. 이름이 같아도 인수가 다르면 같은 메소드가 아니다. ❿번을 호출한다.

❿ 정수를 반환하는 getX() 메소드의 시작점이다.

⓫ x*y의 값 110을 함수를 호출했던 ⓬번으로 반환한다.

객체 변수 parent					
Parent				Child	
x	y	Parent(int x, int y)		x	Child(int x)
		x	y		x
11	10	11	10	10	10

⓬ ⓫번으로부터 반환받은 값 **110**을 출력한다.

결과 110

[문제 12]

B
a
b

[해설]

```
SELECT B                 'B' 속성을 표시한다.
FROM R1                  <R1> 테이블을 대상으로 검색한다.
WHERE C IN (             <R1> 테이블의 'C'가 IN 다음에 쓰인 하위 질의의 결과와 같은 자료만을 대상으로 한다.
        SELECT C         'C' 속성을 표시한다.
        FROM R2          <R2> 테이블을 대상으로 검색한다.
        WHERE D='k');    'D'가 "k"인 자료만을 대상으로 한다.
```

[문제 13]

변환된 문자열 : Nd sc 1

[답안작성 시 주의 사항]

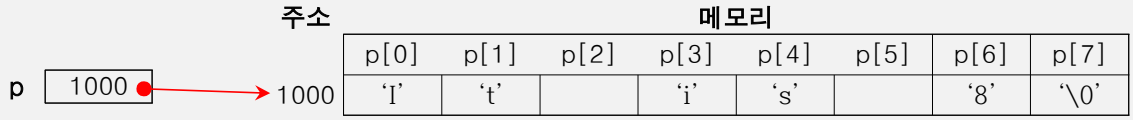
프로그램의 실행 결과는 부분 점수가 없으므로 정확하게 작성해야 합니다. 예를 들어, "변환된 문자열 : " 부분을 제외하고 **Nd sc 1**만 썼을 경우 부분 점수 없이 완전히 틀린 것으로 간주합니다.

[해설]

```
#include <stdio.h>
#include <ctype.h>

int main( ) {
❶       char *p = "It is 8";
❷       char result[100];
❸       int i;
❹       for(i = 0; p[i] != '\0'; i++) {
❺               if(isupper(p[i]))
❻                       result[i] = (p[i] - 'A'+ 5) % 25 + 'A';
❼               else if(islower(p[i]))
❽                       result[i] = (p[i] - 'a'+ 10) % 26 + 'a';
❾               else if(isdigit(p[i]))
❿                       result[i] = (p[i] - '0'+ 3) % 10 + '0';
⓫               else if(!(isupper(p[i]) || islower(p[i]) || isdigit(p[i])))
⓬                       result[i] = p[i];
        }
⓭       result[i] = '\0';
⓮       printf("변환된 문자열 : %s₩n", result);
⓯       return 0;
}
```

❶ 문자형 포인터 변수 p를 선언하고, 문자열 "It is 8"이 저장된 곳의 주소를 저장한다. (다음 그림에서 지정한 주소는 임의로 정한 것이며, 이해를 돕기 위해 10진수로 표현했음)



❷ 100개의 요소를 갖는 문자형 배열 result를 선언한다.

❸ 정수형 변수 i를 선언한다.

❹ 반복 변수 i가 0에서 시작하여 1씩 증가하면서 p[i]가 널 문자('\0')가 아닌 동안 ❺~⓬번을 반복 수행한다.

❺ p[i]의 값이 대문자이면 ❻번을 수행하고, 아니면 ❼번을 수행한다.

- **isupper()** : 인수가 대문자이면 참을 반환하는 함수

❻ (p[i] - 'A' + 5) % 25 + 'A'의 결과를 result[i]에 저장한 후 반복문의 처음인 ❹번으로 이동한다.

※ “It is 8”이라는 문자열이 메모리에 저장될 때는 각각의 문자에 해당하는 아스키코드 값이 저장되며, 계산식에 사용되는 문자 ‘A’도 아스키코드 값으로 변환되어 계산됩니다. 즉 ‘A’는 ‘A’에 해당하는 아스키코드 값인 65로 처리됩니다. 알파벳 대문자의 아스키코드 값은 ‘A(65)’ ~ ‘Z(90)’이며, 알파벳 소문자의 아스키코드 값은 ‘a(97)’ ~ ‘z(122)’입니다.

※ i가 0일 때, p[i]에 “I”가 저장되어 있으므로, (‘I’ - ‘A’ + 5) % 25 + ‘A’, 즉 (73 - 65 + 5) % 25 + 65의 결과인 78이 result[0]에 저장됩니다.

❼ p[i]의 값이 소문자이면 ❽번을 수행하고, 아니면 ❾번을 수행한다.

• **islower()** : 인수가 소문자이면 참을 반환하는 함수

❽ (p[i] - ‘a’ + 10) % 26 + ‘a’의 결과를 result[i]에 저장한 후 반복문의 처음인 ❹번으로 이동한다.

※ i가 1일 때, p[i]에 “t”가 저장되어 있으므로, (‘t’ - ‘a’ + 10) % 26 + ‘a’, 즉 (116 - 97 + 10) % 26 + 97의 결과인 100이 result[1]에 저장됩니다.

❾ p[i]의 값이 숫자이면 ❿번을 수행하고, 아니면 ⓫번을 수행한다.

• **isdigit()** : 인수가 숫자이면 참을 반환하는 함수

❿ (p[i] - ‘0’ + 3) % 10 + ‘0’의 결과를 result[i]에 저장한 후 반복문의 처음인 ❹번으로 이동한다.

※ i가 6일 때, p[i]에 “8”이 저장되어 있으므로, (‘8’ - ‘0’ + 3) % 10 + ‘0’, 즉 (8 - 0 + 3) % 10 + 0의 결과인 1이 result[6]에 저장됩니다.

⓫ p[i]의 값이 대문자 또는 소문자 또는 숫자가 아니면 ⓬번을 수행하고, 아니면 반복문의 처음인 ❹번으로 이동한다.

⓬ p[i]를 result[i]에 저장한 후 반복문의 처음인 ❹번으로 이동한다.

반복문 실행에 따른 변수들의 변화는 다음과 같다.

i	p[i]	result[i]	배열 result
0	‘I’	78	
1	‘t’	100	
2	‘ ’		
3	‘i’	115	
4	‘s’	99	
5	‘ ’		
6	‘8’	1	
7	‘\0’		

[0]	[1]	[2]	[3]	[4]	[5]	[6]	…
‘78’	‘100’	‘ ’	'115'	'99'	‘ ’	‘1’	…

⓭ result[i]에 널 문자(‘\0’)를 저장한다.

	p[0]	p[1]	p[2]	p[3]	p[4]	p[5]	p[6]	p[7]
result	‘78’	‘100’	‘ ’	'115'	'99'	‘ ’	‘1’	‘\0’

⓮ **변환된 문자열** : 을 출력한 후 result의 값을 문자열로 출력한다. 배열 result에 저장된 아스키코드에 해당하는 문자들이 출력됩니다.

결과 1

⓯ main() 함수에서의 ‘return 0’은 프로그램의 종료를 의미한다.

[문제 14]

MC/DC

[문제 15]
Rootkit

[문제 16]
APT

[해설]
APT(Advanced Persistent Threat), 지능형 지속공격

[문제 17]
1

[해설]
SQL도 프로그래밍 언어와 마찬가지로 OR 연산자보다 AND 연산자의 우선순위가 높다. 즉 '식1 AND 식2 OR 식3'과 같이 조건이 제시되었으면 '식1 AND 식2'의 조건을 먼저 확인한 후 그 결과와 식3의 OR 조건을 확인해야 한다.

```
SELECT COUNT(*)          튜플의 개수를 표시한다.
FROM EMP_TBL             <EMP_TBL> 테이블에서 검색한다.
WHERE EMPNO > 100        'EMPNO'가 100보다 크고
   AND SAL >= 3000       'SAL'이 3000 이상이거나,
    OR EMPNO = 200;      'EMPNO'가 200인 튜플만을 대상으로 검색한다.
```

[과정]
① 'EMPNO'가 100보다 큰 튜플은 다음과 같다.

EMPNO	SAL
200	3000
300	2000

② 'SAL'이 3000 이상인 튜플은 다음과 같다.

EMPNO	SAL
200	3000

③ ①, ②의 조건을 동시에 만족(AND)하는 튜플은 다음과 같다.

EMPNO	SAL
200	3000

④ 'EMPNO'가 200인 튜플은 다음과 같다.

EMPNO	SAL
200	3000

⑤ ③번 또는 ④번의 튜플 중 한 번이라도 포함된(OR) 튜플은 다음과 같다.

EMPNO	SAL
200	3000

⑥ COUNT(*) 함수에 따라 ⑤번 튜플의 개수를 표시하면 다음과 같다.

COUNT(*)
1

[문제 18]

9

[해설]

```
class firstArea {
        int x, y;
❹       public firstArea(int x, int y) {
❺               this.x = x;
❻               this.y = y;
        }
        public void print() {
                System.out.println(x+y);
        }
}

class secondArea extends firstArea {
        int bb = 3;
❷       public secondArea(int i) {
❸               super(i, i+1);
❼       }
❾       public void print() {
❿               System.out.println(bb*bb);
⓫       }
}

public class Main {
        public static void main(String[] args) {
❶               firstArea st = new secondArea(10);
❽               st.print();
⓬       }
}
```

모든 Java 프로그램은 반드시 main() 메소드에서 시작한다.

❶ secondArea 클래스의 생성자를 이용하여 firstArea 클래스의 객체 변수 st를 선언한다.

- **[부모클래스명] [객체변수명] = new [자식클래스생성자( )]** : 부모 클래스의 객체 변수를 선언하면서 자식 클래스의 생성자를 사용하면 형 변환이 발생한다.
- 이렇게 형 변환이 발생했을 때 부모 클래스와 자식 클래스에 동일한 속성이나 메소드가 있으면 자식 클래스의 속성이나 메소드로 재정의된다.

<table>
<tr><td colspan="3">객체 변수 st</td></tr>
<tr><td colspan="3">secondArea</td></tr>
<tr><td rowspan="2">bb</td><td>secondArea(int i)</td><td>print()</td></tr>
<tr><td>i</td><td></td></tr>
<tr><td>3</td><td></td><td></td></tr>
</table>

❷ secondArea 클래스의 생성자인 secondArea() 메소드의 시작점이다. ❶번에서 전달한 10을 i가 받는다.

<table>
<tr><td colspan="3">객체 변수 st</td></tr>
<tr><td colspan="3">secondArea</td></tr>
<tr><td rowspan="2">bb</td><td>secondArea(int i)</td><td>print()</td></tr>
<tr><td>i</td><td></td></tr>
<tr><td>3</td><td>10</td><td></td></tr>
</table>

❸ 부모 클래스의 생성자를 호출하며, 인수로 i와 i+1의 값을 전달한다.

ㆍ**super** : 상속한 부모 클래스를 가리키는 예약어

❹ 부모 클래스인 firstArea 클래스의 생성자 firstArea()의 시작점이다. ❸번에서 전달한 i의 값 10을 x가 받고, i+1의 값 11을 y가 받는다.

객체 변수 st							
firstArea					secondArea		
x	y	firstArea(int x, int y)		print()	bb	secondArea(int i)	print()
		x	y			i	
		10	11		3	10	

❺ 'fistArea.x = x;'와 동일하다. firstarea.x에 x의 값 10을 저장한다.

ㆍ**this** : 현재 실행중인 메소드가 속한 클래스를 가리키는 예약어

❻ 'fistArea.y = y;'와 동일하다. firstarea.y에 y의 값 11을 저장한다. 생성자가 종료되면 생성자를 호출했던 ❸번의 다음 줄인 ❼번으로 이동한다.

객체 변수 st							
firstArea					secondArea		
x	y	firstArea(int x, int y)		print()	bb	secondArea(int i)	print()
		x	y			i	
10	11	10	11		3	10	

❼ secondArea() 메소드가 종료되었으므로 호출했던 ❶번의 다음 줄인 ❽번으로 이동한다.

❽ st의 print() 메소드를 호출한다. ❾번으로 이동한다.

print() 메소드는 st 객체의 자료형이 firstArea이므로 firstArea 클래스의 print() 메소드라고 생각할 수 있지만 ❶번에서 클래스 형 변환이 발생하였고, print() 메소드가 자식 클래스에서 재정의되었으므로 secondArea 클래스의 print() 메소드가 수행된다.

❾ 반환값이 없는 print() 메소드의 시작점이다.

❿ bb*bb의 값 9를 출력한다.

결과 9

객체 변수 st							
firstArea					secondArea		
x	y	firstArea(int x, int y)		print()	bb	secondArea(int i)	print()
		x	y			i	bb*bb
10	11	10	11		3	10	9

⓫ print() 메소드가 종료되었으므로 호출했던 ❽번의 다음 줄인 ⓬번으로 이동하여 프로그램을 종료한다.