



시험에 나오는것만 공부한다!

**시나공시리즈**

기출문제

2025년 1회 정보처리기사 실기



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

#### 저작권 안내

이 자료는 시나공 카페 회원을 대상으로 하는 자료로서 개인적인 용도로만 사용할 수 있습니다. 허락 없이 복제하거나 다른 매체에 옮겨 실을 수 없으며, 상업적 용도로 사용할 수 없습니다.

#### \*\*\* 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 사용해야 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 굿지 않은 답안은 정정하지 않은 것으로 간주합니다.
7. 답안의 한글 또는 영문의 오타자는 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 설명에 해당하는 네트워크 보안 관련 용어를 쓰시오. (5점)



- 상호 인증 과정을 거친 후 접속해 있는 서버와 서로 접속되어 클라이언트 사이의 세션 정보를 가로채는 공격 기법이다.
- 접속을 위한 인증 정보 없이도 가로챈 세션을 이용해 공격자가 원래의 클라이언트인 것처럼 위장하여 서버의 자원이나 데이터를 무단으로 사용한다.
- TCP 3-Way-Handshake 과정에 끼어들어서 클라이언트와 서버 간의 동기화된 시퀀스 번호를 가로채 서버에 무단으로 접근하는 공격이 대표적이다.

답 :

**문제 2** 다음은 무결성 제약 조건에 대한 설명이다. 괄호(①~③)에 들어갈 알맞은 용어를 <보기>에서 찾아 쓰시오. (5점)



구분	( ① ) 무결성 제약 조건	( ② ) 무결성 제약 조건	( ③ ) 무결성 제약 조건
제약 대상	속성	튜플	속성, 튜플
NULL 값		기본키	외래키
릴레이션 내 제약 조건 개수	속성의 개수와 동일	1개	0~여러 개

<보기>

• 데이터	• 사용자 정의	• 도메인	• 키
• 속성	• 개체	• 범위	• 참조

답

- ①
- ②
- ③

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.



**문제 3** 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
#include <stdio.h>
char Data[5] = {'B', 'A', 'D', 'E'};
char c;

int main( ) {
    int i, temp, temp2;
    c = 'C';
    printf("%d\n", Data[3]-Data[1]);
    for(i = 0; i < 5; ++i) {
        if(Data[i] > c)
            break;

        temp = Data[i];
        Data[i] = c;
        i++;

        for(; i < 5; ++i) {
            temp2 = Data[i];
            Data[i] = temp;
            temp = temp2;
        }

        for(i = 0; i < 5; i++) {
            printf("%c", Data[i]);
        }
    }
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

**문제 4** 다음 설명에 해당하는 알맞은 용어를 영문 약어로 쓰시오. (5점)



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

답 :

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



```
public class Main {  
    public static void main(String[] args) {  
        int a = 5, b = 0;  
        try {  
            System.out.print(a/b);  
        }  
        catch(ArithmeticException e){  
            System.out.print("출력1");  
        }  
        catch(ArrayIndexOutOfBoundsException e) {  
            System.out.print("출력2");  
        }  
        catch(NumberFormatException e) {  
            System.out.print("출력3");  
        }  
        catch(Exception e) {  
            System.out.print("출력4");  
        }  
        finally {  
            System.out.print("출력5");  
        }  
    }  
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.



**문제 6** 다음 <emp>와 <sal> 테이블을 참조하여 <SQL문>을 실행했을 때 출력되는 결과를 쓰시오. (5점)

<emp>

id	name
1002	홍길동
1004	강감찬
1006	김유신
1008	이순신

<sal>

id	incentive
1002	300
1004	400
1008	1000
1009	500

<SQL문>

```
SELECT name, incentive
FROM emp, sal
WHERE emp.id = sal.id and incentive >= 500;
```

답 :



**문제 7** 다음은 결합도(Coupling)에 대한 설명이다. 괄호(①~③)에 들어갈 알맞은 결합도를 <보기>에서 찾아 기호(㉠~㉣)로 쓰시오. (5점)

- ( ① ) : 한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도이다.
- ( ② ) : 모듈 간의 인터페이스로 배열이나 레코드 등의 자료 구조가 전달될 때의 결합도이다.
- ( ③ ) : 공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도로, 파라미터가 아닌 모듈 밖에 선언된 전역 변수를 사용하여 전역 변수를 갱신하는 방식으로 상호작용하는 때의 결합도이다.

<보기>

- |          |           |          |
|----------|-----------|----------|
| ㉠ 외부 결합도 | ㉡ 스템프 결합도 | ㉢ 제어 결합도 |
| ㉣ 자료 결합도 | ㉤ 내용 결합도  | ㉥ 공통 결합도 |

답

- ①
- ②
- ③

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

**문제 8** 다음 설명에 해당하는 용어를 <보기>에서 찾아 쓰시오. (5점)



사용자에게 컴퓨터가 바이러스에 감염되었다고 속이거나 보안 위험이 있다고 거짓 경고 메시지를 보내어 불안감을 조성한 후 문제 해결을 명목으로 불필요한 소프트웨어 구매를 유도하는 악성 소프트웨어이다.

<보기>

- |         |        |         |
|---------|--------|---------|
| • 랜섬웨어  | • 애드웨어 | • 스파이웨어 |
| • 스케어웨어 | • 키로거  | • 봇넷    |

답 :

**문제 9** 현재 IP 주소가 192.168.35.10이고, 서브넷 마스크가 255.255.252.0인 컴퓨터에서



브로드캐스팅으로 다른 IP로 정보를 전달하려고 한다. 이때 수신할 수 있는 알맞은 IP 주소를 <보기>에서 골라 모두 작성하시오. (5점)

<보기>

- |                                      |
|--------------------------------------|
| <input type="radio"/> 192.168.32.1   |
| <input type="radio"/> 192.168.33.50  |
| <input type="radio"/> 192.168.34.126 |
| <input type="radio"/> 192.168.35.100 |
| <input type="radio"/> 192.168.35.200 |

답 :

**문제 10** 프로토콜에 대한 관한 다음 설명에서 괄호(①, ②)에 들어갈 알맞은 용어를 쓰시오. (5점)



( ① )는 호스트의 IP 주소를 호스트와 연결된 네트워크 접속 장치의 물리적 주소(MAC Address)로 변환하는 프로토콜이고, ( ② )는 ( ① )와 반대로 물리적 주소를 IP 주소로 변환하는 프로토콜이다.

답 :

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

연 습 란



**문제 11** 다음 Java로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
public class Main {
    public static void main(String[] args) {
        new Child( );
        System.out.println(Parent.total);
    }
}

class Parent {
    static int total = 0;
    int v = 1;
    public Parent( ) {
        total += ++v;
        show( );
    }
    public void show( ) {
        total += total;
    }
}

class Child extends Parent {
    int v = 10;
    public Child( ) {
        v += 2;
        total += v++;
        show( );
    }
    @Override
    public void show( ) {
        total += total * 2;
    }
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.



**문제 12** 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
#include <stdio.h>
#include <stdlib.h>
void set(int** arr, int* data, int rows, int cols) {
    for (int i = 0; i < rows * cols; ++i) {
        arr[((i + 1) / rows) % rows][((i + 1) % cols)] = data[i];
    }
}

int main( ) {
    int rows = 3, cols = 3, sum = 0;
    int data[] = {5, 2, 7, 4, 1, 8, 3, 6, 9};
    int** arr;
    arr = (int**) malloc(sizeof(int*) * rows);
    for (int i = 0; i < cols; i++) {
        arr[i] = (int*) malloc(sizeof(int) * cols);
    }
    set(arr, data, rows, cols);
    for (int i = 0; i < rows * cols; i++) {
        sum += arr[i / rows][i % cols] * (i % 2 == 0 ? 1 : -1);
    }
    for(int i = 0; i < rows; i++) {
        free(arr[i]);
    }
    free(arr);
    printf("%d", sum);
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.





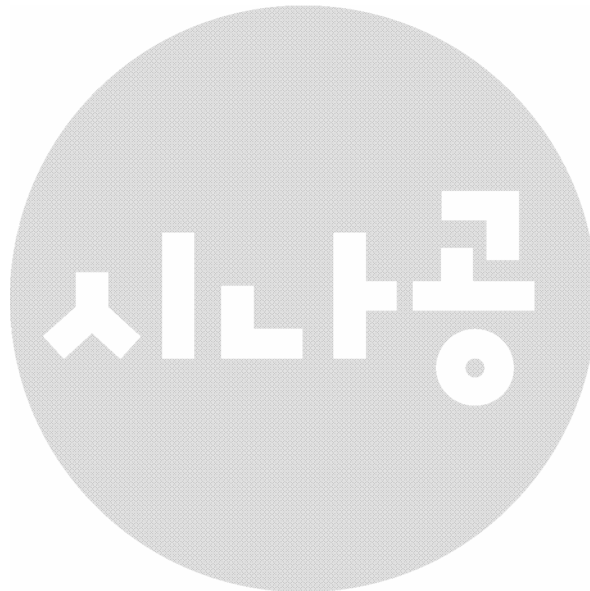
**문제 13** 다음 설명에 해당하는 디자인 패턴을 <보기>에서 찾아 쓰시오. (5점)

- 호환성이 없는 클래스들의 인터페이스를 다른 클래스가 이용할 수 있도록 변환해주는 패턴이다.
- 기존의 클래스를 이용하고 싶지만 인터페이스가 일치하지 않을 때 이용한다.

<보기>

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

답 :



연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.

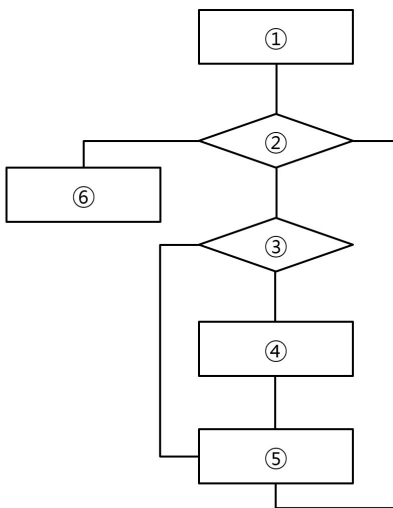


**문제 14** 다음에 제시된 코드를 이용하여 순서도의 번호(①~⑥)에 들어갈 알맞은 코드를 작성하고, 문장 커버리지로 구성할 테스트 케이스를 작성하시오. (5점)

<코드>

```
int Main(int b[], int m, int x) {  
    int a = 0;  
    while (a < m || b[a] < x) {  
        if (b[a] < 0)  
            b[a] = -b[a];  
        a++;  
    }  
    return 1;  
}
```

<순서도>



답

- ①
- ②
- ③
- ④
- ⑤
- ⑥

• 문장 커버리지 기준 테스트 케이스 : ① → ② → (    ) → (    ) → (    ) → (    ) → (    )

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.



**문제 15** 데이터베이스와 관련된 용어들에 대한 다음 설명에서 괄호(①~④)에 들어갈 알맞은 답을

<보기>에서 찾아 기호(㉠~㉤)로 쓰시오. (5점)

- ( ① ) : 릴레이션에서 속성의 개수
- ( ② ) : 릴레이션에서 튜플의 개수
- ( ③ ) : 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합
- ( ④ ) : 하나의 애트리뷰트가 취할 수 있는 같은 타입의 원자값들의 집합

<보기>

- |               |             |           |             |
|---------------|-------------|-----------|-------------|
| ㉠ Domain      | ㉡ Attribute | ㉢ Degree  | ㉣ Candidate |
| ㉤ Cardinality | ㉥ Tuple     | ㉦ Foreign | ㉧ Alternate |

답

- ①
- ②
- ③
- ④



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

```
public class Main {
    public static void main(String[] args) {
        System.out.println(calc("5"));
    }
    static int calc(int value) {
        if (value <= 1) return value;
        return calc(value - 1) + calc(value - 2);
    }
    static int calc(String str) {
        int value = Integer.valueOf(str);
        if (value <= 1) return value;
        return calc(value - 1) + calc(value - 3);
    }
}
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.



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

```
#include <stdio.h>

typedef struct student {
    char* name;
    int score[3];
} Student;

int dec(int enc) {
    return enc & 0xA5;
}

int sum(Student* p) {
    return dec(p->score[0]) + dec(p->score[1]) + dec(p->score[2]);
}

int main( ) {
    Student s[2] = { "Kim", {0xA0, 0xA5, 0xDB}, "Lee", {0xA0, 0xED, 0x81} };
    Student* p = s;
    int result = 0;
    for (int i = 0; i < 2; i++) {
        result += sum(&s[i]);
    }
    printf("%d", result);
    return 0;
}
```

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)

```
public class Main {
    public static void main(String[] args) {
        int[] data = {3, 5, 8, 12, 17};
        System.out.println(func(data, 0, data.length - 1));
    }
    static int func(int[] a, int st, int end) {
        if (st >= end) return 0;
        int mid = (st + end) / 2;
        return a[mid] + Math.max(func(a, st, mid), func(a, mid + 1, end));
    }
}
```

답 :



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

```
class Node:
    def __init__(self, value):
        self.value = value
        self.children = []
def tree(li):
    nodes = [Node(i) for i in li]
    for i in range(1, len(li)):
        nodes[(i - 1) // 2].children.append(nodes[i])
    return nodes[0]
def calc(node, level=0):
    if node is None:
        return 0
    return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in
        node.children)
li = [3, 5, 8, 12, 15, 18, 21]
root = tree(li)
print(calc(root))
```

답 :

연 습 란

※ 다음 여백은 연습란으로 사용하시기 바랍니다.



**문제 20** 다음 C 언어로 구현된 프로그램을 분석하여 그 실행 결과를 쓰시오. (단, 출력문의 출력 서식을 준수하시오.) (5점)

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Data {
    int value;
    struct Data *next;
} Data;

Data* insert(Data* head, int value) {
    Data* new_node = (Data*)malloc(sizeof(Data));
    new_node -> value = value;
    new_node -> next = head;
    return new_node;
}

Data* reconnect(Data* head, int value) {
    if (head == NULL || head -> value == value) return head;
    Data *prev = NULL, *curr = head;
    while (curr != NULL && curr -> value != value) {
        prev = curr;
        curr = curr -> next;
    }
    if (curr != NULL && prev != NULL) {
        prev -> next = curr -> next;
        curr -> next = head;
        head = curr;
    }
    return head;
}

int main( ) {
    Data *head = NULL, *curr;
    for (int i = 1; i <= 5; i++)
        head = insert(head, i);
    head = reconnect(head, 3);
    for (curr = head; curr != NULL; curr = curr -> next)
        printf("%d", curr->value);
    return 0;
}
```

※ 다음 **답** 여백은 연습란으로 사용하시기 바랍니다.

연 습 란

## 기출문제 정답 및 해설

### [문제 1]

세션 하이재킹(Session Hijacking)

### [문제 2]

① 도메인    ② 개체    ③ 참조

### [문제 3]

4

BACDE

※ 답안 작성 시 주의 사항 : 프로그램의 실행 결과는 부분 점수가 없으므로 정확하게 작성해야 합니다. 예를 들어, 출력값 사이에 공백이나逗를 넣어 4, BACDE나 4 BACDE로 썼을 경우 부분 점수 없이 완전히 틀린 것으로 간주됩니다. 또한 C, Java, Python 등의 프로그래밍 언어에서는 대소문자를 구분하기 때문에 출력 결과도 대소문자를 구분하여 정확하게 작성해야 합니다. 예를 들어, 두 번째 줄 결과는 소문자 bacde로 썼을 경우 부분 점수 없이 완전히 틀린 것으로 간주됩니다.

### [해설]

```
#include <stdio.h>
㉠ char Data[5] = {'B', 'A', 'D', 'E'};
㉡ char c;

int main( ) {
㉢     int i, temp, temp2;
㉣     c = 'C';
㉤     printf("%d\n", Data[3]-Data[1]);
㉥     for(i = 0; i < 5; ++i) {
㉦         if(Data[i] > c)
㉧             break;
        }

㉨     temp = Data[i];
㉩     Data[i] = c;
㉪     i++;

㉫     for(; i < 5; ++i) {
㉬         temp2 = Data[i];
㉭         Data[i] = temp;
㉮         temp = temp2;
        }

㉯     for(i = 0; i < 5; i++) {
㉰         printf("%c", Data[i]);
        }
}
```

※ main( ) 함수 밖에서 선언된 변수는 전역 변수로, 코드가 실행되는 동안 어디에서든 접근할 수 있습니다.

㉔ 5개의 요소를 갖는 문자형 배열 Data를 선언하고, 초기화한다.

	[0]	[1]	[2]	[3]	[4]
Data	66	65	68	69	
	'B'	'A'	'D'	'E'	

※ 문자가 메모리에 저장될 때 문자로 저장되는 것이 아니라 해당 문자의 아스키코드 값이 저장됩니다. 즉 'B', 'A', 'D', 'E'는 각각에 해당하는 아스키코드 값이 저장됩니다.

㉕ 문자형 변수 c를 선언한다.

모든 C 언어 프로그램은 반드시 main( ) 함수에서 시작한다.

① 정수형 변수 i, temp, temp2를 선언한다.

② 문자형 전역 변수 c에 'C'를 저장한다.

※ 문자가 메모리에 저장될 때 문자로 저장되는 것이 아니라 해당 문자의 아스키코드 값이 저장됩니다. 즉 'C'는 'C'에 해당하는 아스키코드 값인 67이 저장됩니다.

- 알파벳 대문자의 아스키코드 값은 'A(65)' ~ 'Z(90)'입니다.

- 알파벳 소문자의 아스키코드 값은 'a(97)' ~ 'z(122)'입니다.

③ Data[3]과 Data[1]에는 아스키코드 값 69와 65가 저장되어 있으므로, Data[3]-Data[1]의 결과 4를 정수로 출력한 후 커서를 다음 줄의 처음으로 이동한다.

결과 4

④ 반복 변수 i가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ⑤~⑥번을 반복 수행한다.

⑤ Data[i]가 c보다 크면 ⑥번을 수행하고, 그렇지 않으면 ④번으로 이동하여 반복 과정을 계속 수행한다.

⑥ for문을 벗어나 ⑦번으로 이동한다.

반복문의 실행에 따른 변수들의 변화는 다음과 같다.

c	i	Data[i]	Data[i] > c
67	0	66	No
	1	65	No
	2	68	Yes

⑦ for문을 벗어날 때 i의 값은 2이므로, temp에 Data[2]의 값인 68을 저장한다.

⑧ Data[2]에 c의 값 67을 저장한다.

	[0]	[1]	[2]	[3]	[4]
Data	66	65	67	69	
	'B'	'A'	'C'	'E'	

⑨ 'i = i + 1;'과 동일하다. i는 1 증가하여 3이 된다.

⑩ 초기값이 생략되었지만 i는 3이다. i를 1씩 증가하면서 5보다 작은 동안 ⑪~⑬번을 반복 수행한다.

⑪ ~ ⑬ 임시 변수 temp2를 사용하여 Data[i]와 temp의 값을 교환하는 과정이다.

반복문의 실행에 따른 변수들의 변화는 다음과 같다.

i	Data[i]	temp2	temp	Data 배열				
				[0]	[1]	[2]	[3]	[4]
3	69	69	68	<div><div>66</div><div>65</div><div>67</div><div>69</div><div></div></div>				
	68		68					
4	Null	Null	69	<div><div>66</div><div>65</div><div>67</div><div>68</div><div>69</div></div>				
	69							
5								



⑭ 반복 변수 i가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ⑮번을 반복 수행한다.

⑮ Data[i]의 값을 문자로 출력한다.

결과

4

BACDE

※ 출력 서식 문자열이 '%c'이므로 Data 배열에 저장된 아스키코드 값에 해당하는 문자가 출력됩니다.

#### [문제 4]

CRC

#### [문제 5]

출력1출력5

※ **답안 작성 시 주의 사항** : 프로그램의 실행 결과는 부분 점수가 없으므로 정확하게 작성해야 합니다. 예를 들어, 출력값 사이에 공백이나 콤마를 넣어 **출력1, 출력5**나 **출력1 출력5**로 썼을 경우 부분 점수 없이 완전히 틀린 것으로 간주됩니다.

#### [해설]

```
public class Main {  
    public static void main(String[ ] args) {  
①        int a = 5, b = 0;  
②        try {  
③            System.out.print(a/b);  
        }  
④        catch(ArithmeticException e){  
⑤            System.out.print("출력1");  
        }  
        catch(ArrayIndexOutOfBoundsException e) {  
            System.out.print("출력2");  
        }  
        catch(NumberFormatException e) {  
            System.out.print("출력3");  
        }  
        catch(Exception e) {  
            System.out.print("출력4");  
        }  
⑥        finally {  
⑦            System.out.print("출력5");  
        }  
    }  
}
```

모든 Java 프로그램은 반드시 main( ) 메소드에서 시작한다.

① 정수형 변수 a와 b를 선언하고, 5와 0으로 초기화한다.

② 예외 구문의 시작이다.

③ '5/0'은 ArithmeticException 예외를 발생시키므로 try 블록이 종료되고 catch 블록으로 제어가 이동된다. ④번으로 이동한다.

• ArithmeticException : 0으로 나누는 등의 산술 연산에 대한 예외가 발생한 경우 사용하는 예외 객체

④ ArithmeticException에 해당하는 예외를 다루는 catch문의 시작이다.

⑤ 출력1을 출력한다. try ~ catch문이 종료되었으므로 ⑥번으로 이동한다.

결과 출력1

⑥ try ~ catch문이 모두 종료되면 실행되는 finally문의 시작이다.

⑦ 출력5를 출력한다.

결과 출력1출력5

#### [문제 6]

name	incentive
이순신	1000

#### [해설]

```
SELECT name, incentive
FROM emp, sal
WHERE emp.id = sal.id
      and incentive >= 500;
```

‘name’과 ‘incentive’ 속성을 표시한다.

<emp>와 <sal> 테이블을 대상으로 조회한다.

<emp> 테이블의 ‘id’와 <sal> 테이블의 ‘id’가 같고

‘incentive’가 500 이상인 자료만을 대상으로 한다.

#### [과정]

① <emp> 테이블의 ‘id’와 <sal> 테이블의 ‘id’가 같은 튜플은 다음과 같다.

<emp>

id	name
1002	홍길동
1004	강감찬
1006	김유신
1008	이순신

<sal>

id	incentive
1002	300
1004	400
1008	1000
1009	500

② <emp> 테이블의 ‘id’와 <sal> 테이블의 ‘id’가 같은 튜플 중에서 ‘incentive’가 500 이상인 튜플은 다음과 같다.

<emp>

id	name
1002	홍길동
1004	강감찬
1008	이순신

<sal>

id	incentive
1002	300
1004	400
1008	1000

③ ‘name’과 ‘incentive’ 속성을 표시한다.

name	incentive
이순신	1000

#### [문제 7]

① ㉠ ② ㉡ ③ ㉢

#### [문제 8]

스케어웨어

[문제 9]

㉠, ㉡, ㉢, ㉣, ㉤

[해설]

• 서브넷 마스크 255.255.252.0을 2진수로 표현하면 11111111 11111111 11111100 00000000입니다.

• IP 주소 192.168.35.10의 네트워크 주소는 서브넷 마스크와 AND 연산을 수행하면 됩니다.

11000000 10101000 00100011 00001010 ← 192.168.35.10

11111111 11111111 11111100 00000000 ← 255.255.252.0

-----  
11000000 10101000 00100000 00000000 ← 192.168.32.0

• 브로드 캐스트 주소는 IP 주소 192.168.35.10의 네트워크 주소의 마지막 10비트(서브넷 마스크에서 0의 개수)를 1로 설정하면 됩니다.

11000000 10101000 00100011 11111111 ← 192.168.35.255

∴ 문제에서 요구한 수신 가능한 IP 주소는 네트워크 시작 주소와 브로드 캐스트 주소를 제외한 192.168.32.1 ~ 192.168.35.254까지로, 보기로 제시된 주소가 모두 수신 가능한 주소입니다.

[문제 10]

① ARP(Address Resolution Protocol)    ② RARP(Reverse Address Resolution Protocol)

[문제 11]

54

[해설]

```

public class Main {
    public static void main(String[] args) {
        ❶      new Child( );
        ❷      System.out.println(Parent.total);
    }
}

A class Parent {
    static int total = 0;           // total은 클래스 변수로 Parent 클래스에서는 공유된다. 클래스가
                                   // 로딩될 때 생성된다.

    int v = 1;                     // v는 인스턴스 변수로 해당 클래스 안에서만 사용된다. 클래스가
                                   // 접근될 때 생성된다.

    ❸      public Parent( ) {
        ❹      total += ++v;
        ❺      show( );
    }
    C public void show( ) {
        total += total;
    }
}

B class Child extends Parent {
    int v = 10;

    ❻      public Child( ) {
        ❼      v += 2;
        ❽      total += v++;
        ❾      show( );
    }

    @Override
    D ❺❶ public void show( ) {
        ❻❷      total += total * 2;
    }
}

```

① Parent 클래스를 정의한다.

② Child 클래스를 정의하고 부모 클래스로 Parent 클래스를 지정하면서 Parent 클래스에 속한 변수와 메소드를 상속받는다.

③ 반환값 없는 메소드 show( )를 정의한다.

④ 반환값 없는 메소드 show( )를 정의한다. ②에서 Child 클래스는 Parent 클래스의 메소드를 사용할 수 있다고 했으므로 Child 클래스에는 이름이 같은 메소드(③, ④) show( )가 두 개 존재하게 된다. 이와 같이 상속으로 인해 동일한 이름의 메소드가 여러 개인 경우, 부모 클래스에 정의된 show( ) 메소드(③)는 자식 클래스의 show( ) 메소드(④)에 의해 재정의되어 자식 클래스의 show( ) 메소드(④)만 사용되는데, 이를 메소드 오버라이딩 또는 메소드 재정의라고 한다.

모든 Java 프로그램은 반드시 main( ) 메소드에서 시작한다.

❶ Child 클래스의 생성자를 호출한다. Child 클래스는 Parent 클래스에 속한 변수와 메소드를 상속받는 관계이므로, 부모 클래스인 Parent 클래스의 생성자가 먼저 호출된 후 Child 생성자가 호출된다.

❷ Parent 클래스의 생성자인 Parent( ) 메소드의 시작점이다.

❸ ++v는 전치 연산이므로 'total = total + (v=v+1)'와 동일하다. 즉 v의 값을 1 증가시킨 후 total에 누적한다.

Parent 클래스		Child 클래스
total	v	v
0	1	
2	2	

- ④ show( ) 메소드를 호출한다. Parent 클래스의 show( ) 메소드는 Child 클래스의 show( ) 메소드에 의해 재정의되었으므로 ⑤번으로 이동한다.
- ⑤ 반환값이 없는 show( ) 메소드의 시작점이다.
- ⑥ 'total = total + (total\*2);'와 동일하다. total이 2이므로 '(2\*2)'의 결과 4를 total에 누적한다. show( ) 메소드가 종료되었으므로 메소드를 호출했던 곳으로 이동한다. 이어서 Parent 클래스의 생성자인 Parent( ) 메소드도 종료되었으므로 Child 생성자가 호출된다. ⑦번으로 이동한다.

Parent 클래스		Child 클래스
total	v	v
0	1	
2	2	
6		

- ⑦ Child 클래스의 생성자인 Child( ) 메소드의 시작점이다.
- ⑧ 'v = v + 2;'와 동일하다. v의 값에 2를 누적한다. 변수 v는 Parent 클래스와 Child 클래스 각각에서 선언된 인스턴스 변수로 해당 클래스에서 구분되어 사용된다.

Parent 클래스		Child 클래스
total	v	v
0	1	10
2	2	12
6		

- ⑨ v++는 후치 연산이므로 'total = total + v, v=v+1'과 동일하다. 즉 현재 v의 값 12를 total에 누적시킨 후 v의 값을 1 증가시킨다. total 변수는 상속 관계에 있는 Child 클래스에서도 공유한다.

Parent 클래스		Child 클래스	
total	v	total	v
0	1	0	10
2	2	2	12
6		6	13
18		18	

- ⑩ show( ) 메소드를 호출한다. Parent 클래스의 show( ) 메소드는 Child 클래스의 show( ) 메소드에 의해 재정의되었으므로 ⑪번으로 이동한다.
- ⑪ 반환값이 없는 show( ) 메소드의 시작점이다.
- ⑫ 'total = total + (total\*2);'와 동일하다. total이 18이므로 '(18\*2)'의 결과인 36을 total에 누적한다. show( ) 메소드가 종료되었으므로 메소드를 호출했던 ①번의 다음 줄인 ⑬번으로 이동한다.

Parent 클래스		Child 클래스	
total	v	total	v
0	1	0	10
2	2	2	12
6		6	13
18		18	
54		54	

- ⑬ Parent의 클래스의 total을 출력한다.
- 결과 54

[문제 12]

13

[해설]

```
#include <stdio.h>
#include <stdlib.h>
⑧ void set(int** arr, int* data, int rows, int cols) {
⑨     for (int i = 0; i < rows * cols; ++i) {
⑩         arr[((i + 1) / rows) % rows][((i + 1) % cols) % cols] = data[i];
    }
}

int main( ) {
①     int rows = 3, cols = 3, sum = 0;
②     int data[] = {5, 2, 7, 4, 1, 8, 3, 6, 9};
③     int** arr;
④     arr = (int**) malloc(sizeof(int*) * rows);
⑤     for (int i = 0; i < rows; i++) {
⑥         arr[i] = (int*) malloc(sizeof(int) * cols);
    }
⑦     set(arr, data, rows, cols);
⑪     for (int i = 0; i < rows * cols; i++) {
⑫         sum += arr[i / rows][i % cols] * (i % 2 == 0 ? 1 : -1);
    }
⑬     for(int i = 0; i < rows; i++) {
⑭         free(arr[i]);
    }
⑮     free(arr);
⑯     printf("%d", sum);
}
```

모든 C 언어 프로그램은 반드시 main( ) 함수에서 시작한다.

① 정수형 변수 rows, cols, sum을 선언하고, 각각 3, 3, 0으로 초기화한다.

② 9개의 요소를 갖는 정수형 배열 data를 선언하고 초기화한다. 개수를 지정하지 않았으므로, 초기값으로 지정된 수만큼 배열의 요소가 만들어진다.

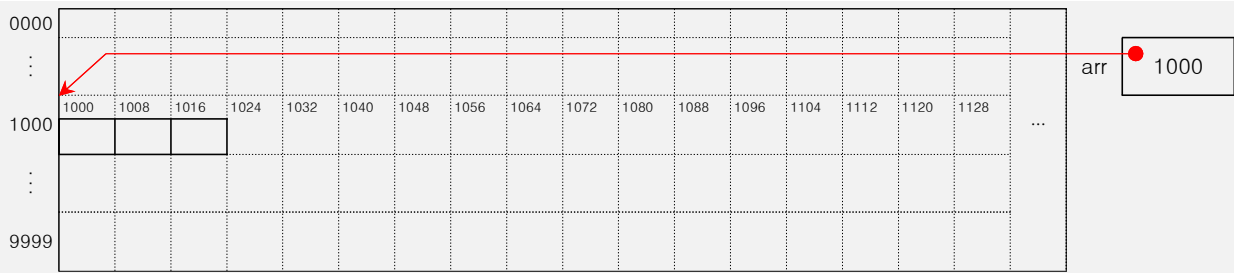
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
data	5	2	7	4	1	8	3	6	9

③ 정수형 이중포인터 변수 arr을 선언한다.

④ malloc 함수가 메모리에서 정수형 포인터 자료형의 크기 × rows, 즉 24Byte의 빈 영역을 찾아 할당한 다음 그 영역의 시작 주소를 이중포인터 변수 arr에 저장한다. 이제 arr은 할당된 공간의 시작 주소를 가리킨다.

※ malloc 함수가 동적으로 할당하는 것이므로 여기서 지정한 주소 1000은 임의로 정한 것이며, 이해를 돕기 위해 주소를 10진수로 표현했습니다.

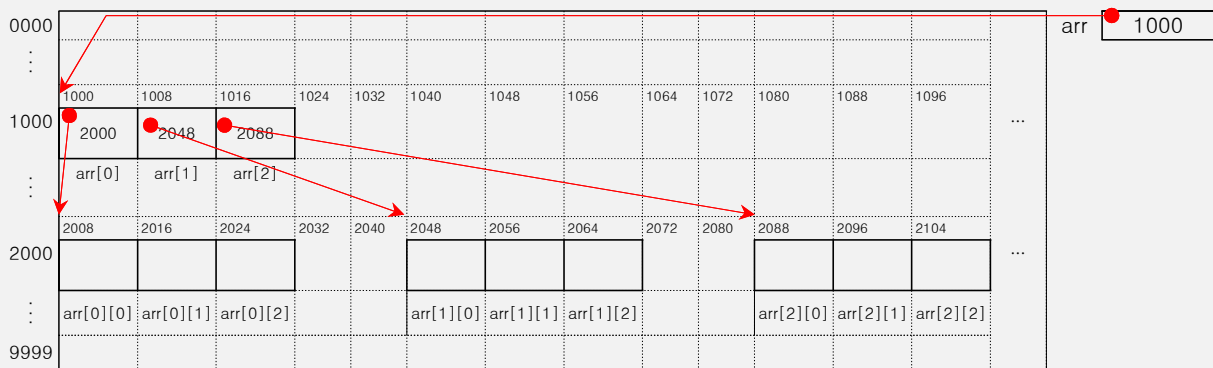
메모리



⑤ 반복 변수  $i$ 가 0에서 시작하여 1씩 증가하면서 3보다 작은 동안 ⑥번을 반복 수행한다.

⑥ malloc 함수가 메모리에서 정수형 자료형의 크기  $\times$  cols, 즉 12Byte의 빈 영역을 찾아 할당한 다음 그 영역의 시작 주소를 포인터 변수  $arr[i]$ 에 저장한다.  $arr$ 이 이중 포인터 변수이므로  $arr[0] \sim arr[2]$ 은 포인터 변수가 된다.

### 메모리



⑦  $arr$ ,  $data$ ,  $rows$ ,  $cols$ 를 인수로 하여  $set()$  함수를 호출한다.

⑧  $set$  함수의 시작점이다. ⑦번에서 전달받은  $arr$ ,  $data$ ,  $rows$ ,  $cols$ 를  $arr$ ,  $data$ ,  $rows$ ,  $cols$ 가 받는다.

⑨ 정수형 반복 변수  $i$ 가 0에서 시작하여 1씩 증가하면서 9보다 작은 동안 ⑩번을 반복 수행한다.

⑩  $data[i]$ 의 값을 ' $arr[(i + 1) / rows] \% rows[(i + 1) \% cols]$ '의 위치에 저장한다.  $set()$  함수가 종료되었으므로, 함수를 호출했던 ⑦번의 다음 문장인 ⑪번으로 이동한다.

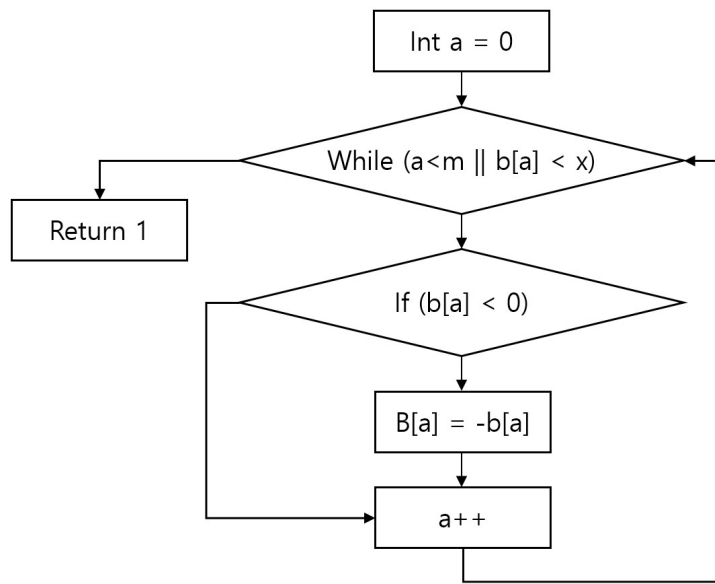
$rows$ 와  $cols$ 의 값이 3이므로, 반복문의 실행에 따른 변수들의 변화는 다음과 같다.

i	((i+1)/3)%3	(i+1)%3	data[i]	arr[i][]	data 배열 / arr 배열										
0	1/3%=0	1	5	arr[0][1]	data										
1	2/3%3=0	2	2	arr[0][2]											
2	3/3%3=1	0	7	arr[1][0]											
3	4/3%3=1	1	4	arr[1][1]											
4	5/3%3=1	2	1	arr[1][2]											
5	6/3%3=2	0	8	arr[2][0]	arr										
6	7/3%3=2	1	3	arr[2][1]											
7	8/3%3=2	2	6	arr[2][2]											
8	9/3%3=0	0	9	arr[0][0]											
9															

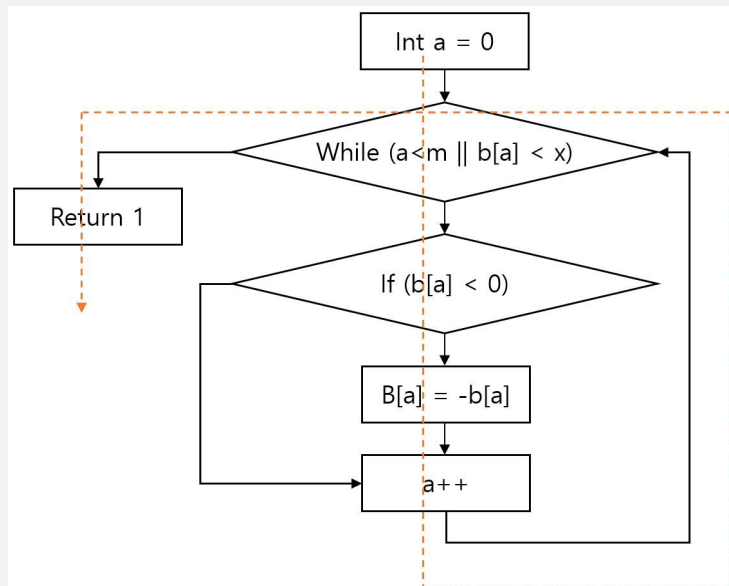
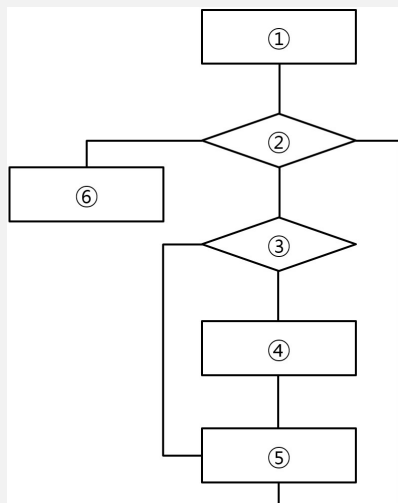
### 메모리







- 문장 커버리지를 만족하려면 모든 문장이 최소 한 번 이상 실행되어야 하므로 다음과 같은 순서로 진행되어야 합니다.



∴ 문장 커버리지 기준 테스트 케이스는 '① → ② → ③ → ④ → ⑤ → ② → ⑥' 순입니다.

#### [문제 15]

① □    ② □    ③ △    ④ ▢

#### [문제 16]

4

#### [해설]

```

public class Main {
    public static void main(String[] args) {
        ①⑩      System.out.println(calc("5"));
    }
    ⑥⑥      static int calc(int value) {
        ⑦          if (value <= 1) return value;
        ⑧          return calc(value - 1) + calc(value - 2);
    }
    ②②      static int calc(String str) {
        ③          int value = Integer.valueOf(str);
        ④          if (value <= 1) return value;
        ⑤⑨      return calc(value - 1) + calc(value - 3);
    }
}

```

⑥ 정수를 반환하는 calc(int value)를 정의한다. 인수가 int 형일 때 호출된다.

② 정수를 반환하는 calc(String str)를 정의한다. 인수가 String 형일 때 호출된다.

※ ⑥, ② 두 개의 메소드는 이름은 같지만 인수의 자료형이 다르므로 서로 다른 메소드입니다. 이렇게 이름은 같지만 인수를 받는 자료형을 달리하여 여러 기능을 정의하는 것을 오버로딩(Overloading)이라고 합니다.

모든 Java 프로그램은 반드시 main( ) 메소드에서 시작한다.

① calc("5") 메소드의 인수가 문자이므로 ②번으로 이동한다.

※ 5가 큰따옴표로 묶여 있으므로 문자로 인식됩니다.

② 정수를 반환하는 calc 함수의 시작점이다. ①번에서 전달받은 "5"를 문자열 변수 str이 받는다.

③ 정수형 변수 value를 선언하고, 정수로 변환된 숫자 5를 저장한다.

• Integer.valueOf( ) : 문자열을 정수형 객체로 변환함

④ value가 1보다 작거나 같으면 value를 반환하고, 아니면 ⑤번으로 이동한다.

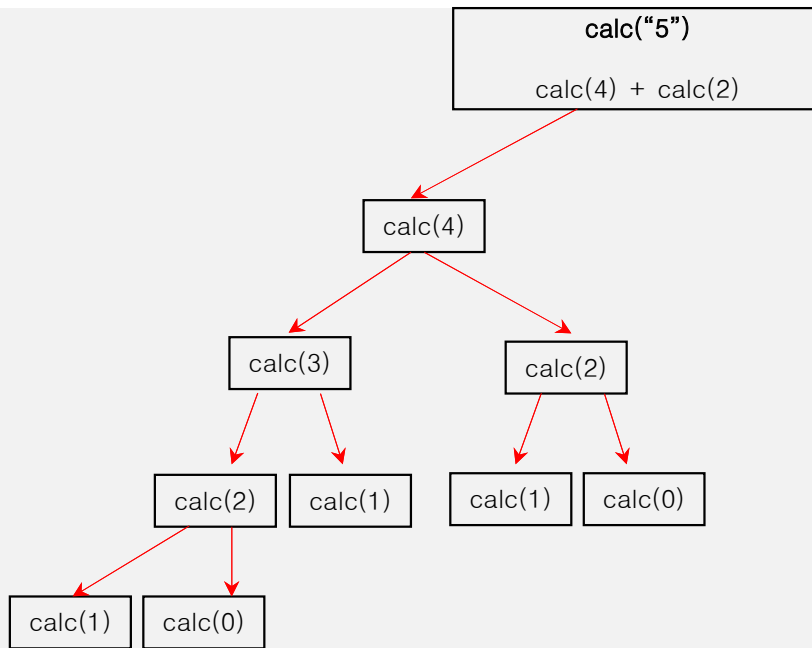
⑤ value-1을 인수로 calc( ) 메소드를 호출한 후 돌려받은 값과 value-3을 인수로 calc( ) 메소드를 호출한 후 돌려받은 값을 합한 값을 함수를 호출했던 ①번으로 반환한다. value는 5이므로 calc(4)와 calc(2)가 호출되는데, calc(4)와 calc(2) 메소드의 인수가 정수이므로 인수가 정수형일 때 호출되는 calc(int value) 메서드가 호출된다. ⑥번으로 이동한다.

⑥ 정수를 반환하는 calc 함수의 시작점이다. ⑤번에서 전달받은 4를 value가 받는다.

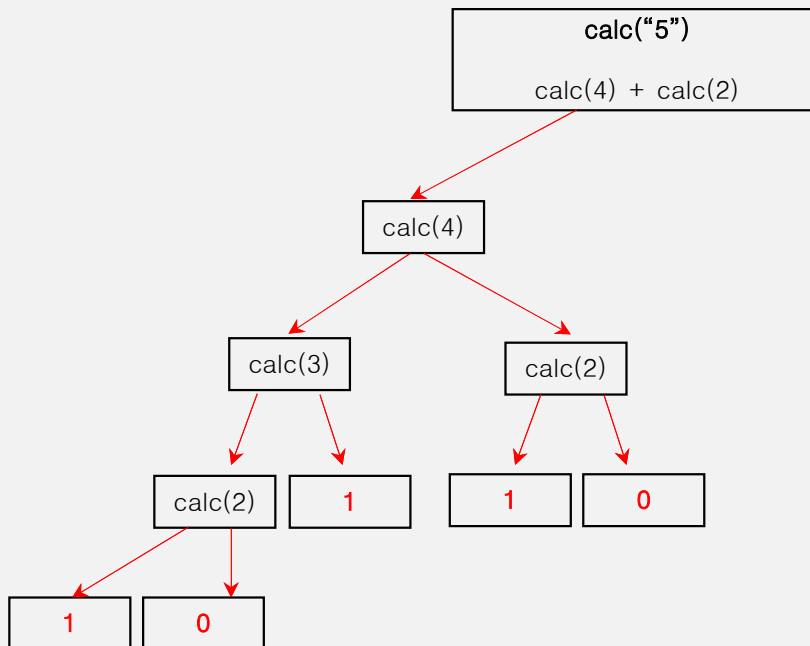
⑦ value가 1보다 작거나 같으면 value를 반환하고, 아니면 ⑧번으로 이동한다.

⑧ value-1을 인수로 calc( ) 메소드를 호출한 후 돌려받은 값과 value-2를 인수로 calc( ) 메소드를 호출한 후 돌려받은 값을 합한 값을 함수를 호출했던 ⑨번으로 반환한다. 먼저 calc(3)를 호출한다.

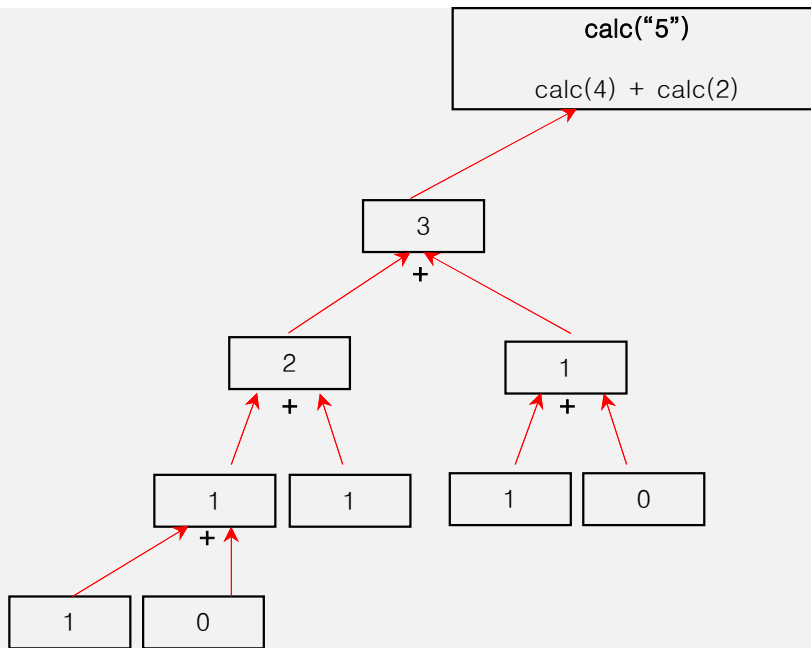
• ⑥~⑧번에 해당하는 재귀호출문은 다음과 같이 표현할 수 있다.



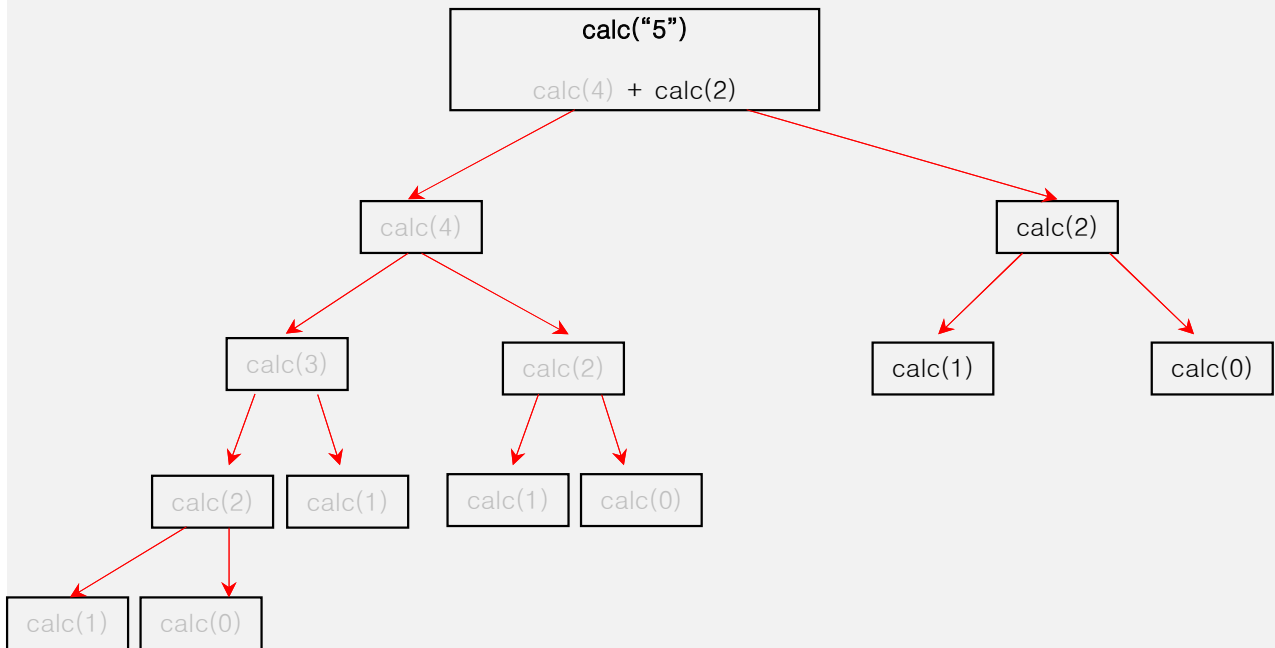
· ⑦번에 의해 인수가 1 이하인 경우는 인수의 값을 그대로 반환하므로 이를 반영하면 다음과 같다.



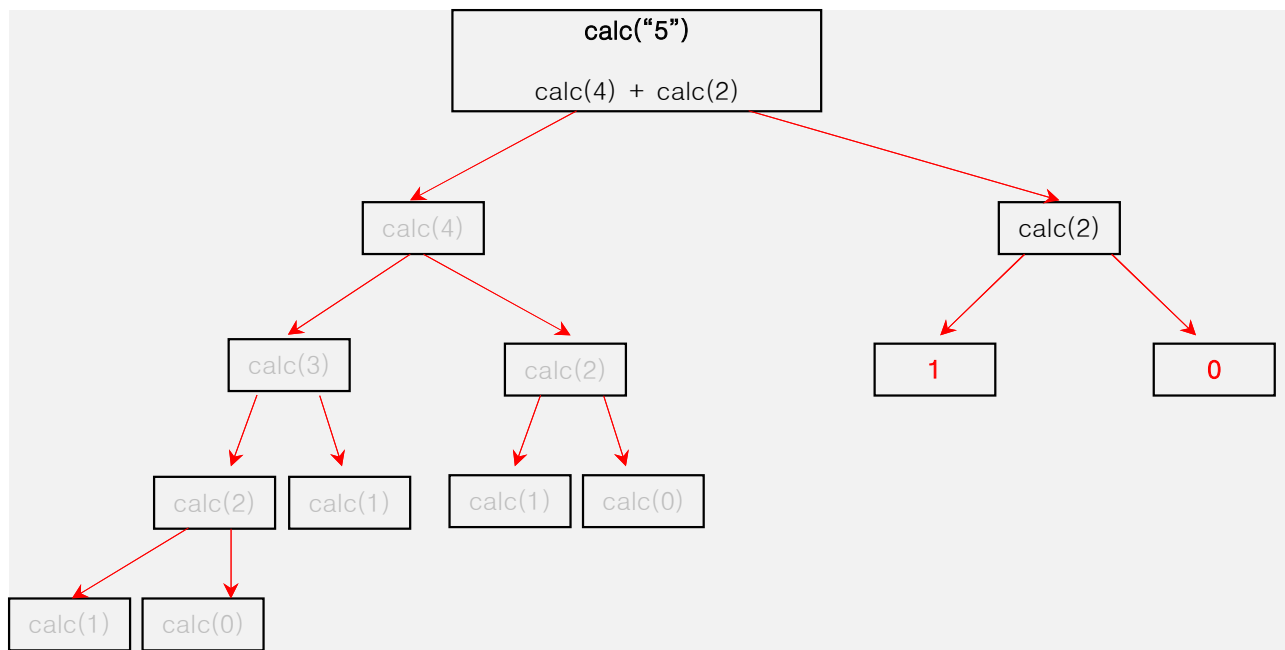
· 값이 반환되어 적용되는 과정은 다음과 같다.



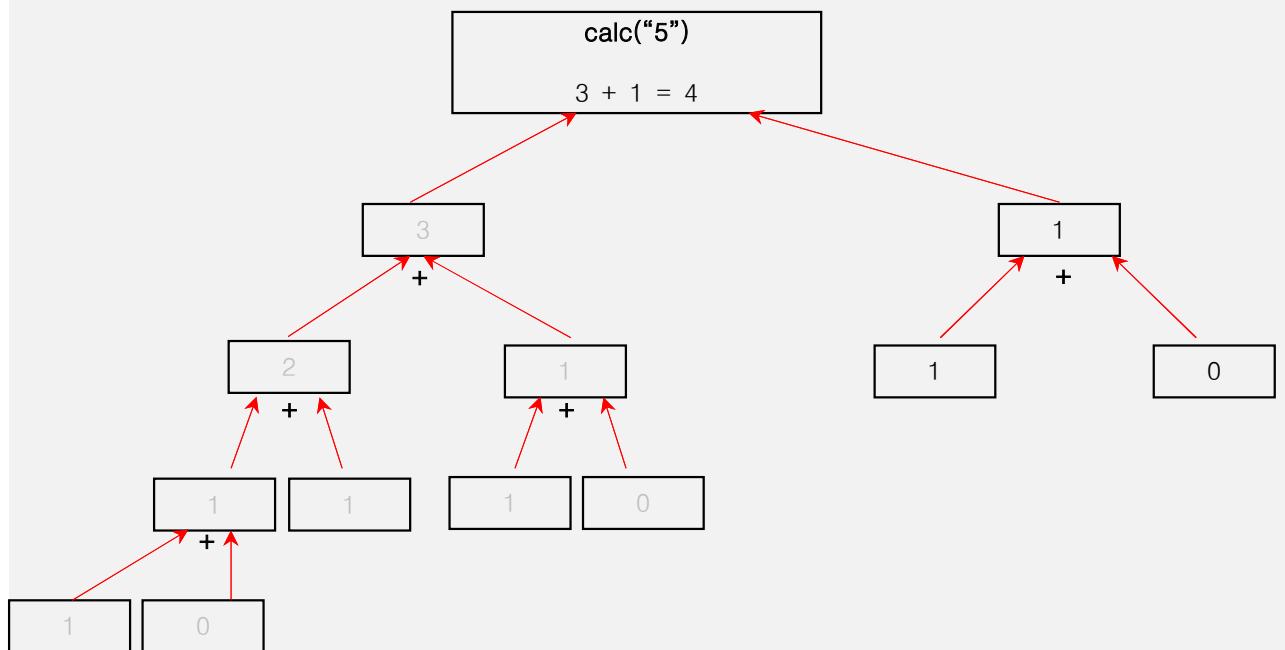
· calc(4) 호출에 이어서 calc(2) 호출에 대한 재귀호출문은 다음과 같이 표현할 수 있다.



· ⑦번에 의해 인수가 1 이하인 경우는 인수의 값을 그대로 반환하므로 이를 반영하면 다음과 같다.



· 값이 반환되어 적용되는 과정은 다음과 같다.



⑨ ⑧번에서 돌려받은 값  $4(3+1)$ 를 ⑩번으로 반환한다.

⑩ ⑨번에서 돌려받은 값 4를 출력한다.

결과 **4**

[문제 17]

908

[해설]

```

#include <stdio.h>

typedef struct {                // 구조체를 정의한다.
    char* name;                // 문자형 포인터 변수 name을 선언한다.
    int score[3];              // 3개의 요소를 갖는 정수형 배열 score를 선언한다.
} Student;                    // 구조체의 이름은 Student이다.

❸ int dec(int enc) {
❹     return enc & 0xA5;
❺ }

❻ int sum(Student* p) {
❼❿     return dec(p->score[0]) + dec(p->score[1]) + dec(p->score[2]);
}

int main( ) {
❶     Student s[2] = { "Kim", {0xA0, 0xA5, 0xDB}, "Lee", {0xA0, 0xED, 0x81} };
❷     Student* p = s;
❸     int result = 0;
❹     for (int i = 0; i < 2; i++) {
❺         result += sum(&s[i]);
❻     }
❼     printf("%d", result);
❽     return 0;
}

```

모든 C 언어 프로그램은 반드시 main( ) 함수에서 시작한다.

❶ Student 구조체 배열 s를 선언하고, 다음과 같이 초기화한다.

#### 메모리

배열 s	char* name	score		
		score[0]	score[1]	score[2]
s[0]	Kim	0xA0	0xA5	0xDB
s[1]	Lee	0xA0	0xED	0x81

※ 0x는 16진수를 의미합니다. 2진수로 표현하면 다음과 같습니다.

	char* name	score		
		score[0]	score[1]	score[2]
s[0]	Kim	10100000	10100101	11011011
s[1]	Lee	10100000	11101101	10000001

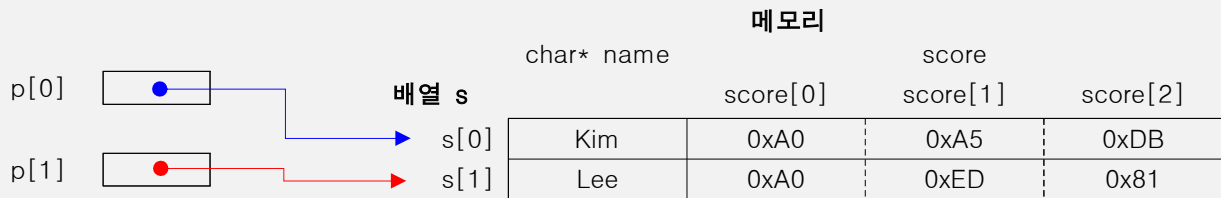
❷ Student 구조체 포인터 변수 p를 선언하고 배열 s의 시작 주소를 저장한다.

❸ 정수형 변수 result를 선언하고 0으로 초기화한다.

❹ 반복 변수 i가 0에서 시작하여 1씩 증가하면서 2보다 작은 동안 ❺번을 반복 수행한다.

❺ s[i]의 주소를 인수로 sum( ) 함수를 호출한다. ❻번으로 이동한다.

❻ 정수를 반환하는 sum( ) 함수의 시작점이다. ❺번에서 전달받은 s[i]의 주소를 Student 구조체 포인터 변수 p가 받는다.



⑦ p가 가리키는 곳의 score[0] 값인 0xA0을 인수로 dec( ) 함수를 호출한 후 돌려받은 값과 p가 가리키는 곳의 score[1] 값인 0xA5를 인수로 dec( ) 함수를 호출한 후 돌려받은 값과 p가 가리키는 곳의 score[2] 값인 0xDB를 인수로 dec( ) 함수를 호출한 후 돌려받은 값을 합한 값을 함수를 호출했던 ⑩번으로 반환한다. 먼저 dec(0xA0)을 호출한다. ⑧번으로 이동한다.

⑧ 정수를 반환하는 dec( ) 함수의 시작점이다. ⑦번에서 전달받은 0xA0을 enc가 받는다.

⑨ enc와 0xA5를 &(비트 and) 연산한 결과를 함수를 호출했던 ⑩번으로 반환한다.

⑩ dec( ) 함수를 호출한 후 돌려받은 값을 합한 값을 함수를 호출했던 ⑤번으로 반환한다. 반환받은 값은 result에 누적한다.

반복문의 실행에 따른 변수들의 변화와 ④~⑩ 과정을 수행한 후 반환되는 값은 다음과 같다.

main() 함수	dec() 함수			
i	enc = score[0]	enc = score[1]	enc = score[2]	반환 값
0	0xA0 = 10100000 0xA5 = 10100101 ----- & 10100000	0xA5 = 10100101 0xA5 = 10100101 ----- & 10100101	0xDB = 11011011 0xA5 = 10100101 ----- & 10000001	111000110 (10진수 : 454)
1	0xA0 = 10100000 0xA5 = 10100101 ----- & 10100000	0xED = 11101101 0xA5 = 10100101 ----- & 10100101	0x81 = 10000001 0xA5 = 10100101 ----- & 10000001	111000110 (10진수 : 454)
2				

⑪ result의 값 1110001100을 10진수로 출력한다.

결과 **908**

⑫ main( ) 함수에서의 'return 0'은 프로그램의 종료를 의미한다.

## [문제 18]

20

## [해설]

```

public class Main {
    public static void main(String[] args) {
        ①      int[] data = {3, 5, 8, 12, 17};
        ②⑦      System.out.println(func(data, 0, data.length - 1));
    }
    ③      static int func(int[] a, int st, int end) {
        ④          if (st >= end) return 0;
        ⑤          int mid = (st + end) / 2;
        ⑥          return a[mid] + Math.max(func(a, st, mid), func(a, mid + 1, end));
    }
}

```

모든 Java 프로그램은 반드시 main( ) 메소드에서 시작한다.

① 5개의 요소를 갖는 정수형 배열 data를 선언하고 초기화한다. 개수를 지정하지 않았으므로, 초기값으로 지정

된 수만큼 배열의 요소가 만들어진다.

	[0]	[1]	[2]	[3]	[4]
data	3	5	8	12	17

❷ data, 0과 data.length-1, 즉 data 배열 요소의 개수에서 1을 뺀 값인 4를 인수로 하여 func( ) 함수를 호출한다.

- **length** : 배열 클래스의 속성으로, 배열 요소의 개수가 저장되어 있음

❸ 정수를 반환하는 func( ) 함수의 시작점이다. ❷번에서 전달받은 data, 0, 4를 a, st, end가 받는다.

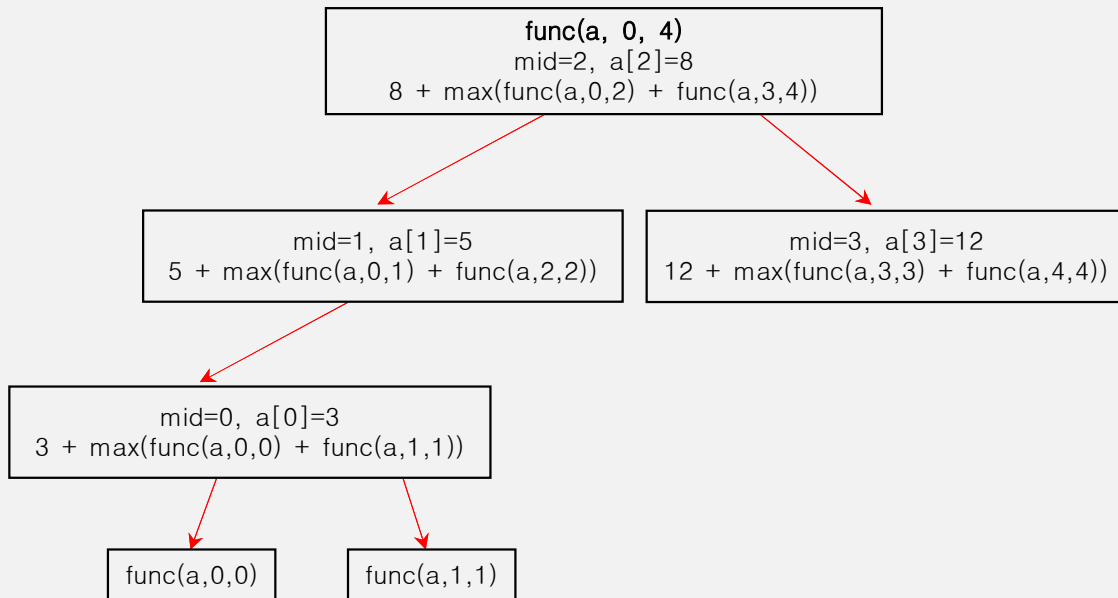
❹ st가 end보다 크거나 같으면 함수를 호출했던 곳으로 0을 반환한다. st가 end보다 작으므로 ❺번으로 이동한다.

❺ 정수형 변수 mid를 선언하고, '(st + end) / 2'의 값을 mid에 저장한다. st가 0이고 end가 4이므로 mid에는 2가 저장된다.

❻ a[mid]와 a, st, mid를 인수로 func( ) 함수를 호출한 후 돌려받은 값과 a, mid+1, end를 인수로 func( ) 함수를 호출한 후 돌려받은 값 중 큰 값을 합한 값을 함수를 호출했던 ❷번으로 반환한다.

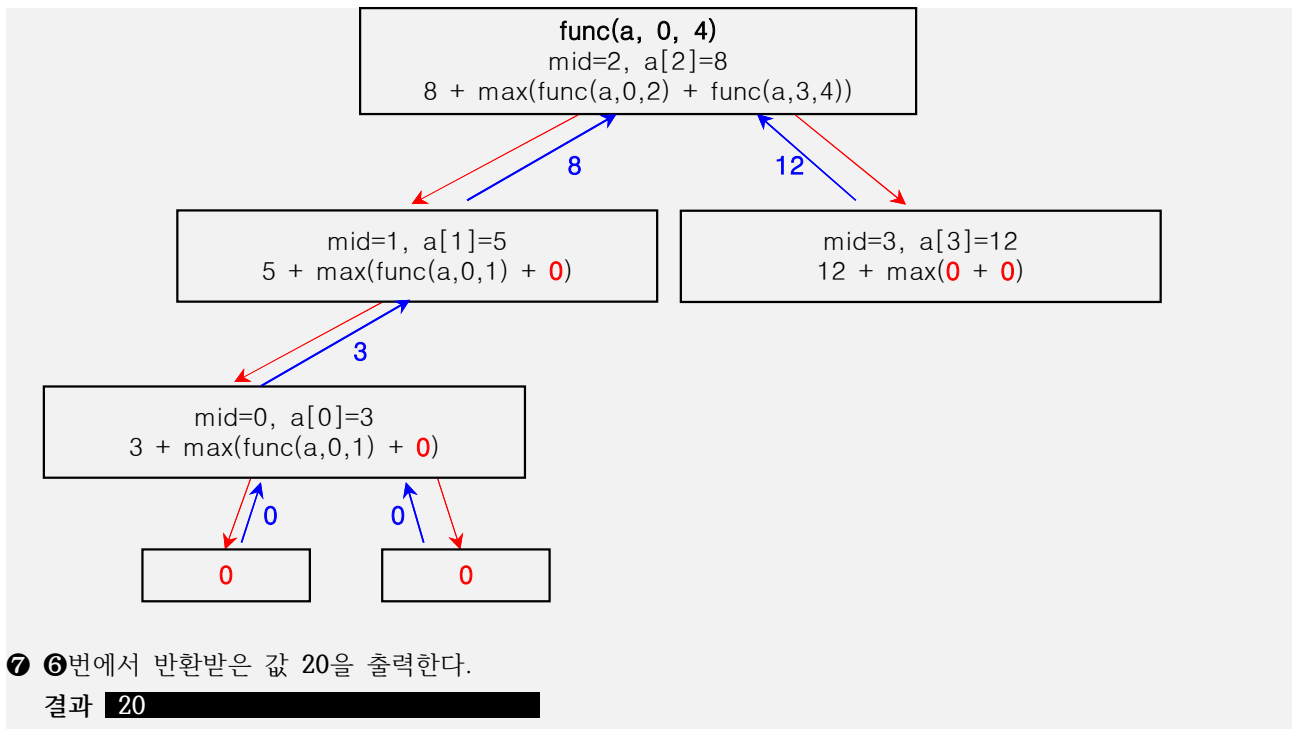
- **Math.max(인수1, 인수2)** : 인수1과 인수2 중 큰 값을 반환함

- ❸~❺번에 해당하는 재귀호출문은 다음과 같이 표현할 수 있다.



- ❹번에 의해 st가 end보다 크거나 같은 경우는 0을 반환하므로, 이를 반영하여 값이 반환되어 적용되는 과정은 다음과 같다.





[문제 19]

13

[해설]

```
class Node:                                // Node 클래스 정의부의 시작점이다.
    def __init__(self, value):              // Node 클래스의 생성자 메소드로 객체가 생성될 때 자동으로 호출된다.
        self.value = value                  // 변수 value를 선언하고 메소드가 호출될 때 전달받은 value로 초기화한다.
        self.children = []                 // children 리스트를 선언한다.
```

```
① li = [3, 5, 8, 12, 15, 18, 21]
② root = tree(li)
   print(calc(root))
```

calc( ) 메소드 정의부의 다음 줄인 15번째 줄부터 실행한다.

① 리스트 li를 선언하면서 초기값을 지정한다. 초기값으로 지정한 수만큼 리스트의 요소가 만들어진다.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
li	3	5	8	12	15	18	21

② 변수 root를 선언하고 li를 인수로 tree( ) 메소드를 호출한 후 돌려받은 값을 저장한다. ③번으로 이동한다.

```
③ def tree(li):
④     nodes = [Node(i) for i in li]
⑤     for i in range(1, len(li)):
⑥         nodes[i - 1] // 2].children.append(nodes[i])
⑦     return nodes[0]
```

③ tree( ) 메소드의 시작점이다. ②번에서 전달받은 li를 li가 받는다.

④ 리스트 nodes를 선언하고, li 리스트의 각 요소의 값을 반복 변수 i에 저장하면서 li 리스트의 요소수만큼

Node 객체를 생성하여 nodes 리스트에 저장한다.

	[0]			[1]			[2]			[3]			[4]			[5]			[6]		
nodes	Node(3)			Node(5)			Node(8)			Node(12)			Node(15)			Node(18)			Node(21)		
	value	children	...	value	children	...	value	children	...	value	children	...	value	children	...	value	children	...	value	children	...
	3			5			8			12			15			18			21		

첫 번째 반복( $i = 1$ )

⑤ 반복 변수  $i$ 에 1부터  $li$ 의 길이인 7까지 순차적으로 저장하며 ⑥번을 반복 수행한다.

※ `len()` : 문자열이나 리스트의 길이를 반환함

⑥ ( $i-1$ )을 2로 나눈 몫을 nodes의 위치 값으로 하여, 해당 위치의 children 리스트에 nodes[i]를 추가한다.  $i$ 가 1이므로 nodes[0].children.append(nodes[1])을 수행한다. 즉 nodes 리스트의 0번째 요소인 Node(3) 객체의 children 리스트에 nodes 리스트의 1번째 요소인 Node(5) 객체를 저장한다.

※ `//` : 몫을 구하는 연산자

※ `append` : 리스트의 마지막에 값을 추가하는 메소드

	[0]			[1]			[2]					
nodes	Node(3)			Node(5)			Node(8)					
	value	children[0]	children[1]	...	value	children[0]	children[1]	...	value	children[0]	children[1]	...
	3	Node(5)			5				8			

두 번째 반복( $i = 2$ )

⑥  $i$ 가 2이므로 nodes[0].children.append(nodes[2])을 수행한다.

	[0]			[1]			[2]					
nodes	Node(3)			Node(5)			Node(8)					
	value	children[0]	children[1]	...	value	children[0]	children[1]	...	value	children[0]	children[1]	...
	3	Node(5)	Node(8)		5				8			

세 번째 반복( $i = 3$ )

⑥  $i$ 가 3이므로 nodes[1].children.append(nodes[3])을 수행한다.

	[0]			[1]			[2]					
nodes	Node(3)			Node(5)			Node(8)					
	value	children[0]	children[1]	...	value	children[0]	children[1]	...	value	children[0]	children[1]	...
	3	Node(5)	Node(8)		5	Node(12)			8			

네 번째 반복( $i = 4$ )

⑥  $i$ 가 4이므로 nodes[1].children.append(nodes[4])을 수행한다.

	[0]			[1]			[2]					
nodes	Node(3)			Node(5)			Node(8)					
	value	children[0]	children[1]	...	value	children[0]	children[1]	...	value	children[0]	children[1]	...
	3	Node(5)	Node(8)		5	Node(12)	Node(15)		8			

다섯 번째 반복( $i = 5$ )

⑥  $i$ 가 5이므로 nodes[2].children.append(nodes[5])을 수행한다.

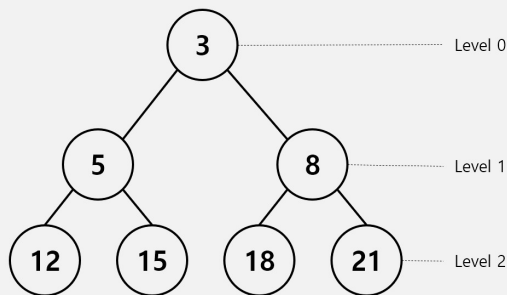
nodes	[0]				[1]				[2]				...
	Node(3)				Node(5)				Node(8)				
	value	children[0]	children[1]	...	value	children[0]	children[1]	...	value	children[0]	children[1]	...	
	3	Node(5)	Node(8)		5	Node(12)	Node(15)		8	Node(18)			

여섯 번째 반복(i = 6)

⑥ i가 6이므로 nodes[2].children.append(nodes[6])을 수행한다.

nodes	[0]				[1]				[2]				...
	Node(3)				Node(5)				Node(8)				
	value	children[0]	children[1]	...	value	children[0]	children[1]	...	value	children[0]	children[1]	...	
	3	Node(5)	Node(8)		5	Node(12)	Node(15)		8	Node(18)	Node(21)		

※ 반복문에 의해 nodes 리스트에 저장된 값을 그래프로 표현하면 다음과 같습니다.



⑦ nodes[0], 즉 가장 상위 노드인 Node(3)을 tree( ) 메소드를 호출했던 ⑧번으로 반환한다.

```

① li = [3, 5, 8, 12, 15, 18, 21]
②⑧ root = tree(li)
⑨ print(calc(root))

```

⑧ ⑦번으로부터 반환받은 nodes[0]을 root에 저장한다.

⑨ root를 인수로 calc( ) 메서드를 호출한 후 돌려받은 값을 출력한다. ⑩번으로 이동한다.

```

⑩ def calc(node, level=0):
⑪     if node is None:
        return 0
⑫     return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in node.children)

```

⑩ calc( ) 메서드의 시작점이다. ⑨번에서 전달받은 root를 node가 받고, 변수 level을 0으로 초기화한다.

⑪ node가 None이면, Calc( ) 메서드를 호출했던 곳으로 0을 반환한다. node가 None이 아니므로 ⑫번으로 이동한다.

※ None은 값이 없음을 의미하는 객체입니다.

⑫ (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in node.children)을 연산한 후 함수를 호출했던 ⑨번으로 결과를 반환한다.

(node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in node.children)

㉞

㉟

- ㉞ : level을 2로 나눈 나머지가 1이면, 즉 level이 홀수이면 node.value를 반환하고 아니면 0을 반환한다.
- ㉟ : node.children의 각 요소를 반복 변수 n에 저장하면서 node.children의 요소수만큼 calc( ) 메소드를 호출한 후 돌려받은 값을 모두 더한다.

level이 0이므로 ㉠은 0이 반환된다. node가 Node(3)이며, Node(3)의 children은 Node(5)와 Node(8)이다. 먼저 n에 Node(5)를 저장한 후 calc(Node(5), 1)을 호출한다. ㉡번으로 이동한다.

```
㉡ def calc(node, level=0):  
㉢     if node is None:  
        return 0  
㉣     return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in  
        node.children)
```

㉡ calc( ) 메서드의 시작점이다.

㉢ node가 None이 아니므로 ㉣번을 수행한다.

㉣ level이 1, 즉 홀수이므로 node.value의 값이 반환된다. node가 Node(5)이므로 node.value의 값은 5이다. Node(5)의 children은 Node(12)와 Node(15)이다. 먼저 n에 Node(12)를 저장한 후 calc(Node(12), 2)을 호출한다. ㉤번으로 이동한다.

```
㉤ def calc(node, level=0):  
㉥     if node is None:  
        return 0  
        return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in  
        node.children)
```

㉤ calc( ) 메서드의 시작점이다.

㉥ node가 Node(12)이므로 node의 children이 없다. 즉 node가 None이므로, 함수를 호출했던 ㉡번으로 0을 반환한다.

```
㉡ def calc(node, level=0):  
㉢     if node is None:  
        return 0  
㉣㉤     return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in  
        node.children)
```

㉤ ㉣번에서 Node(5)의 children 중 Node(12)를 먼저 호출한 후 돌아왔으므로 이어서 n에 Node(15)를 저장한 후 calc(Node(15), 2)를 호출한다. ㉥번을 호출한다.

```
㉥ def calc(node, level=0):  
㉦     if node is None:  
        return 0  
        return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in  
        node.children)
```

㉥ calc( ) 메서드의 시작점이다.

㉦ node가 Node(12)이므로 node의 children이 없다. 즉 node가 None이므로, 함수를 호출했던 ㉡번으로 0을 반환한다.

```

⑬ def calc(node, level=0):
⑭     if node is None:
        return 0
⑮⑱㉑ return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in
        node.children)

```

㉑ calc(Node(12), 2)와 calc(Node(15), 2)가 0을 반환하였으므로 5를 반환하면서 제어를 calc(Node(5), 1)을 호출했던 ㉑번으로 옮긴다.

(node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in node.children)

㉑

㉒

- ㉑ : 5 (calc(Node(12), 2)와 calc(Node(15), 2)를 호출할 때 node의 value는 5였으므로)
- ㉒ : 0 (㉑번과 ㉒번에서 0을 반환하였으므로)

```

⑩ def calc(node, level=0):
⑪     if node is None:
        return 0
⑫㉒ return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in
        node.children)

```

㉒ ㉑번에서 Node(3)의 children 중 Node(5)를 먼저 호출한 후 돌아왔으므로 이어서 n에 Node(8)을 저장한 후 calc(Node(8), 1)를 호출한다. ㉒번을 호출한다.

```

㉓ def calc(node, level=0):
㉔     if node is None:
        return 0
㉕     return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in
        node.children)

```

㉓ calc( ) 메서드의 시작점이다.

㉔ node가 None이 아니므로 ㉕번을 수행한다.

㉕ level이 1, 즉 홀수이므로 node.value의 값이 반환된다. node가 Node(8)이므로 node.value의 값은 8이다. Node(8)의 children은 Node(18)과 Node(21)이다. 먼저 n에 Node(18)을 저장한 후 calc(Node(18), 2)을 호출한다. ㉕번으로 이동한다.

```

㉖ def calc(node, level=0):
㉗     if node is None:
        return 0
        return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in
        node.children)

```

㉖ calc( ) 메서드의 시작점이다.

㉗ node가 Node(18)이므로 node의 children이 없다. 즉 node가 None이므로, 함수를 호출했던 ㉖번으로 0을 반환한다.

```

②③ def calc(node, level=0):
②④     if node is None:
        return 0
②⑤②⑧     return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in
        node.children)

```

②⑧ ②⑤번에서 Node(8)의 children 중 Node(18)을 먼저 호출한 후 돌아왔으므로 이어서 n에 Node(21)을 저장한 후 calc(Node(21), 2)를 호출한다. ②⑨번을 호출한다.

```

②⑨ def calc(node, level=0):
③⑩     if node is None:
        return 0
        return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in
        node.children)

```

②⑨ calc( ) 메서드의 시작점이다.

③⑩ node가 Node(21)이므로 node의 children이 없다. 즉 node가 None이므로, 함수를 호출했던 ③⑩번으로 0을 반환한다.

```

⑩⑩ def calc(node, level=0):
⑩⑪     if node is None:
        return 0
⑩⑫③①     return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in
        node.children)

```

③① calc(Node(18), 2)와 calc(Node(21), 2)가 0을 반환하였으므로 8을 반환하면서 제어를 calc(Node(8), 1)을 호출했던 ③②번으로 옮긴다.

(node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in node.children)

㉟

㉠

- ㉟ : 8 (calc(Node(18), 2)와 calc(Node(21), 2)를 호출할 때 node의 value는 8이었으므로)
- ㉠ : 0 (②⑦번과 ③⑩번에서 0을 반환하였으므로)

```

⑩⑩ def calc(node, level=0):
⑩⑪     if node is None:
        return 0
⑩⑫②④③②     return (node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in
        node.children)

```

③② Calc(Node(5), 1)와 Calc(Node(8), 1)이 5와 8을 반환하였으므로 13을 반환하면서 제어를 calc(root)를 호출했던 ③③번으로 옮긴다.

(node.value if level % 2 == 1 else 0) + sum(calc(n, level + 1) for n in node.children)

㉟

㉠

- ㉟ : 0 (Calc(Node(5), 1)과 Calc(Node(8), 1)을 호출할 때 level이 0이었으므로)
- ㉠ : 13 (②②번과 ③①번에서 5와 8을 반환하였으므로)

```

❶    li = [3, 5, 8, 12, 15, 18, 21]
❷❸  root = tree(li)
❹❸  print(calc(root))

```

❸ ❷번으로부터 반환받은 13을 출력한다.

결과 13

## [문제 20]

35421

## [해설]

```

#include <stdio.h>
#include <stdlib.h>
typedef struct Data {           // Data 구조체를 정의한다.
    int value;                  // 정수형 변수 value를 선언한다.
    struct Data *next;         // Data 구조체의 포인터 변수 next를 선언한다.

```

struct Data	
int value (4Byte)	struct Data *next (4Byte)
데이터를 저장할 멤버	다음 노드의 주소를 저장할 멤버

```

} Data;

```

```

int main( ) {
❶    Data *head = NULL, *curr;
❷    for (int i = 1; i <= 5; i++)
❸        head = insert(head, i);
    head = reconnect(head, 3);
    for (curr = head; curr != NULL; curr = curr -> next)
        printf("%d", curr->value);
    return 0;
}

```

모든 C 언어 프로그램은 반드시 main( ) 함수에서 시작한다.

❶ Data 자료형 포인터 변수 head와 curr을 선언하고, head에 NULL을 저장한다.

※ NULL은 값이 없음을 의미합니다.

❷ 반복 변수 i가 1에서 시작하여 1씩 증가하면서 5보다 작거나 같은 동안 ❸번을 반복 수행한다.

❸ head와 i를 인수로 insert( ) 함수를 호출한 후 돌려받은 값은 head에 저장한다.

첫 번째 반복(i = 1)

```

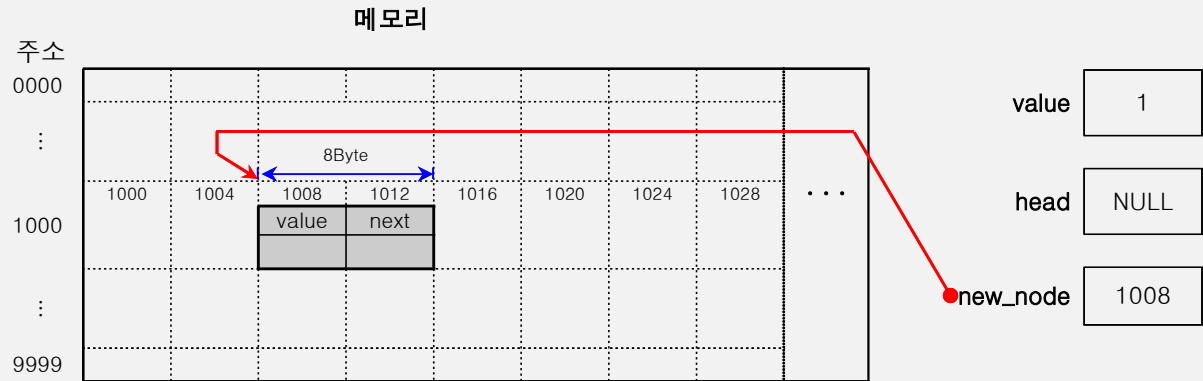
❹ Data* insert(Data* head, int value) {
❺    Data* new_node = (Data*)malloc(sizeof(Data));
❻    new_node -> value = value;
❼    new_node -> next = head;
❽    return new_node;
}

```

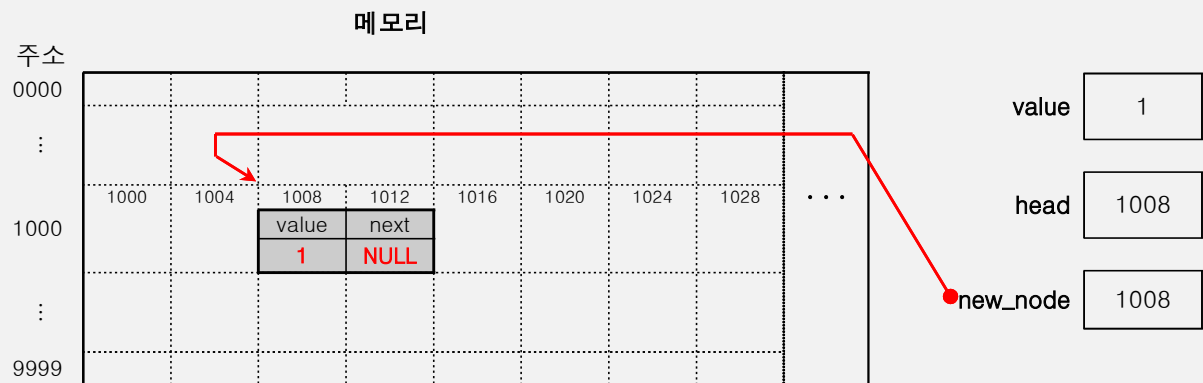
❹ Data 포인터 자료형을 반환하는 insert( ) 함수의 시작점이다. ❸번에서 전달받은 head와 i를 head와 value가 받는다.

❺ malloc 함수가 메모리에서 Data 구조체의 크기인 8Byte의 빈 영역을 찾아 할당한 다음 그 영역의 시작 주

소를 Data 자료형 포인터 변수인 new\_node에 저장한다. 이제 new\_node는 할당된 공간의 시작 주소를 가리키게 된다. (malloc 함수가 동적으로 할당하는 것이므로 여기서 지정한 주소 1008은 임의로 정한 것이며, 이해를 돕기 위해 10진수로 표현했습니다.)

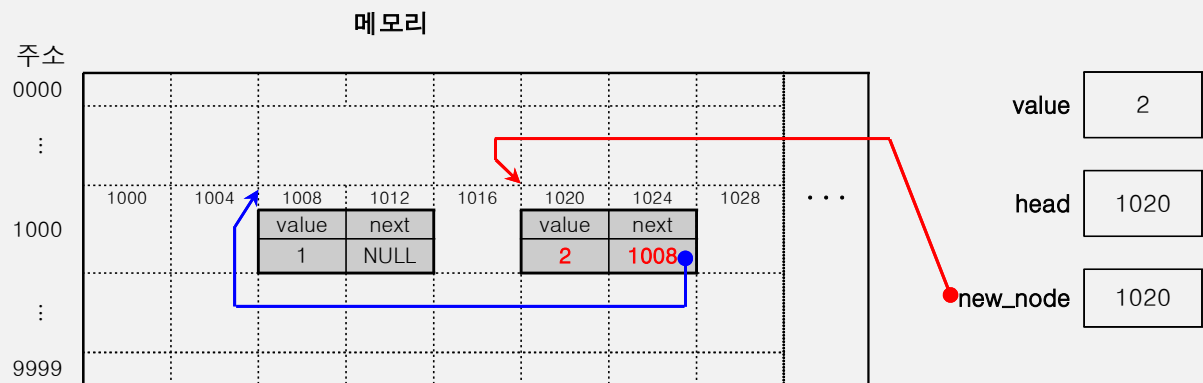


- ⑥ new\_node의 value에 변수 value의 값 1을 저장한다.
- ⑦ new\_node의 next에 변수 head의 값 NULL을 저장한다.
- ⑧ new\_node의 시작 주소를 가지고 insert( ) 함수를 호출했던 main( ) 함수로 제어를 옮긴다.



#### 두 번째 반복(i = 2)

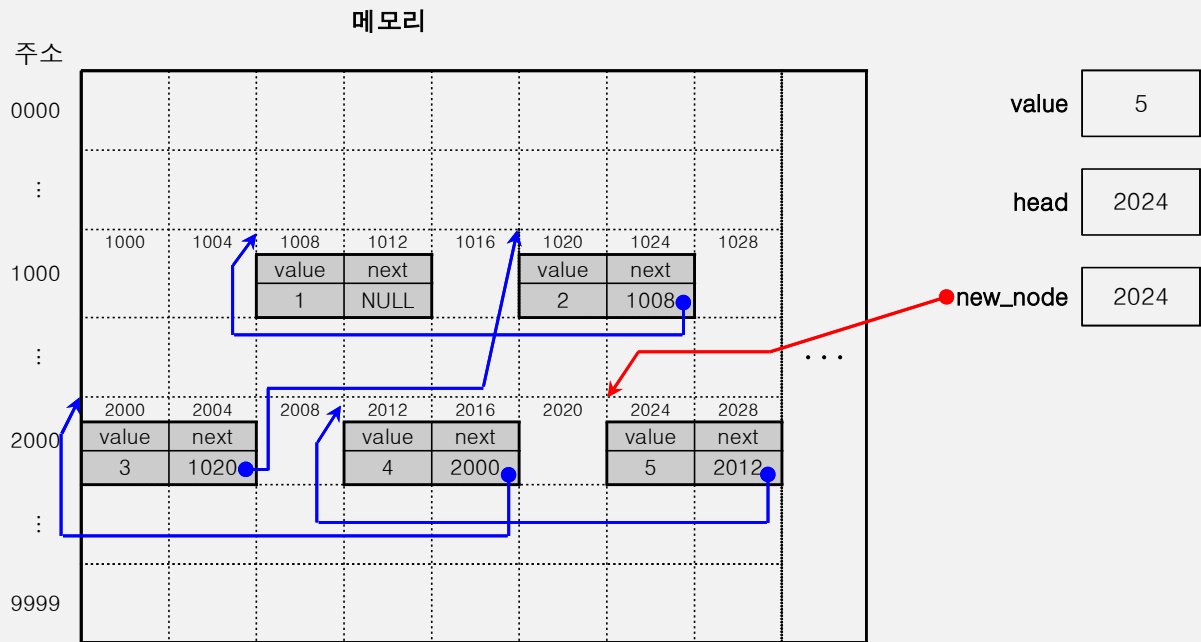
- ④ Data 포인터 자료형을 반환하는 insert( ) 함수의 시작점이다. ③번에서 전달받은 head와 i를 head와 value가 받는다.
- ⑤ malloc 함수가 메모리에서 Data 구조체의 크기인 8Byte의 빈 영역을 찾아 할당한 다음 그 영역의 시작 주소를 Data 자료형 포인터 변수인 new\_node에 저장한다.
- ⑥ new\_node의 value에 변수 value의 값 2를 저장한다.
- ⑦ new\_node의 next에 변수 head의 값 1008을 저장한다. next는 다음 노드의 주소를 가리키는 것이므로, 이제 이 노드의 다음 노드는 value 값이 1인 노드이다.
- ⑧ new\_node의 시작 주소를 insert( ) 함수를 호출했던 main( ) 함수로 제어를 옮긴다.



※ 위의 과정을 통해 반복 변수 i 값을 value로 하는 노드가 매 반복 과정에서 새로 생성되며, 생성된 각 노드의



next는 이전 노드의 head를 갖게 되므로, 다음과 같이 표현할 수 있습니다.



※ 마지막 반복문에서 반환되는 값은 new\_node의 시작 주소, 즉 value가 5인 노드의 시작 주소인 2024가 반환됩니다.

```
int main( ) {
    ❶ Data *head = NULL, *curr;
    ❷ for (int i = 1; i <= 5; i++)
    ❸     head = insert(head, i);
    ❹ head = reconnect(head, 3);
    for (curr = head; curr != NULL; curr = curr -> next)
        printf("%d", curr->value);
    return 0;
}
```

❹ head와 3을 인수로 reconnect( ) 함수를 호출한 후 돌려받은 값을 head에 저장한다. 반복문이 종료될 때의 head에는 value가 5인 노드의 시작 주소가 저장되어 있었으므로, reconnect(2024, 3)을 호출한다. ❿번으로 이동한다.

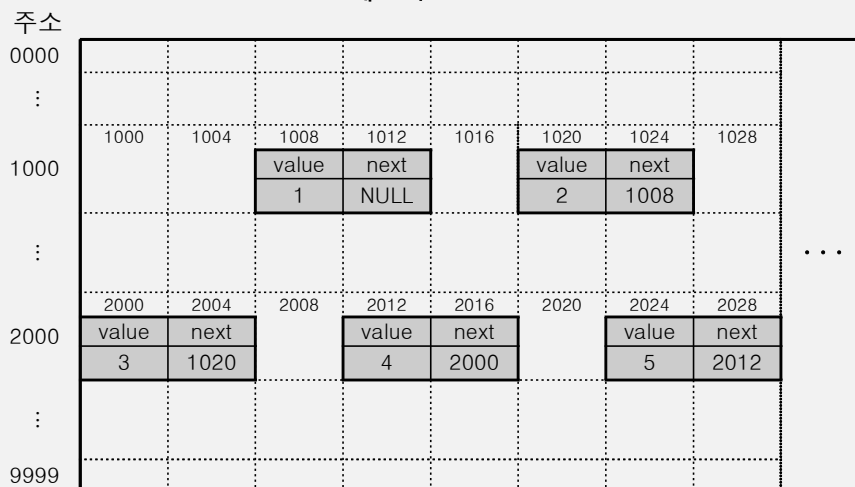
```

10 Data* reconnect(Data* head, int value) {
11     if (head == NULL || head -> value == value) return head;
12     Data *prev = NULL, *curr = head;
13     while (curr != NULL && curr -> value != value) {
14         prev = curr;
15         curr = curr -> next;
16     }
17     if (curr != NULL && prev != NULL) {
18         prev -> next = curr -> next;
19         curr -> next = head;
20         head = curr;
21     }
22     return head;
23 }

```

- ⑩ Data 포인터 자료형을 반환하는 reconnect( ) 함수의 시작점이다. ⑨번에서 전달받은 head와 3을 head와 value가 받는다.
- ⑪ head가 NULL이거나 head의 value가 value의 값과 같으면 함수를 호출했던 곳으로 head를 반환한다. 현재 head는 NULL이 아니고 head의 value는 5이므로 조건을 만족하지 않아 ⑫번으로 이동한다.
- ⑫ Data 자료형 포인터 변수 prev와 curr을 선언하고, head에 NULL을 curr에는 head를 저장한다.
- ⑬ curr가 NULL이 아니고 curr의 value가 value, 즉 3이 아니면 ⑭~⑮번을 수행한다.
- ⑭ prev에 curr을 저장한다.
- ⑮ curr의 next를 curr에 저장한다.

### 메모리



head 2024

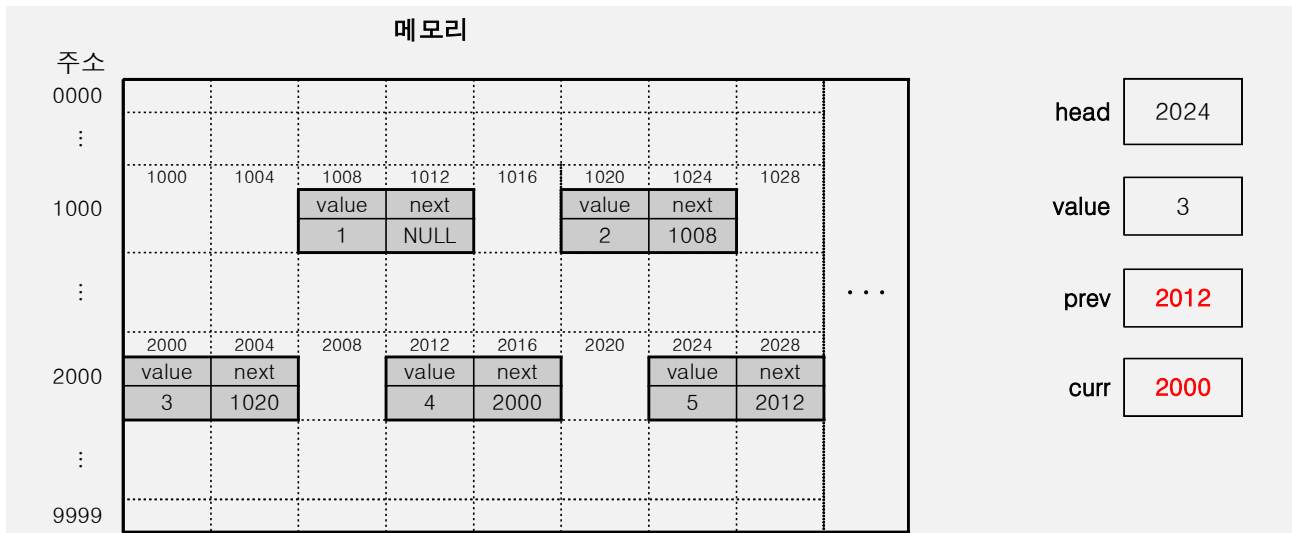
value 3

prev NULL

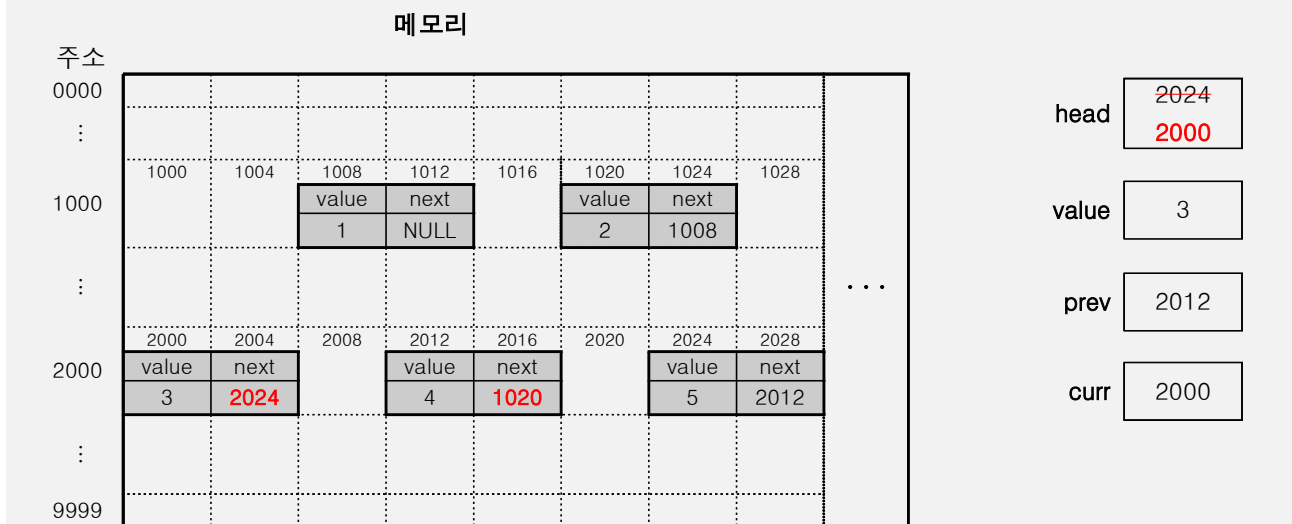
curr 2024

반복문의 실행에 따른 변수들의 변화는 다음과 같다.

반복	value	prev	curr	curr->value	조건 만족
초기	3	NULL	2024	5	Yes
1회	2024	2024	2012	4	Yes
2회	2012	2012	2000	3	No



- ⑮ curr와 prev가 NULL이 아니면 ⑰~⑲번을 수행한다. curr은 2000이고 prev는 2012로, NULL이 아니므로 ⑰번으로 이동한다.
- ⑰ curr의 next인 1020을 prev의 next에 저장한다.
- ⑱ head를 curr의 next에 저장한다.
- ⑲ curr을 head에 저장한다.



- ⑳ head를 reconnect( ) 함수를 호출했던 main( ) 함수의 ㉑번으로 반환한다.

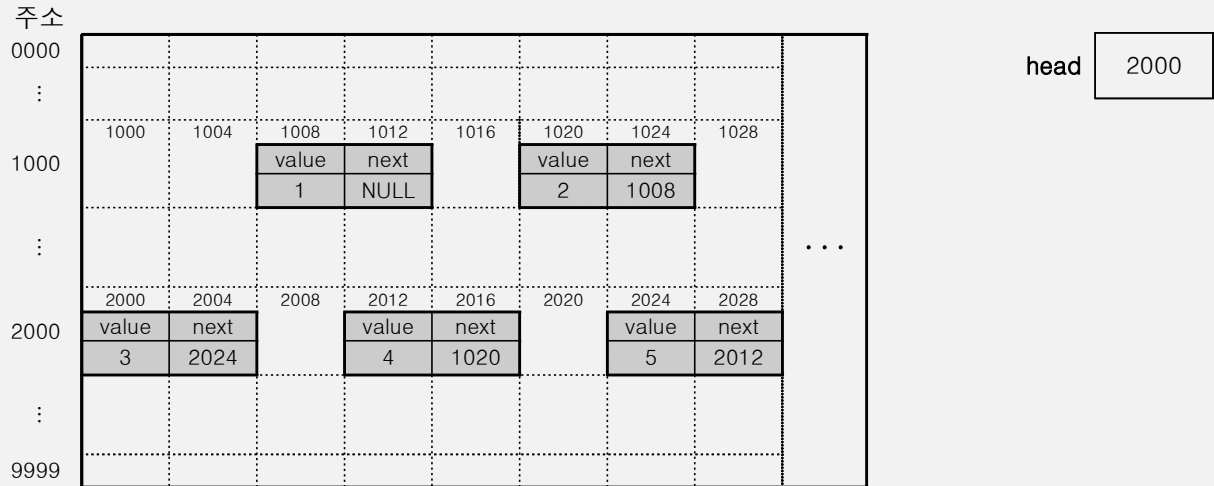
```

int main( ) {
  ① Data *head = NULL, *curr;
  ② for (int i = 1; i <= 5; i++)
  ③     head = insert(head, i);
  ⑨㉑ head = reconnect(head, 3);
  ㉒ for (curr = head; curr != NULL; curr = curr -> next)
  ㉓     printf("%d", curr->value);
  ㉔ return 0;
}

```

- ㉑ ㉒번으로부터 반환받은 2000을 head에 저장한다.
- ㉒ curr가 head에서 시작하여 curr의 next를 curr에 저장하면서 NULL이 아닌 동안 ㉓번을 반복한다.

## 메모리



반복문의 실행에 따른 변수들의 변화는 다음과 같다.

curr	curr->next	curr->value	출력
2000	2024	3	<b>3</b>
2024	2012	5	<b>35</b>
2012	1020	4	<b>354</b>
1020	1008	2	<b>3542</b>
1008	NULL	1	<b>35421</b>
NULL			

결과 35421

㉔ main( ) 함수에서의 'return 0'은 프로그램의 종료를 의미한다.