기출문제
2023년 2회 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 실행시킨 결과가 “43215”일 때, <처리조건>을 참고하여 괄호에 들어갈 알맞은 식을 쓰시오. (5점)

```
#include <stdio.h>
main() {
    int n[] = { 5, 4, 3, 2, 1 };
    for (int i = 0; i < 5; i++)
        printf("%d", (          ) );
}
```

<처리조건>

괄호의 식에 사용할 문자는 다음으로 제한한다.

- n, i
- +, −, /, *, %
- 0~9, (,), [,]

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 2 다음 C 언어로 구현된 프로그램과 <처리조건>을 참고하여 괄호(①~④)에 들어갈 알맞은 식을 쓰시오. (5점)

```
#include <stdio.h>
main() {
        int m = 4620;
        int a = (    ①    );
        int b = (    ②    );
        int c = (    ③    );
        int d = (    ④    );
        printf("1000원의 개수 : %d₩n", a);
        printf("500원의 개수 : %d₩n", b);
        printf("100원의 개수 : %d₩n", c);
        printf("10원의 개수 : %d₩n", d);
}
```

<처리조건>

괄호(①~④)의 식에 사용할 문자는 다음으로 제한한다.

- a, b, c, d, m, i, d
- +, −, /, *, %
- 0~9, (,)

답

- ①
- ②
- ③
- ④

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 3 다음 C 언어로 구현된 프로그램을 분석하여 “홍길동”, “김철수”, “박영희”를 차례로 입력했을 때 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
#include <stdio.h>
char n[30];
char* getname() {
        printf("이름 입력 : ");
        gets(n);
        return n;
}

main() {
        char* n1 = getname();
        char* n2 = getname();
        char* n3 = getname();
        printf("%s₩n", n1);
        printf("%s₩n", n2);
        printf("%s₩n", n3);
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 4 다음 <학생> 테이블에 (9816021, '한국산', 3, '경영학개론', '050-1234-1234')인 데이터를 삽입하고자 한다. <처리조건>을 참고하여 적합한 SQL문을 작성하시오. (5점)

<학생>

학번	이름	학년	신청과목	연락처
9815932	김태산	3	경영정보시스템	050-5234-1894
9914511	박명록	2	경제학개론	050-1415-4986
0014652	이익명	1	국제경영	050-6841-6781
9916425	김혜리	2	재무관리	050-4811-1187
9815945	이지영	3	인적자원관리	050-9785-8845

<처리조건>

- 최소한의 코드로 작성될 수 있도록 SQL문을 구성한다.
- 명령문 마지막의 세미콜론(;)은 생략이 가능하다.
- 인용 부호가 필요한 경우 작은따옴표(' ')를 사용한다.

답 :

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

```
#include <stdio.h>
main() {
        int n[] = { 73, 95, 82 };
        int sum = 0;
        for (int i = 0; i < 3; i++)
                sum += n[i];

        switch (sum / 30) {
        case 10:
        case 9: printf("A");
        case 8: printf("B");
        case 7:
        case 6: printf("C");
        default: printf("D");
        }
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 6 화이트박스 테스트의 검증 기준에 대한 다음 설명에 해당하는 용어를 <보기>에서 찾아 쓰시오. (5점)

테스트 케이스를 소스 코드의 조건문에 포함된 개별 조건식의 결과가 True인 경우와 False인 경우가 한 번 이상 수행되도록 설계한다.

<보기>

• 문장 커버리지	• 분기 커버리지	• 조건 커버리지	• 분기/조건 커버리지

답 :

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

```
#include <stdio.h>
main() {
        int c = 0;
        for (int i = 1; i <= 2023; i++)
                if (i % 4 == 0)
                        c++;
        printf("%d", c);
}
```

답 :

문제 8 소프트웨어 데이터의 비정상적인 수정이 감지되면 소프트웨어를 오작동하게 만들어 악용을 방지하는 기술이다. 해시 함수, 핑거 프린트, 워터마킹 등의 보안 요소를 생성하여 소프트웨어에 삽입하고, 실행코드를 난독화하며, 실행 시 원본 비교 및 데이터 확인을 수행함으로써 소프트웨어를 보호하는 이 기술을 가리키는 용어를 쓰시오.

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

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

```
#include <stdio.h>
#define MAX_SIZE 10

int isWhat[MAX_SIZE];
int point = -1;

int isEmpty() {
        if (point == -1) return 1;
        return 0;
}

int isFull() {
        if (point == 10) return 1;
        return 0;
}

void into(int num) {
        if (isFull() == 1) printf("Full");
        else isWhat[++point] = num;
}

int take() {
        if (isEmpty() == 1) printf("Empty");
        else return isWhat[point--];
        return 0;
}

main() {
        into(5); into(2);
        while (!isEmpty()) {
                printf("%d", take());
                into(4); into(1); printf("%d", take());
                into(3); printf("%d", take()); printf("%d", take());
                into(6); printf("%d", take()); printf("%d", take());
        }
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 10 다음은 데이터베이스 구축까지의 과정을 나열한 것이다. 괄호(①~⑤)에 들어갈 알맞은 용어를 <보기>에서 찾아 쓰시오. (5점)

(①)	→	(②)	→	(③)	→	(④)	→	(⑤)

<보기>

• 개념적 설계	• 데이터베이스 구현	• 물리적 설계	• 요구 조건 분석
• 인터페이스 설계	• 논리 스키마 설계	• 논리적 설계	• 트랜잭션 작성

답

- ①
- ②
- ③
- ④
- ⑤

문제 11 디자인 패턴에 대한 다음 설명에서 괄호(①, ②)에 들어갈 알맞은 용어를 <보기>에서 찾아 쓰시오. (5점)

- (①) : 하나의 객체를 생성하면 생성된 객체를 어디서든 참조할 수 있지만, 여러 프로세스가 동시에 참조할 수 없는 패턴으로, 불필요한 메모리 낭비를 최소화 할 수 있음
- (②) : 각 클래스들의 데이터 구조에서 처리 기능을 분리하여 별도로 구성함으로써, 클래스를 수정하지 않고도 새로운 연산의 추가가 가능함

<보기>

생성 패턴	구조 패턴	행위 패턴
추상 팩토리(Abstract Factory) 프로토타입(Prototype) 싱글톤(Singleton)	어댑터(Adapter) 브리지(Bridge) 프록시(Proxy)	인터프리터(Interpreter) 중재자(Mediator) 옵서버(Observer) 방문자(Visitor)

답

- ①
- ②

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 12 **다음 설명에서 괄호(①~⑤)에 들어갈 알맞은 용어를 <보기>에서 찾아 쓰시오. (5점)**

전송 오류의 발생에는 감쇠, 지연 왜곡, 잡음 등 다양한 원인이 있으며, 이러한 오류를 검출하고 수정하는 것으로 알려진 대표적인 방식이 (①) 코드 방식이다.

(①) 코드 방식은 하나의 데이터 단위에 (④) 비트를 추가하여 오류를 검출하여 교정이 가능한 코드로, 2bit의 오류를 검출할 수 있으며 1bit의 오류를 교정한다. 데이터 비트 외에 잉여 비트가 많이 필요하다는 단점이 있다.

(①) 코드 방식은 수신측에서 오류를 정정하는 (②)에 해당한다. (②)는 데이터 전송 과정에서 오류가 발생하면 송신측에 재전송을 요구하는 (③)와는 달리 재전송 요구 없이 스스로 수정하기 때문에 연속적인 데이터 전송이 가능하다.

(③)는 (④) 검사, (⑤) 등을 통해 오류를 검출하고 ARQ(Automatic Repeat reQuest)로 오류를 제어한다.

(④) 검사는 오류 검사를 위해 데이터 비트 외에 1bit의 체크 비트를 추가하는 것으로 1bit의 오류만 검출할 수 있다. 1의 개수에 따라 짝수 (④)와 홀수 (④)로 나뉜다.

(⑤)는 다항식 코드를 사용하여 오류를 검출하는 방식이다. 동기식 전송에서 주로 사용되며, HDLC 프레임의 FCS(프레임 검사 순서 필드)에 사용되는 방식이다. 집단 오류를 검출할 수 있고, 검출률이 높으므로 가장 많이 사용한다.

<보기>

• NAK	• CRC	• FEC	• BCD
• Parity	• Hamming	• MD5	• BEC

답

- ①
- ②
- ③
- ④
- ⑤

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 13 HDLC(High-level Data Link Control)에 대한 다음 설명에서 괄호(①~⑤)에 들어갈 알맞은 용어를 <보기>에서 찾아 쓰시오. (5점)

HDLC는 비트(Bit) 위주의 프로토콜로, 각 프레임에 데이터 흐름을 제어하고 오류를 검출할 수 있는 비트열을 삽입하여 전송한다. 포인트 투 포인트(Point-to-Point) 및 멀티 포인트(Multi-Point), 루프(Loop) 등 다양한 데이터 링크 형태에 동일하게 적용이 가능하다는 특징이 있다.

HDLC의 프레임 구조는 헤더, 텍스트, 트레일러로 구분되며, 헤더는 다시 플래그, 주소부, 제어부로 구분할 수 있는데, 제어부에는 프레임의 종류를 식별하기 위해 사용한다. 제어부의 첫 번째, 두 번째 비트를 사용하여 (①) 프레임, (②) 프레임, (③) 프레임으로 구분한다.

(①) 프레임은 I 프레임으로 불리며, 제어부가 '0'으로 시작하는 프레임으로, 사용자 데이터를 전달하거나 피기백킹(Piggybacking) 기법을 통해 데이터에 대한 확인 응답을 보낼 때 사용된다.
(②) 프레임은 S 프레임으로 불리며, 제어부가 '10'으로 시작하는 프레임으로, 오류 제어와 흐름 제어를 위해 사용된다.
(③) 프레임은 U 프레임으로 불리며, 제어부가 '11'로 시작하는 프레임으로, 링크의 동작 모드 설정과 관리를 한다.

(③) 프레임에서 설정할 수 있는 동작 모드에는 표준 응답 모드, (④), (⑤)의 세 가지로 구분된다.

표준 응답 모드는 반이중 통신을 하는 포인트 투 포인트(Point-to-Point) 또는 멀티 포인트(Multi-Point) 불균형 링크 구성에 사용되며, 종국은 주국의 허가(Poll)가 있을 때에만 송신하는 특징이 있다.
(④)는 포인트 투 포인트(Point-to-Point) 균형 링크에서 사용되며, 혼합국끼리 허가 없이 언제나 전송할 수 있다.
(⑤)는 전이중 통신을 하는 포인트 투 포인트(Point-to-Point) 불균형 링크 구성에 사용되며, 종국은 주국의 허가(Poll) 없이도 송신이 가능하지만 링크 설정이나 오류 복구 등의 제어 기능은 주국만 가능하다.

<보기>

• 비동기 응답 모드	• 주소부	• 제어부	• ARQ
• 정보	• 비번호	• 감독	• 플래그
• 비동기 균형 모드			

답

- ①
- ②
- ③
- ④
- ⑤

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 14 다음 JAVA로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
public class Test {
        public static void main(String[] args) {
                String str1 = "Programming";
                String str2 = "Programming";
                String str3 = new String("Programming");
                System.out.println(str1==str2);
                System.out.println(str1==str3);
                System.out.println(str1.equals(str3));
                System.out.println(str2.equals(str3));
    }
}
```

답 :

문제 15 다음 <보기>에 나열된 암호화 알고리즘을 대칭키 암호화 알고리즘과 비대칭키 암호화 알고리즘으로 구분하시오. (5점)

<보기>

• RSA	• DES	• ARIA	• ECC	• SEED	• AES

답

• ① 대칭키 암호화 알고리즘 :

• ② 비대칭키 암호화 알고리즘 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 16 **암호화 알고리즘에 대한 다음 설명에서 괄호에 들어갈 알맞은 용어를 쓰시오. (5점)**

(　　)는 임의의 길이의 입력 데이터나 메시지를 고정된 길이의 값이나 키로 변환하는 알고리즘으로, 복호화가 거의 불가능한 일방향 함수이다. 무결성 검증을 위해 사용될 뿐만 아니라 정보보호의 다양한 분야에서 활용되며, 종류에는 SHA 시리즈, MD5, N-NASH, SNEFRU 등이 있다.

답 :

문제 17 **다음 <처리조건>에 부합하는 <SQL문>이 완성되도록 괄호에 적합한 옵션을 쓰시오. (5점)**

<처리조건>

• <학생> 뷰를 제거한다. • <학생> 뷰를 참조하는 모든 데이터도 연쇄적으로 제거한다.

<SQL문>

DROP VEW 학생 (　　);

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 18 다음은 데이터를 오름차순으로 정렬하는 선택정렬 알고리즘을 C 언어 프로그램으로 구현한 것이다. 프로그램을 분석하여 괄호에 들어갈 알맞은 연산자를 쓰시오. (5점)

```
#include <stdio.h>
main() {
        int E[] = { 64, 25, 12, 22, 11 };
        int n = sizeof(E) / sizeof(E[0]);
        int i = 0;
        do {
                int j = i + 1;
                do {
                        if (E[i] (    ) E[j]) {
                                int tmp = E[i];
                                E[i] = E[j];
                                E[j] = tmp;
                        }
                        j++;
                } while (j < n);
                i++;
        } while (i < n - 1);
        for (int i = 0; i <= 4; i++)
                printf("%d ", E[i]);
}
```

<처리조건>

괄호의 연산자는 다음으로 제한한다.

- +=, -=, *=, /=
- ==, !=, >, >= <, <=
- &&, ||

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

문제 19 다음 Python으로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
a = "engineer information programming"
b = a[:3]
c = a[4:6]
d = a[29:]
e = b + c + d
print(e)
```

답 :

문제 20 애플리케이션 테스트에 관한 다음 설명에서 괄호(①, ②)에 들어갈 알맞은 용어를 쓰시오. (5점)

하향식 통합 테스트는 프로그램의 상위 모듈에서 하위 모듈 방향으로 통합하면서 테스트하는 기법이다. 깊이 우선 통합법이나 넓이 우선 통합법을 사용하며, 주요 제어 모듈의 종속 모듈들을 (①)으로 대체한다는 특징이 있다.

상향식 통합 테스트는 프로그램의 하위 모듈에서 상위 모듈 방향으로 통합하면서 테스트하는 기법이다. 하위 모듈들을 클러스터(Cluster)로 결합하며, 상위 모듈에서 데이터의 입·출력을 확인하기 위해 더미 모듈인 (②)를 작성한다는 특징이 있다.

답

- ①
- ②

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

기출문제 정답 및 해설

[문제 1]

n[(i + 1) % 5]

[해설]

배열에 순서대로 저장된 숫자 5, 4, 3, 2, 1을 4, 3, 2, 1, 5로 출력하는 문제입니다.

배열 n

n[0]	n[1]	n[2]	n[3]	n[4]
5	4	3	2	1

→ 4, 3, 2, 1, 5

반복 변수 i를 이용하여 출력하는 데,

i가 0이면 1번째, 즉 n[1]의 값 4를 출력해야 하고,

i가 1이면 2번째, 즉 n[2]의 값 3을 출력해야 하고,

i가 2면 3번째, 즉 n[3]의 값 2를 출력해야 하고,

i가 3이면 4번째, 즉 n[4]의 값 1을 출력해야 하고,

i가 4면 0번째, 즉 n[0]의 값 5를 출력해야 합니다.

즉, 반복 변수 i를 배열의 첨자에 이용하려면 (i+1)%5와 같이 사용하면 됩니다.

i=0, 1%5 → 1

i=1, 2%5 → 2

i=2, 3%5 → 3

i=3, 4%5 → 4

i=4, 5%5 → 0

```
#include <stdio.h>
main() {
❶    int n[] = { 5, 4, 3, 2, 1 };
❷    for (int i = 0; i < 5; i++)
❸        printf("%d", n[(i + 1) % 5]);
}
```

❶ 5개의 요소를 갖는 정수형 배열 n을 선언하고 초기화한다.

n

[0]	[1]	[2]	[3]	[4]
5	4	3	2	1

❷ 반복 변수 i가 0부터 1씩 증가하면서 5보다 작은 동안 ❸번을 반복 수행한다.

❸ n[(i + 1) % 5]의 값을 정수로 출력한다. 반복문 실행에 따른 변수들의 변화는 다음과 같다.

i	(i+1)%5	출력
0	1	**4**
1	2	4**3**
2	3	43**2**
3	4	432**1**
4	0	4321**5**
5		

[문제 2]

① m / 1000

② m % 1000 / 500

③ m % 500 / 100

④ m % 100 / 10

[해설]

이 문제는 4620원에 포함된 1000원, 500원, 100원, 10원 단위의 개수를 구하는 문제입니다.

① 1000원 단위의 개수

4620/1000 → 4.62이지만 정수 나눗셈이므로 몫은 4, 즉 1000원 단위의 개수는 4입니다.

∴ m/1000

② 500원 단위의 개수

• 620원에 포함된 500원 단위의 개수를 구합니다.

• 4620%1000 → 620

• 620/500 → 1.24, 500원 단위의 개수는 1입니다.

∴ m%1000/500

③ 100원 단위의 개수

• 120원에 포함된 100원 단위의 개수를 구합니다.

• 4620%500 → 120

• 120/100 → 1.2, 100원 단위의 개수는 1입니다.

∴ m%500/100

④ 10원 단위의 개수

• 20원에 포함된 10원 단위의 개수를 구합니다.

• 4620%10 → 20

• 20/10 → 2.0, 10원 단위의 개수는 2입니다.

∴ m%100/10

```
#include <stdio.h>
main() {
❶       int m = 4620;
❷       int a = m / 1000;
❸       int b = m % 1000 / 500;
❹       int c = m % 500 / 100;
❺       int d = m % 100 / 10;
❻       printf("1000원의 개수 : %dWn", a);
❼       printf("500원의 개수 : %dWn", b);
❽       printf("100원의 개수 : %dWn", c);
❾       printf("10원의 개수 : %dWn", d);
}
```

❶ 정수형 변수 m을 선언하고 4620으로 초기화한다.

❷ 정수형 변수 a를 선언하고 'm / 1000'의 값 4로 초기화한다.

❸ 정수형 변수 b를 선언하고 'm % 1000 / 500'의 값 1로 초기화한다.

❹ 정수형 변수 c를 선언하고 'm % 500 / 100'의 값 1로 초기화한다.

❺ 정수형 변수 d를 선언하고 'm % 100 / 10'의 값 2로 초기화한다.

❻ 화면에 **1000원의 개수 :** 와 a의 값 **4**를 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과
```
1000원의 개수 : 4
```

❼ 화면에 **500원의 개수 :** 와 b의 값 **1**을 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과
```
1000원의 개수 : 4
500원의 개수 : 1
```

❽ 화면에 **100원의 개수 :** 와 c의 값 **1**을 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과
```
1000원의 개수 : 4
500원의 개수 : 1
100원의 개수 : 1
```

❾ 화면에 **10원의 개수 :** 와 d의 값 **2**를 출력하고 커서를 다음 줄의 처음으로 옮긴다.

결과
```
1000원의 개수 : 4
500원의 개수 : 1
100원의 개수 : 1
10원의 개수 : 2
```

[문제 3]

박영희

박영희

박영희

[해설]

```
#include <stdio.h>
char n[30];                          30개의 요소를 갖는 문자형 배열 n을 전역변수로 선언한다. 전역변수이기 때문에 이
                                     프로그램 안에서는 어디서든 사용할 수 있으며, 저장된 값이 유지된다.
❷❽⓮ char* getname() {
❸❾⓯   printf("이름 입력 : ");
❹❿⓰   gets(n);
❺⓫⓱   return n;
}

main() {
❶⓰     char* n1 = getname();
❼⓬     char* n2 = getname();
⓭⓲     char* n3 = getname();
⓳       printf("%sWn", n1);
⓴       printf("%sWn", n2);
㉑       printf("%sWn", n3);
}
```

모든 C언어 프로그램은 반드시 main( ) 함수에서 시작한다.

❶ 문자형 포인터 변수 n1을 선언하고 getname( ) 함수를 호출한 후 돌려받은 값으로 초기화한다.

❷ 문자형 포인터 값을 반환하는 getname( ) 함수의 시작점이다.

❸ 화면에 **이름 입력 :** 을 출력한다.

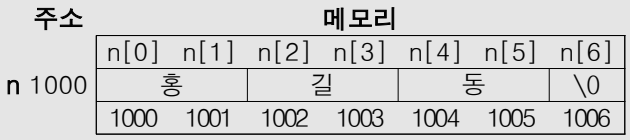
입력 화면 **이름 입력 :**

❹ 사용자로부터 문자열을 입력받아 n에 저장한다. 문제에서 처음에 **홍길동**을 입력한다고 하였으므로 n에는 **홍길동**이 저장된다.

입력 화면 **이름 입력 : 홍길동**

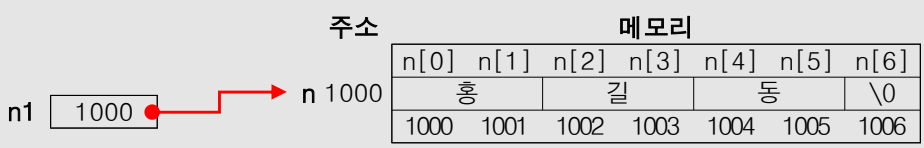
※ 다음 그림에서 배열 n의 주소는 임의로 정한 것이며, 이해를 돕기 위해 10진수로 표현했습니다.

※ 문자열을 저장하는 경우 문자열의 끝을 의미하는 널 문자(\0)가 추가로 저장되며, 출력 시 널 문자는 표시되지 않습니다.



❺ n의 시작 주소를 함수를 호출했던 ❶번으로 반환한다.

❻ ❺번에서 돌려받은 주소값을 n1에 저장한다.



❼ 문자형 포인터 변수 n2을 선언하고 getname( ) 함수를 호출한 후 돌려받은 값으로 초기화한다.

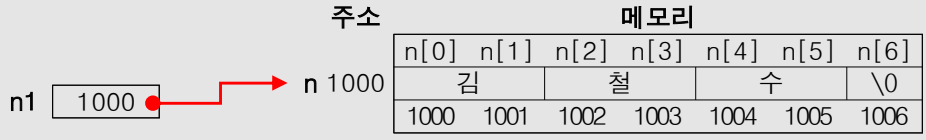
❽ 문자형 포인터 값을 반환하는 getname( ) 함수의 시작점이다.

❾ 화면에 **이름 입력 :** 을 출력한다.

입력 화면
```
이름 입력 : 홍길동
이름 입력 :
```

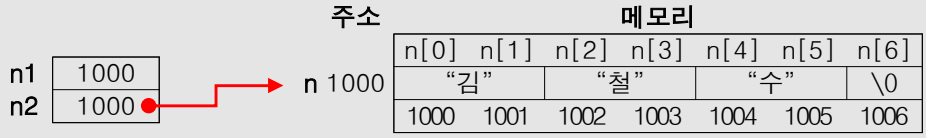
⑩ 사용자로부터 문자열을 입력받아 n에 저장한다. 문제에서 두 번째로 **김철수**를 입력한다고 하였으므로 n에는 **김철수**가 저장된다.

입력 화면
```
이름 입력 : 홍길동
이름 입력 : 김철수
```



⑪ n의 시작 주소를 함수를 호출했던 ❼번으로 반환한다.

⑫ ⑪번에서 돌려받은 주소값을 n2에 저장한다.



⑬ 문자형 포인터 변수 n3을 선언하고 getname( ) 함수를 호출한 후 돌려받은 값으로 초기화한다.

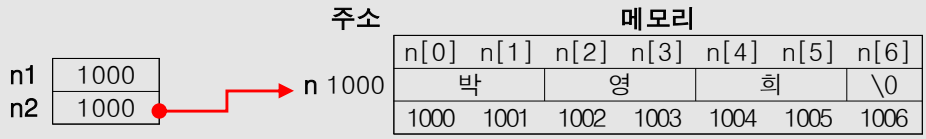
⑭ 문자형 포인터 값을 반환하는 getname( ) 함수의 시작점이다.

⑮ 화면에 **이름 입력 :** 을 출력한다.

입력 화면
```
이름 입력 : 홍길동
이름 입력 : 김철수
이름 입력 :
```

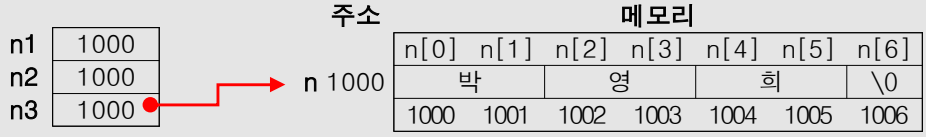
⑯ 사용자로부터 문자열을 입력받아 n에 저장한다. 문제에서 세 번째로 **박영희**를 입력한다고 하였으므로 n에는 **박영희**가 저장된다.

입력 화면
```
이름 입력 : 홍길동
이름 입력 : 김철수
이름 입력 : 박영희
```



⑰ n의 시작 주소를 함수를 호출했던 ⑬번으로 반환한다.

⑱ ⑰번에서 돌려받은 주소값을 n3에 저장한다.



⑲ n1이 가리키는 곳의 문자열 **박영희**를 출력한 후 커서를 다음 줄의 처음으로 옮긴다.

결과
```
박영희
```

⑳ n2가 가리키는 곳의 문자열 **박영희**를 출력한 후 커서를 다음 줄의 처음으로 옮긴다.

결과
```
박영희
박영희
```

㉑ n3이 가리키는 곳의 문자열 **박영희**를 출력한 후 커서를 다음 줄의 처음으로 옮긴다.

결과
```
박영희
박영희
박영희
```

[문제 4]

INSERT INTO 학생 VALUES(9816021, '한국산', 3, '경영학개론', '050-1234-1234');

[답안 작성 시 주의 사항]

SQL에 사용되는 예약어, 필드명, 변수명 등은 대소문자를 구분하지 않기 때문에 대문자로만 또는 소문자로만 작성해도 정답으로 인정됩니다.

[풀이]

```
INSERT INTO 학생                          <학생> 테이블에 삽입하라.
VALUES(9816021, '한국산', 3, '경영학개론', '050-1234-1234');
                                  첫 번째 필드부터 순서대로 9816021, '한국산', 3, '경영학개론',
                                  '050-1234-1234'를 삽입하라.
```

[문제 5]

BCD

[해설]

```
#include <stdio.h>
main() {
❶       int n[] = { 73, 95, 82 };
❷       int sum = 0;
❸       for (int i = 0; i < 3; i++)
❹               sum += n[i];

❺       switch (sum / 30) {
        case 10:
        case 9: printf("A");
❻       case 8: printf("B");
❼       case 7:
❽       case 6: printf("C");
❾       default: printf("D");
        } ❿
}
```

❶ 3개의 요소를 갖는 정수형 배열 n을 선언하고 초기화한다.

	[0]	[1]	[2]
n	73	95	82

❷ 정수형 변수 sum을 선언하고 0으로 초기화한다.

❸ 반복 변수 i가 0부터 1씩 증가하면서 3보다 작은 동안 ❹번을 반복 수행한다.

❹ sum에 n[i]의 값을 누적시킨다. 반복문 실행에 따른 변수들의 변화는 다음과 같다.

i	n[i]	sum
0	73	73
1	95	168
2	82	250
3		

❺ sum/30은 8.333이지만 정수형 연산이므로 8에 해당하는 case문으로 이동한다.

❻ 화면에 B를 출력한다.

결과 B

❼ 실행문이 없으므로 다음 줄로 이동한다.

❽ 화면에 C를 출력한다.

결과 BC

❾ 화면에 D를 출력한다. switch문이 종료되었으므로 ❿번으로 이동하여 프로그램을 종료한다.

결과 BCD

[문제 6]

조건 커버리지

[문제 7]

505

[해설]

```
#include <stdio.h>
main() {
❶       int c = 0;
❷       for (int i = 1; i <= 2023; i++)
❸               if (i % 4 == 0)
❹                       c++;
❺       printf("%d", c);
}
```

❶ 정수형 변수 c를 선언하고 0으로 초기화한다.

❷ 반복 변수 i가 1부터 1씩 증가하면서 2023보다 작거나 같은 동안 ❸번을 반복 수행한다.

❸ i를 4로 나눈 나머지가 0이면 ❹번을 수행한다. 2023안에 포함된 4의 배수를 세는 알고리즘이다.

❹ 'c = c + 1;'과 동일하다. c의 값을 1씩 누적시킨다. 반복문 실행에 따른 변수들의 변화는 다음과 같다.

i	c
1	0
2	
3	
4	1
5	
6	
7	
8	2
⋮	⋮
2020	505
2021	
2022	
2023	
2024	

❺ c의 값을 정수로 출력한다.

결과 505

[문제 8]

※ 다음 중 하나를 쓰면 됩니다.

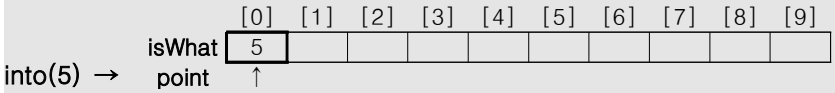
템퍼 프루핑, Tamper Proofing

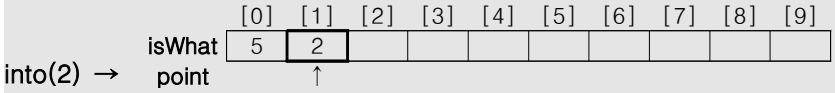
[문제 9]

213465

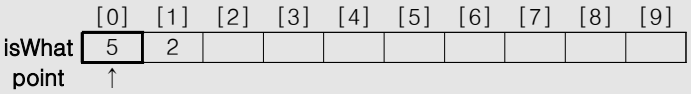
[해설]

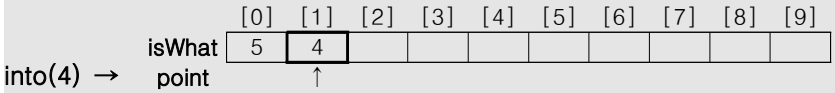
이 문제에서 into()는 isWhat 배열에 값을 저장하는 함수이고, take()는 isWhat 배열의 값을 출력하는 함수입니다. 코드가 실행되는 과정을 개괄적으로 살펴보면 다음과 같습니다.

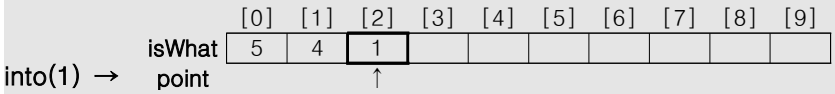




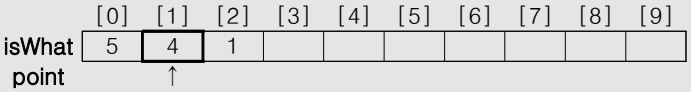
take() → 출력 **2**

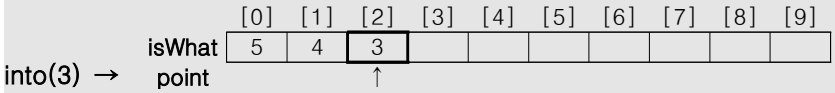




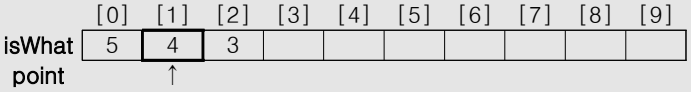


take() → 출력 **21**

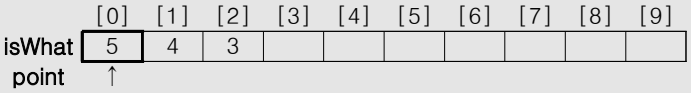


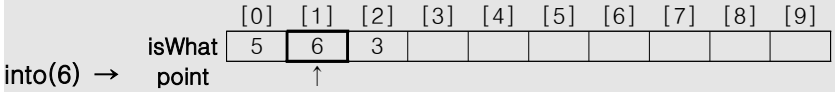


take() → 출력 **213**

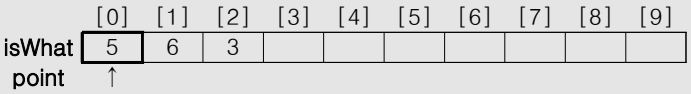


take() → 출력 **2134**

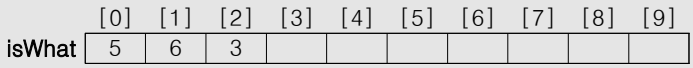




take() → 출력 **21346**

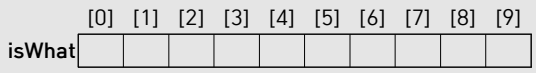


take() → 출력 **213465**



- point는 isWhat 배열에 저장될 값이나 배열에서 출력될 값의 위치를 지정하는 변수입니다. point가 -1이 되면, 즉 point가 isWhat 배열을 벗어나면 프로그램이 종료됩니다.
- into() 함수가 isWhat 배열에 값을 저장하기 전에 isFull() 함수가 호출되어 isWhat 배열을 검사합니다. isWhat 배열이 모두 채워져 있으면 화면에 **Full**을 출력합니다.
- take() 함수가 isWhat 배열의 값을 출력하기 전에 isEmpty() 함수가 호출되어 isWhat 배열을 검사합니다. isWhat 배열이 비어 있으면 화면에 **Empty**를 출력합니다.

```
#include <stdio.h>
#define MAX_SIZE 10          10을 MAX_SIZE로 정의한다. 프로그램 안에서 MAX_SIZE는 숫자 10과 동일하다.
int isWhat[MAX_SIZE];        10개의 요소를 갖는 배열 isWhat을 정수형 전역변수로 선언한다.
```




```
int point = -1;              point를 정수형 전역변수로 선언하고 -1로 초기화한다.

⓫⓲ int isEmpty() {
⓬⓳     if (point == -1) return 1;
⓭⓴     return 0;
}

❹ int isFull() {
❺       if (point == 10) return 1;
❻       return 0;
}

❷ void into(int num) {
❸❼     if (isFull() == 1) printf("Full");
❽       else isWhat[++point] = num;
}

⓰ int take() {
⓱㉑     if (isEmpty() == 1) printf("Empty");
㉒       else return isWhat[point--];
        return 0;
}

main() {
        ❶ into(5); ❾ into(2);
        ⓾⓮㉝ while (!isEmpty()) {
                ⓯㉓ printf("%d", take());
                ㉔ into(4); ㉕ into(1); ㉖ printf("%d", take());
                ㉗ into(3); ㉘ printf("%d", take()); ㉙ printf("%d", take());
                ㉚ into(6); ㉛ printf("%d", take()); ㉜ printf("%d", take());
        } ㉝
}
```

모든 C언어 프로그램은 반드시 main( ) 함수에서 시작한다.

❶ 5를 인수로 into( ) 함수를 호출한다.

❷ 반환값이 없는 into( ) 함수의 시작점이다. ❶번에서 전달받은 5를 num이 받는다.

❸ isFull( ) 함수를 호출하고 반환받은 값이 1이면 화면에 **Full**을 출력하고, 아니면 ❽번으로 이동한다.
❹ 정수를 반환하는 isFull( ) 함수의 시작점이다.
❺ point의 값이 10이면 함수를 호출했던 ❸번으로 1을 반환하고 아니면 ❻번으로 이동한다. point의 값이 -1 이므로 ❻번으로 이동한다.
❻ 함수를 호출했던 ❸번으로 0을 반환한다.
❼ ❻번에서 반환받은 값이 0이므로, ❽번으로 이동한다.
❽ isWhat[point]에 num의 값 5를 저장하는데, ++point는 전치연산이므로 point의 값이 먼저 1 증가되어 isWhat[0]에 5를 저장한다. into( ) 함수가 종료되었으므로 함수를 호출했던 ❶번의 다음 코드인 ❾번으로 이동한다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
isWhat	5									

❾ 2를 인수로 into( ) 함수를 호출한다. 앞서 진행된 ❷~❽번 과정을 반복하면 point는 1이 되고, isWhat[1]에는 2가 저장된다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
isWhat	5	2								

※ into( ) 함수의 역할은 인수로 전달된 값을 현재 isWhat 배열에 저장된 마지막 값 뒤에 저장하는 일입니다.

❿ isEmpty( )를 호출한 후 not 연산을 수행한 결과가 참(1)이면 ⓯~㉜번을 반복 수행한다.
⓫ 정수를 반환하는 isEmpty( ) 함수의 시작점이다.
⓬ point가 -1이면 함수를 호출했던 ⓮번으로 1을 반환하고, 아니면 ⓭번으로 이동한다. point의 값은 1이므로 ⓭번으로 이동한다.
⓭ 함수를 호출했던 ⓮번으로 0을 반환한다.
⓮ ⓭번에서 돌려받은 값 0(거짓)에 대한 not 연산은 1(참)이므로 ⓯번으로 이동한다.
⓯ take( ) 함수를 호출한 후 반환받은 값을 정수로 출력한다.
⓰ take( ) 함수의 시작점이다.
⓱ isEmpty( ) 함수를 호출한 후 반환받은 값이 1이면 **Empty**를 출력하고, 아니면 ㉒번으로 이동한다.
⓲ 정수를 반환하는 isEmpty( ) 함수의 시작점이다.
⓳ point가 -1이면 함수를 호출했던 ㉑번으로 1을 반환하고, 아니면 ⓴번으로 이동한다. point의 값은 1이므로 ⓴번으로 이동한다.
⓴ 함수를 호출했던 ㉑번으로 0을 반환한다.
㉑ ⓴번에서 반환받은 값이 0이므로 ㉒번으로 이동한다.
㉒ 함수를 호출했던 ㉓번으로 isWhat[point]의 값을 반환한다. point--는 후치연산이므로 먼저 isWhat[1]의 값 2를 반환하고, point의 값이 감소되어 point의 값은 0이 된다.
㉓ ㉒번에서 반환받은 값 2를 정수로 출력한다.

결과 2

㉔ 4를 인수로 into( ) 함수를 호출한다. ❷~❽번 과정을 반복하면 point는 1이 되고, isWhat[1]에는 4가 저장된다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
isWhat	5	4								

㉕ 1을 인수로 into( ) 함수를 호출한다. ❷~❽번 과정을 반복하면 point는 2가 되고, isWhat[2]에는 1이 저장된다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
isWhat	5	4	1							

㉖ ⓯~㉓번 과정을 반복하면 point는 1이 되고, isWhat[2]의 값 1이 출력된다.

결과 21

※ take( ) 함수의 역할은 현재 isWhat 배열에 저장된 마지막 값을 출력하는 것 입니다.

㉗ 3을 인수로 into( ) 함수를 호출한다. ❷~❽번 과정을 반복하면 point는 2가 되고, isWhat[2]에는 3이 저장된다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
isWhat	5	4	3							

㉘ ⓯~㉓번 과정을 반복하면 point는 1이 되고, isWhat[2]의 값 3이 출력된다.

결과 213

㉙ ⓯~㉓번 과정을 반복하면 point는 0이 되고, isWhat[1]의 값 4가 출력된다.

결과 2134

㉚ 6을 인수로 into( ) 함수를 호출한다. ❷~❽번 과정을 반복하면 point는 1이 되고, isWhat[1]에는 6이 저장된다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
isWhat	5	6								

㉛ ⓯~㉓번 과정을 반복하면 point는 0이 되고, isWhat[1]의 값 6이 출력된다.

결과 21346

㉜ ⓯~㉓번 과정을 반복하면 point는 -1이 되고, isWhat[0]의 값 5가 출력된다. 반복문이 종료되었으므로 while문의 처음인 ㉝번으로 이동한다.

결과 213465

㉝ ⓾~⓮번 과정을 반복한다. 이 때 point의 값이 -1이므로 ⓬번에서 1(참)이 반환되어 not 연산을 수행하므로 while(0)이 되어 반복이 종료된다. 이어서 ㉝번으로 이동하여 프로그램이 종료된다.

[문제 10]

① 요구 조건 분석
② 개념적 설계
③ 논리적 설계
④ 물리적 설계
⑤ 데이터베이스 구현

[문제 11]

① 싱글톤(Singleton)
② 방문자(Visitor)

[문제 12]

① Hamming
② FEC
③ BEC
④ Parity
⑤ CRC

[문제 13]

① 정보
② 감독
③ 비번호
④ 비동기 균형 모드
⑤ 비동기 응답 모드

[문제 14]

true
false
true
true

[해설]

```
public class Test {
        public static void main(String[] args) {
❶               String str1 = "Programming";
❷               String str2 = "Programming";
❸               String str3 = new String("Programming");
❹               System.out.println(str1==str2);
❺               System.out.println(str1==str3);
❻               System.out.println(str1.equals(str3));
❼               System.out.println(str2.equals(str3));
    }
}
```

❶ 문자열 객체 str1을 선언하고 "Programming"으로 초기화한다.
❷ 문자열 객체 str2를 선언하고 "Programming"으로 초기화한다.
※ 같은 문자열을 저장하는 문자열 객체는 동일한 주소를 가집니다.
❸ 문자열 객체 str3를 선언하고 "Programming"으로 초기화한다.
※ ❶, ❷와 달리 new String( )을 사용하는 경우 새로운 메모리 공간을 할당하여 문자열을 저장합니다. 즉 str1과 str2는 동일한 주소를 저장하고, str3은 다른 주소를 저장하고 있습니다.
❹ str1과 str2가 같으면 참을 의미하는 **true**를, 아니면 거짓을 의미하는 **false**를 출력한다. str1과 str2는 같은 주소를 저장하고 있으므로 **true**를 출력한다.

결과
```
true
```

❺ str1과 str3이 같으면 참을 의미하는 **true**를, 아니면 거짓을 의미하는 **false**를 출력한다. str1과 str3은 다른 주소를 저장하고 있으므로 **false**를 출력한다.

결과
```
true
false
```

❻ str1에 저장된 문자열과 str3에 저장된 문자열을 비교하여 같으면 참을 의미하는 **true**를, 아니면 거짓을 의미하는 **false**를 출력한다. str1과 str3에 저장된 문자열은 모두 "Programming"이므로 **true**를 출력한다.
• **A.equals(B)** : A 문자열과 B 문자열을 비교하여 두 데이터가 같으면 참을, 아니면 거짓을 반환한다.

결과
```
true
false
true
```

❼ str2에 저장된 문자열과 str3에 저장된 문자열을 비교하여 같으면 참을 의미하는 **true**를, 아니면 거짓을 의미하는 **false**를 출력한다. str2와 str3에 저장된 문자열은 모두 "Programming"이므로 **true**를 출력한다.

결과
```
true
false
true
true
```

[문제 15]

① DES, ARIA, SEED, AES

② RSA, ECC

[문제 16]

※ 다음 중 하나를 쓰면 됩니다.

해시, Hash

[문제 17]

CASCADE

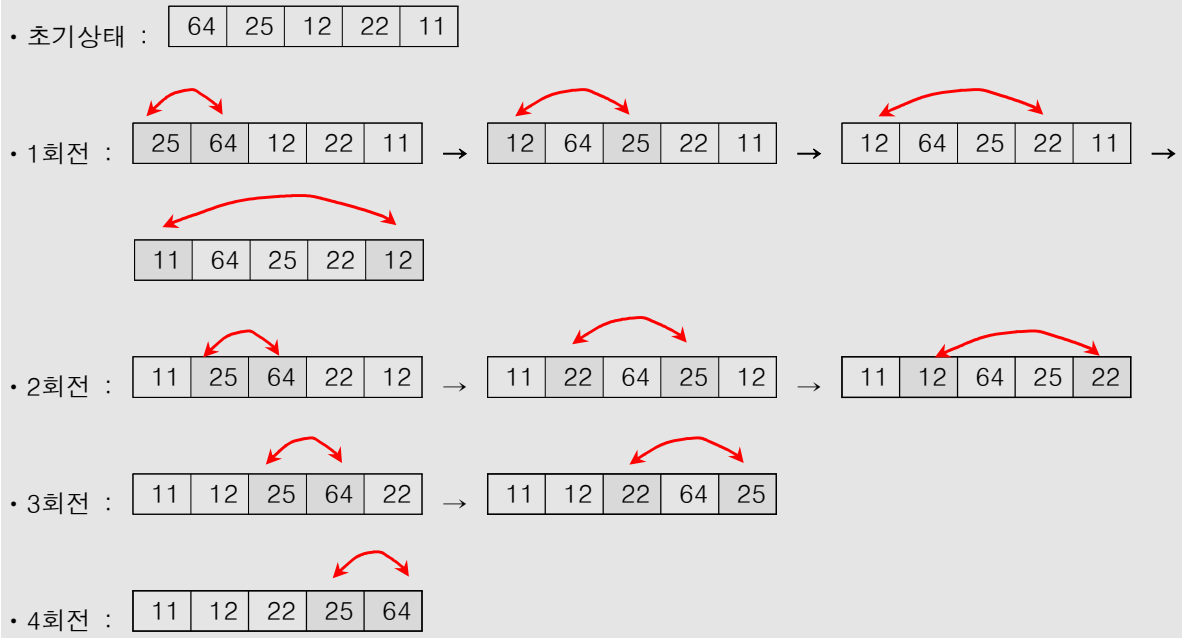
[문제 18]

>

[해설]

알고리즘의 이해

선택 정렬은 n개의 레코드 중에서 최소값을 찾아 첫 번째 레코드 위치에 놓고, 나머지 (n-1)개 중에서 다시 최소값을 찾아 두 번째 레코드 위치에 놓는 방식을 반복하여 정렬하는 방식입니다.



문제에서 오름차순으로 정렬한다고 하였고, 코드에서 반복 변수 `i`가 0에서 n-1까지 증가할 때 `j`가 `i`+1부터 n까지 증가하는 것으로 보아 `i`가 기준값의 배열 위치이고 `j`가 비교 대상의 배열 위치임을 알 수 있습니다. 오름차순 정렬은 비교 기준 값이 비교 대상 보다 큰 경우에 위치 교환이 이루어져야 하므로 괄호에 들어갈 연산자는 **>** 입니다.

```
#include <stdio.h>
main() {
❶       int E[] = { 64, 25, 12, 22, 11 };
❷       int n = sizeof(E) / sizeof(E[0]);
❸       int i = 0;
❹       do {
❺               int j = i + 1;
❻               do {
❼                       if (E[i] > E[j]) {
❽                               int tmp = E[i];
❾                               E[i] = E[j];
❿                               E[j] = tmp;
                        }
⓫                       j++;
⓬               } while (j < n);
⓭               i++;
⓮       } while (i < n - 1);
⓯       for (int i = 0; i <= 4; i++)
⓰               printf("%d ", E[i]);
}
```

❶ 5개의 요소를 갖는 정수형 배열 E를 선언하고 초기화한다.

	[0]	[1]	[2]	[3]	[4]
E	64	25	12	22	11

❷ 정수형 변수 n을 선언하고 E의 크기를 E[0]의 크기로 나눈 값으로 초기화한다. 배열 전체 길이를 배열 한 개의 요소의 길이로 나누는 것이므로 배열의 길이인 5가 반환되어 n에 저장된다.

- sizeof( ) : 객체의 크기를 구하는 함수

❸ 정수형 변수 i를 선언하고 0으로 초기화한다.

❹ do ~ while 반복문의 시작점이다. ❺~⓭번을 반복 수행한다.

❺ 정수형 변수 j를 선언하고 i+1의 값으로 초기화한다.

❻ do ~ while 반복문의 시작점이다. ❼~⓫번을 반복 수행한다.

❼ E[i]가 E[j]보다 크면 ❽~❿번을 수행한다.

❽~❿ E[i]와 E[j]의 값을 교환하는 과정이다.

⓫ 'j = j + 1;'과 동일하다. j의 값을 1씩 누적시킨다.

⓬ j가 n보다 작은 동안 ❼~⓫번을 반복 수행한다.

⓭ 'i = i + 1;'과 동일하다. i의 값을 1씩 누적시킨다.

⓮ i가 n-1보다 작은 동안 ❺~⓭번을 반복 수행한다. 반복문 실행에 따른 변수들의 변화는 다음과 같다.

n	i	j	배열 E [0] [1] [2] [3] [4]	tmp
5	0	1	64 25 12 22 11	64
		2	25 64 25 12	25
		3	12	12
		4	11	
		5		
	1	2	11 64 25 22 12	64
		3	25 64 25 22	25
		4	22	22
		5	12	
	2	3	11 12 64 25 22	64
		4	25 64 25	25
		5	22	
	3	4	11 12 22 64 25	64
		5	25 64	
	4			

⓯ 반복 변수 i가 0부터 1씩 증가하면서 4보다 작거나 같은 동안 ⓰번을 반복 수행한다.

⓰ E[i]의 값과 공백 한 칸을 출력한다.

결과 11 12 22 25 64

[문제 19]

engneing

[해설]

```
❶ a = "engineer information programming"
❷ b = a[:3]
❸ c = a[4:6]
❹ d = a[29:]
❺ e = b + c + d
❻ print(e)
```

❶ 변수 a에 문자열을 저장한다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]	[19]	[20]	[21]	[22]	[23]	[24]	[25]	[26]	[27]	[28]	[29]	[30]	[31]
a	e	n	g	i	n	e	e	r		i	n	f	o	r	m	a	t	i	o	n		p	r	o	g	r	a	m	m	i	n	g

❷ 변수 b에 a에 저장된 문자열 중에서 처음 위치부터 2(3-1) 위치까지의 요소를 가져와 저장한다.

	[0]	[1]	[2]
b	e	n	g

❸ 변수 c에 a에 저장된 문자열 중에서 4부터 5(6-1) 위치까지의 요소를 가져와 저장한다.

	[0]	[1]
c	n	e

❹ 변수 d에 a에 저장된 문자열 중에서 29부터 마지막 위치까지의 요소를 가져와 저장한다.

	[0]	[1]	[2]
d	i	n	g

❺ b, c, d에 저장된 문자열을 합한 후 변수 e에 저장한다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
e	e	n	g	n	e	i	n	g

❻ e의 값을 출력한다.

결과 engneing

[문제 20]

※ 번호별로 다음 중 하나를 쓰면 됩니다.

① 스텁, stub

② 드라이버, driver