3. 화이트박스 시험

1. 개요

화이트박스시험은 **원시코드**를 **눈**으로 직접 보면서 모듈구조를 시험한다.
화이트박스시험은 모듈의 논리구조를 조직적으로 검사하는 시험이다.**(구조시험)**
원시코드 흐름은 잘 관찰할 수 있으나 고객의 요구사항이 빠진 것은 찾기 어렵다.

〈1에서 10까지 정수 합 구하기〉

```
void main(void)
{
  int k, hap = 0;
  for(k=1; k<10; k++)        //오류 : 조건 K<10가 잘못되어 있음
   hap = hap + k;            //hap = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 = 45
  printf("%d\n", hap);       //출력 : 45
}
```

↓

↓ **오류 수정(화이트박스 시험으로)**

↓ **원시코드를 눈으로 보면서 오류를 수정한다.**

↓

〈1에서 10까지 정수 합 구하기〉

```
void main(void)
{
  int k, hap = 0;
  for(k=1; k<=10; k++)       //오류 수정 : 조건 K<=10으로 수정함
   hap = hap + k;            //hap = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55
  printf("%d\n", hap);       //출력 : 55
}
```

- 원시코드의 알고리즘 구조를 **논리구조**라고 한다.
- **화이트박스 시험**을 논리구조 시험이라 하는데, 간단하게 **구조시험**이라고 한다.

2. 화이트박스 시험 종류

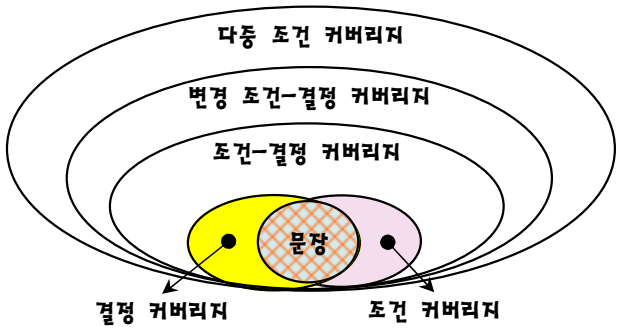
화이트박스 시험 종류는 다음과 같다.

〈화이트박스 시험〉

문장시험(문장 커버리지), 결정시험(결정 커버리지), 조건시험(조건 커버리지)
조건-결정 시험(조건-결정 커버리지)
경로시험(경로 커버리지), 기초경로시험(기본경로 커버리지)
변경 조건-결정 커버리지(modified condition-decision coverage, MC/DC)
다중 조건 커버리지(multiple condition coverage, MCC) 등

- 문장시험을 문장 커버리지(coverage)라고도 한다.
- 즉, 시험을 커버리지(coverage)라고 표현하는 것이다.
- 영어 단어 **coverage**는 **범위, 보장** 등의 의미를 가지고 있다.
- 원시코드를 시험할 때는 범위가 중요하고, 시험 결과 정확성이 보장되어야 할 것이다.

// 화이트박스 시험 포함 관계



- **결정** 커버리지를 100% 만족하면 **문장** 커버리지를 100% 만족한다.
- **조건-결정** 커버리지를 100% 만족하면 **결정** 커버리지를 100% 만족한다.
- **조건-결정** 커버리지를 100% 만족하면 **조건** 커버리지를 100% 만족한다.

- **조건** 커버리지를 100% 만족해도 **결정** 커버리지를 100% 만족하는 것은 **아니다.**
- **조건** 커버리지를 100% 만족해도 **문장** 커버리지를 100% 만족하는 것은 **아니다.**

- 일반적으로, 커버리지 포함 관계를 위의 그림처럼 설명하지만
- 엄격하게 말하면, 모든 커버리지가 100% 위의 그림과 같은 포함 관계를 가지는 것은 **아니다**.
- 그리고, 각 시험은 독립적이지 않고, **중복된 개념**이다.

① 문장 커버리지(statement coverage) / 라인 커버리지(line coverage) - SC

- 문장 커버리지는 모든 **문장**들이 적어도 한번 실행되도록 테스트 한다.
- 문장 커버리지는 가장 약한 수준의 커버리지 검증 기준이다.

〈문장 커버리지 분석〉

```
int test(int x)
{
  int a = 5;          //문장 1
  if(a > 3)           //문장 2
    a++;              //문장 3
  return a + x;       //문장 4
}
```

- 주어진 코드는 문장 4개로 구성된 함수이다.

- 4개의 문장이 모두 수행되도록 검증하면 문장 커버리지가 100% 만족된다. 라고 한다.

a = 5	• a = 5이면, 조건은 참이므로 모든 문장이 수행된다. ↓ 4 / 4 * 100 = 100(%) //문장 커버리지 100% 만족 ↓ • a=5는 문장 커버리지를 만족하는 **테스트 케이스**가 된다.
a = 2	• a = 2이면, 조건은 거짓이므로 모든 문장이 수행되지 않는다. ↓**a++;은 수행되지 않음** 3 / 4 * 100 = 75(%) //문장 커버리지 75% 만족

- 문장 커버리지 만족 비율은 원시코드 라인 수를 기준으로 수행된 라인 수의 비율이다.
- 문장 커버리지 만족 비율(%) = (수행된 라인 수 / 전체 라인 수) * 100

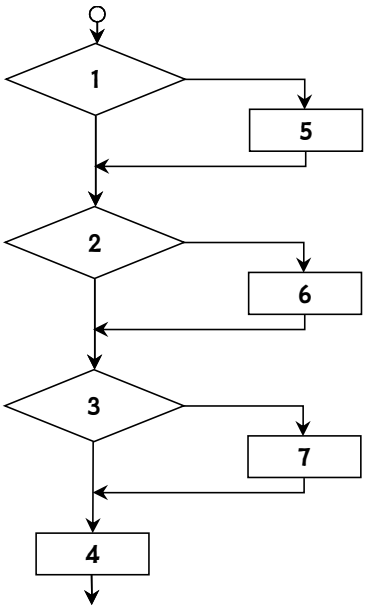
- 주어진 코드에서 문장 커버리지를 100% 만족시키는 **테스트 케이스는 최소 1개**가 된다.

탐구

독립적인 모든 경로

어떤 함수에 n개의 결정(decision, 분기)이 있으면, **2^n개의 독립적인 경로**가 존재한다.
만약, 어떤 함수에 결정문이 3개 있으면, **2 × 2 × 2 = 8**(개)의 경로가 존재한다.

〈결정문이 3개일 때, 독립적인 모든 경로〉



[의사코드]

```
void test(int a)
{
  if(a == 1)    //결정문 1
    print 5;
  if(a == 2)    //결정문 2
    print 6;
  if(a == 3)    //결정문 3
    print 7;
  print 4;
}
```

↓
↓ 독립적인 모든 경로, **2 × 2 × 2 = 8**(개)
↓

① 1, 2, 3, 4
② 1, 5, 2, 3, 4
③ 1, 2, 6, 3, 4
④ 1, 2, 3, 7, 4
⑤ 1, 5, 2, 6, 3, 4
⑥ 1, 5, 2, 3, 7, 4
⑦ 1, 2, 6, 3, 7, 4
⑧ 1, 5, 2, 6, 3, 7, 4

② 결정 커버리지(decision coverage) / 분기 커버리지(branch coverage) - DC

- 각 **결정문**이 적어도 **한번은 참, 한번은 거짓**이 실행되도록 테스트 한다.**(참 또는 거짓)**
- 각 **결정문**의 내부 조건 각각이 아닌 해당 결정문의 **전체 결과가 참/거짓**이면 된다.

〈결정 커버리지 분석〉

```
int test(int b)
{
  int a = 150;
  if(a > 100)          //결정문(분기문) 1
    a++;
  if(b > 200)          //결정문(분기문) 2
    b++;
  return a + b;
}
```

- 주어진 코드는 결정문이 2개로 구성된 함수이다.

- 결정문이 2개이므로 분기는 총 4가지가 된다.(2×2=4)
- 만약, 어떤 함수에 n개의 결정(decision, 분기)이 있으면, **2^n개의 경로**가 존재한다.
- 어떤 함수에 결정문이 3개 있으면, **2 × 2 × 2 = 8**(개)의 경로가 존재한다.

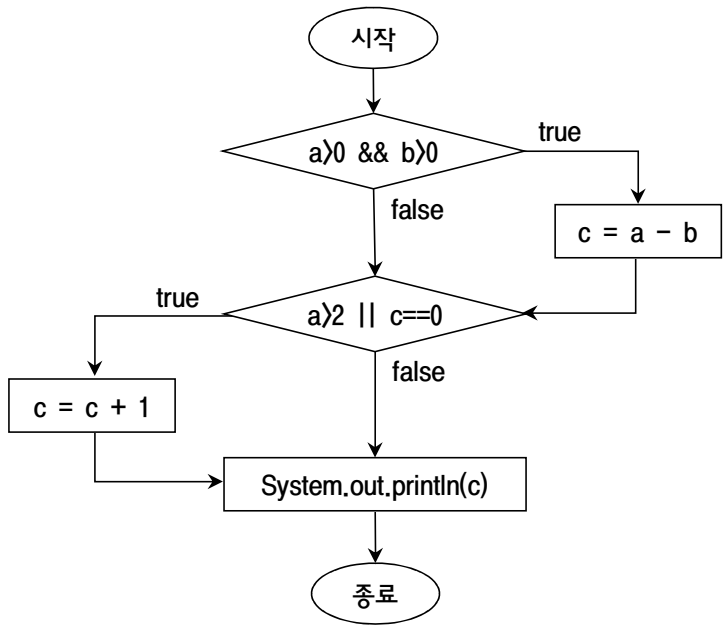
- 만약, a = 150이고, b = 150이면 분기는 2가지만 수행된다.
  ↓
- 검증 비율 = (수행된 분기 수 / 전체 분기 수) * 100 = 2 / 4 * 100 = 50(%)
  ↓결정 커버리지를 100% 만족시키기 위해서는
- 또 다른 테스트 케이스가 필요하다는 것이다.

- 결정문이 2개이면, 결정 커버리지를 100% 만족시키는 **테스트 케이스**는 **최소 2개**가 필요

테스트 케이스	if(a 〉 100)	if(b 〉 200)
a = 300, b = 300;	참	참
a = 100, b = 100;	거짓	거짓

예제	다음 그래프는 Java 프로그램에 대한 제어흐름을 나타낸 것이다. 다음에 주어진 테스트 케이스에 대해서 분기 커버리지 만족 여부를 논하시오.

테스트 케이스	
a=2, b=1, c=0	a=1, b=0, c=0



〈분기 커버리지(branch coverage)〉

- 각 결정문이 적어도 한번은 참, 한번은 거짓이 실행되도록 테스트 한다.**(참 또는 거짓)**
- 각 결정문의 내부 조건 각각이 아닌 해당 결정문의 **전체 결과가 참/거짓**이면 된다.

↓**분기 커버리지 만족 여부 분석**

결정문 (분기문)	테스트 케이스	
	a=2, b=1, c=0	a=1, b=0, c=0
a>0 && b>0	참	거짓
	c = a - b c = 2 - 1 c = 1	**변수값 변경 없음**
	a=2, b=1, c=1	a=1, b=0, c=0
a>2 || c==0	거짓	참

↳ 실행 과정에서 **c=1**로 변경되어 결정문은 거짓이 된다.

결론 : 각 결정문이 적어도 한번은 **참**, 한번은 **거짓**이 되므로 분기 커버리지를 **만족**한다.

③ 조건 커버리지(condition coverage) - CC

- 조건 커버리지는 결정문 내의 **각 조건이 적어도 한번은 참, 한번은 거짓**이 되도록 시험한다.
- 조건 커버리지는 결정문 전체의 결과인 참, 거짓과는 관계가 없다.(주의!)
- 연산자 &&, || 전후에 사용된 각 조건이 참 또는 거짓이 되도록 테스트 한다.

〈조건 커버리지 분석〉

```
void test(int a, int b, int c)
{
  int p = 100;
  if((a > 0 && b < 20) || c > 80)          //결정문(조건문)
    p++;
  printf("%d", p);
}
```

- 주어진 코드는 결정문에 조건식이 3개 있는 경우이다.

```
if((a > 0 && b < 20) || c > 80)          //결정문(조건문)
```

↓

3개의 각 조건식이 **적어도 한번은 참, 한번은 거짓**이 되도록 시험하는 것이 조건 커버리지이다.
(조건 커버리지에서 중요한 것은 결정문 전체의 결과인 참, 거짓과는 관계가 없다)

// 조건 커버리지 만족하는 테스트 케이스 값

조건	a 〉 0		b 〈 20		c 〉 80	
테스트 케이스	a = 5	a = 0	b = 20	b = 10	c = 50	c = 90
참/거짓	참	거짓	거짓	참	거짓	참
만족여부	○		○		○	

④ 조건-결정 커버리지(condition-decision coverage) - C/DC

- 조건-결정 커버리지 = 결정 커버리지 + 조건 커버리지
- 조건-결정 커버리지는 결정 커버리지와 조건 커버리지를 포함하는 커버리지이다.
- 조건-결정 커버리지는 **모든 결정문**과 **각 조건이 참 또는 거짓**이 되도록 테스트 한다.

〈조건-결정 커버리지 분석〉

```
void test(int a, int b)
{
  int p = 100;
  if((a > 0 && b > 100)          //결정문(조건문)
    p++;
  printf("%d", p);
}
```

- 주어진 코드는 결정문에 조건식이 2개 있는 경우이다.

// 조건 커버리지 만족하는 테스트 케이스 값

조건	a 〉 0		b 〉 100	
테스트 케이스	a = 1	a = 0	b = 200	b = 100
참/거짓	참	거짓	참	거짓
만족여부	○		○	

// 결정 커버리지 만족하는 테스트 케이스 값

조건	a 〉 0 && b 〉 100	
테스트 케이스	a = 1, b = 200	a = 0, b = 200
참/거짓	참	거짓
만족여부	○	

// 결정 커버리지 만족하지 않는 테스트 케이스 값

조건	a 〉 0 && b 〉 100	
테스트 케이스	a = 0, b = 200	a = 0, b = 200
참/거짓	거짓	거짓
만족여부	×	

예제	다음 파이썬 원시코드에 대해 주어진 테스트 데이터를 이용하여 테스트를 수행한다. 조건 커버리지, 결정 커버리지, 문장 커버리지를 평가할 때 만족하는 비율(%)은?

원시코드	테스트 데이터

if(a>0 and b>5):
  a = b + 1
else:
  a = b + 2
	t1 : (a=1, b=3)
	t2 : (a=0, b=6)

[2013년 감리사 유형]

// 조건 커버리지

조건식	a 〉 0		b 〉 5	
테스트 데이터	a = 1	a = 0	b = 3	b = 6
참/거짓	참	거짓	거짓	참
만족여부	○		○	

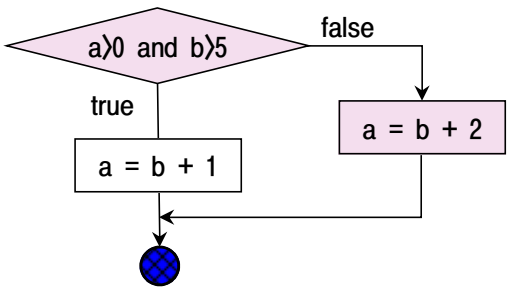
• 각 조건이 한번은 **참**, 한번은 **거짓**이므로 조건 커버리지는 100% 만족한다.

// 결정 커버리지

조건식	if(a>0 and b>5)	결과
테스트 데이터 t1	(a=1, b=3)	거짓
테스트 데이터 t2	(a=0, b=6)	거짓

• **결정문**이 **한번은 참, 한번은 거짓**이 실행되도록 테스트 한다.
• 조건이 **모두 거짓**이므로 결정 커버리지는 100% 만족하지 않는다.
• 결정 커버리지는 50% 만족한다.

// 문장 커버리지



• 문장 커버리지는 3개의 문장 중에서 2개의 문장만 검토된다.
• 해서, 문장 커버리지는 약66% 만족한다.(2/3)

기출문제 분석

1. 개발자 A는 〈명세〉에 따라 〈코드〉를 작성한 후 테스팅을 수행하였다. A는 100% 문장 커버리지를 달성하면서 동시에 프로그램의 오류를 발견할 수 있었다. A가 사용한 테스트 입력은? (단, 단축 연산(short-circuit evaluation)은 수행하지 않는다) [2019년 국가 7급]

<명 세>	두 정수를 입력 받아 두 정수 중 적어도 하나가 음수이면 두 정수의 곱을 반환하고 그렇지 않다면 두 정수의 합을 반환한다.

<코 드>	int foo(int v1, int v2)
	{
	  int v3 = v1 * v2;
	  if (v1 >= 0 || v2 >= 0) v3 = v1 + v2;
	  return v3;
	}

① (v1 = −2, v2 = 2)
② (v1 = 2, v2 = 0)
③ (v1 = −2, v2 = −2)
④ (v1 = 0, v2 = 0)

♤ **문장 커버리지**

• 문장 커버리지는 프로그램 내의 모든 문장들을 적어도 한번 이상 실행되도록 한다.

// 프로그램 분석

```
int foo(int v1, int v2)
{
  int v3 = v1 * v2;
  if (v1 >= 0 || v2 >= 0) v3 = v1 + v2;    //두 정수 중 적어도 하나가 음수이면
  return v3;
}
```

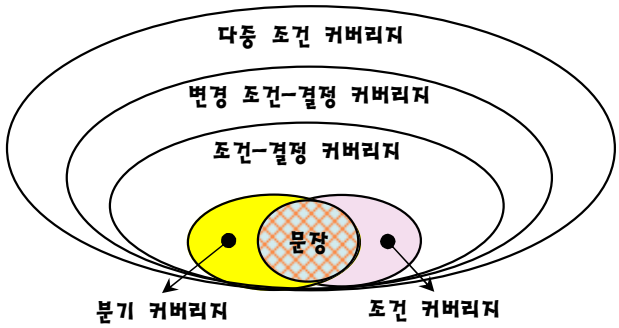
• 두 정수 중 적어도 하나가 음수이면, v3 = v1 + v2;가 실행, 두 정수의 합을 반환한다.
• A는 100% 문장 커버리지를 달성하면서 동시에 프로그램 오류를 발견할 수 있다.
• 해서, 주어진 명세를 기준으로 A가 사용한 테스트 입력은 (v1 = −2, v2 = 2)이다.

정답 : ①

2. 화이트박스 테스트 커버리지에 대한 설명으로 옳은 것은? [2022년 국가 7급]

① 문장 커버리지는 입력 데이터가 미리 정의한 유형에 적합한지를 검증하는 방법이다.
② 동일한 프로그램에 대해 분기 커버리지를 달성하면 조건 커버리지를 달성한다.
③ 동일한 프로그램에 대해 조건 커버리지를 달성하면 분기 커버리지를 달성한다.
④ 동일한 프로그램에 대해 수정 조건-분기 커버리지
(modified condition decision coverage, MCDC)를 달성하면 조건 커버리지를 달성한다.

♤ 화이트박스 테스트 커버리지 – 화이트박스 시험 포함 관계



정답 : ④

3. 다음 C 프로그램을 문장 커버리지(statement coverage)를 사용하여 테스트할 때 필요한 최소의 테스트케이스 개수는? [2015년 국가 7급]

```
int foo(int a[10], int x) {
  int i = 0;
  int count = 0;
  for (i=0; i<10; i++) if (a[i] == x) count++;
  return count;
}
```

① 1 ② 2 ③ 3 ④ 4

♤ 문장 커버리지

• 문장 커버리지는 프로그램 내의 모든 문장들을 적어도 한번 이상 실행하도록 한다.
• 최소 테스트 수 : 1개 (예 : a[0] = 1, x = 1이면 모든 문장이 적어도 한 번씩 실행)

정답 : ①

4. 결정 명령문 내의 각 조건식이 참, 거짓을 한 번 이상 갖도록 조합하여 테스트 케이스를 설계하는 방법은? [2018년 국가 9급]

① 문장 검증 기준(statement coverage)
② 조건 검증 기준(condition coverage)
③ 분기 검증 기준(branch coverage)
④ 다중 조건 검증 기준(multiple condition coverage)

♤ **화이트박스 테스트**

// 문장 커버리지 – 가장 약한 커버리지 기준

• 검사하려는 프로그램 내의 모든 문장들을 적어도 한번 이상 실행하도록 한다.

// 분기 커버리지 – 일명, 결정 커버리지(decision coverage)

• 각 분기(참 또는 거짓)를 적어도 한번 이상 실행시키는 시험이다.
• 각 분기의 내부 조건 자체가 아닌 해당 조건으로 인한 전체 결과가 참/거짓이면 된다.
• 분기 커버리지가 만족되면 문장 커버리지도 만족된다.

// 조건 커버리지 – 분기 커버리지 보다 강력

• 조건문 전체의 결과(참, 거짓)와 무관하게, 연산자 &&, || 전후에 사용된 각 조건식이 적어도 한 번은 참, 한 번은 거짓이 되도록 시험한다.
→ 조건 커버리지 테스트는 조건문 전체의 결과인 참, 거짓과는 관계가 없다.(주의!)

// 다중 조건 검증 기준(multiple condition coverage, MCC)

• MCC는 조건과 분기에 대하여 가능한 모든 조합이 실행되도록 검증한다.
• MCC가 만족되면 condition/branch coverage도 만족시킨다.

• 다음은 MCC를 만족시키는 예이다.(가능한 모든 조합이 4가지인 경우)

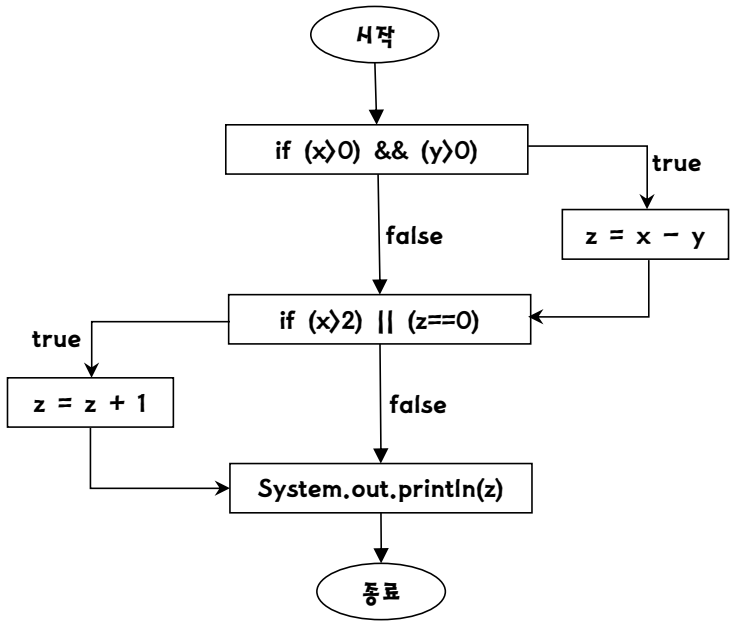
조건식	a>1	b<8	전체 조건식 결과

if (a>1 && b<8)
a++;
else
b++;
	a=2(참)	b=5(참)	참

	a=2(참)	b=9(거짓)	거짓
	a=0(거짓)	b=5(참)	거짓
	a=0(거짓)	b=9(거짓)	거짓

정답 : ②

5. 다음 제어흐름 그래프에 나타난 프로그램을 테스트할 때, 옳지 않은 것은? [2017년 국가 7급]



① 분기 커버리지를 만족하는 최소의 테스트 케이스는 2개이다.
② 기본경로의 개수는 3개이다.
③ 문장 커버리지를 만족하는 최소의 테스트 케이스는 1개이다.
④ {(x:1, y:2, z:0), (x:5, y:0, z:0)}은 분기 커버리지를 만족하지 못한다.

♤ **분기 커버리지(branch coverage) : 최소 테스트 케이스 수는 2개**

- 각 결정문이 적어도 한번은 참, 한번은 거짓이 실행되도록 테스트 한다.(참 또는 거짓)
- 각 결정문의 내부 조건 자체가 아닌 해당 조건으로 인한 전체 결과가 참/거짓이면 된다.
  → 각 if문에 대하여 참/거짓이 각각 수행되어야 100% 달성될 수 있다.

// 테스트 케이스

	(x:1, y:2, z:0)인 경우	(x:5, y:0, z:0)인 경우	분기 커버리지
if (x>0) && (y>0)	참	거짓	만족 함
if (x>2) || (z==0)	z=x-y=1-2=-1(거짓)	참	만족 함

- 테스트 케이스 {(x:1, y:2, z:0), (x:5, y:0, z:0)}은 분기 커버리지를 만족한다.
- 테스트 케이스 {(x:1, y:2, z:0)에서 참이면 z=x-y=1-2=-1로 z **값이 변하는 것**에 주의!

정답 : ④

6. 다음 C 프로그램이 조건 커버리지(condition coverage)를 100% 만족하기 위한 테스트 데이터 집합은? (단, short-circuit evaluation은 수행하지 않는다) [2012년 전산 7급]

```
---------------------------------------
void foo(int x, int y, int z)
{
  if (x > 10 && y == 10) z = 5;
  if (x == 10 || z > 3) z = z + 10;
  printf("%d", z);
}
---------------------------------------
```

① (x: 10, y: 10, z:10),
(x: 20, y: 10, z: 3)
② (x: 10, y: 20, z: 0),
(x: 20, y: 20, z: 20)
③ (x: 20, y: 10, z: 3),
(x: 20, y: 20, z: 20)
④ (x: 20, y: 10, z: 10),
(x: 10, y: 20, z: 0)

♤ 조건 커버리지를 100% 만족하기 위한 테스트 데이터 집합

- 결정문 내의 <u>각 조건이 적어도 한번은 참, 한번은 거짓</u>이 되도록 테스트 한다.
- 조건 커버리지는 결정문 전체의 결과인 참, 거짓과는 관계가 없다.(주의!)
- 연산자 &&, || 전후에 사용된 각 조건이 참 또는 거짓이 되도록 테스트 한다.

④ (x: 20, y: 10, z: 10),
(x: 10, y: 20, z: 0)

	(x: 20, y: 10, z: 10)	(x: 10, y: 20, z: 0)	조건 커버리지
if(x > 10 && y == 10)	x 〉 10은 참 y == 10은 참	x 〉 10은 거짓 y == 10은 거짓	만족 함
if(x == 10 || z > 3)	x == 10은 거짓 z 〉 3은 참	x == 10은 참 z 〉 3은 거짓	만족 함

정답 : ④

7. 다음 소스코드와 테스트 입력을 사용한 테스트 커버리지 설명으로 옳지 않은 것은? (단, 단축 연산(short-circuit evaluation)은 수행하지 않는다) [2021년 국가 7급]

문장번호	소스코드	테스트 입력

1
2
3
4
	void temp(int x, int y) {
	if(x < 0 && y >= 0)
	printf("%d",y);
	if(x >= 0)
	printf("%d",x);
	}
		T = {t1:(x=-2, y=3), t2:(x=2, y=0)}

① t2만 사용하면 문장(statement) 커버리지가 75%이다.
② T를 사용하면 문장 커버리지가 100%이다.
③ T를 사용하면 분기(branch) 커버리지가 100%이다.
④ T를 사용하면 조건(condition) 커버리지가 100%이다.

♤ **테스트 커버리지**

// 조건 커버리지(condition coverage) - CC

- 조건 커버리지는 결정문 내의 **각 조건이 적어도 한번은 참, 한번은 거짓**이 되도록 시험한다.
- 조건 커버리지는 결정문 전체의 결과인 참, 거짓과는 관계가 없다.(주의!)
- 연산자 &&, || 전후에 사용된 각 조건이 참 또는 거짓이 되도록 테스트 한다.

```
void temp(int x, int y) {
  if(x < 0 && y >= 0)        //결정문(조건문)
    printf("%d",y);
  if(x >= 0)                 //결정문(조건문)
    printf("%d",x);
}
```

조건	x < 0	y >= 0	x >= 0
테스트 t1	x = −2	y = 3	x = −2
참/거짓	참	참	거짓
테스트 t2	x = 2	y = 0	x = 2
참/거짓	거짓	참	참
만족여부	○	×	○

- T를 사용하면 조건(condition) 커버리지를 **100% 만족하지 못한다.**

정답 : ④

8. 다음 소스코드에 대해 조건(condition) 커버리지와 분기(branch) 커버리지를 각각 100 % 만족하는 테스트 케이스 집합은? (단, 단축연산(short-circuit evaluation)은 수행하지 않는다) [2023년 국가 7급]

```
if (x >= 0 || y < 4)
    z = x + y;
else
    z = x - y;
```

① {(x = −1, y = 2), (x = 1, y = 6)}
② {(x = −1, y = 2), (x = 2, y = 2)}
③ {(x = 1, y = 6), (x = −2, y = 6)}
④ {(x = 2, y = 2), (x = −2, y = 6)}

♤ **테스트 케이스 집합**

// 조건 커버리지(condition coverage) - CC

- 조건 커버리지는 결정문 내의 **각 조건이 적어도 한번은 참, 한번은 거짓**이 되도록 시험한다.
- 조건 커버리지는 결정문 전체의 결과인 참, 거짓과는 관계가 없다.(주의!)
- 연산자 &&, || 전후에 사용된 각 조건이 참 또는 거짓이 되도록 테스트 한다.

조건	x >= 0	y < 4	x >= 0	y < 4	설 명
테스트 케이스	x = 2	y = 2	x = −2	y = 6	
참/거짓	참	참	거짓	거짓	
만족여부	○		○		← 조건 커버리지 100% 만족

// 분기 커버리지(branch coverage) - DC

- 각 결정문이 적어도 한번은 참, 한번은 거짓이 실행되도록 테스트 한다.**(참 또는 거짓)**
- 각 결정문의 내부 조건 각각이 아닌 해당 결정문의 **전체 결과가 참/거짓**이면 된다.

조건	x >= 0 || y < 4		x >= 0 || y < 4		설 명
테스트 케이스	x = 2	y = 2	x = −2	y = 6	
참/거짓	참	참	거짓	거짓	
만족여부	참		거짓		← 분기 커버리지 100% 만족

④ {(x = 2, y = 2), (x = −2, y = 6)} : **조건** 커버리지와 **분기** 커버리지를 각각 **100 %** 만족

정답 : ④

9. 다음 C 프로그램에 대해 조건-결정 커버리지(condition-decision coverage)를 적용하여 테스트할 때 이를 만족하는 테스트 데이터 집합은? [2014년 국가직 7급]

```
-------------------------------------
 void example(int a, int b, int c)
 {
   int p = 10;
   if ((a > 0 && b <= 15) || c > 65)
     p = p * 2;
   else
     p = p * 1;
   printf("%d", p);
 }
-------------------------------------
```

① (a : 0, b : 20, c : 70), (a : 5, b : 10, c : 60)
② (a : 0, b : 10, c : 60), (a : 5, b : 20, c : 70)
③ (a : 5, b : 20, c : 70), (a : 5, b : 20, c : 60)
④ (a : 5, b : 10, c : 60), (a : 0, b : 10, c : 60)

♤ **조건-결정 커버리지**

- **조건-결정** 커버리지는 **결정** 커버리지와 **조건** 커버리지를 **포함**하는 커버리지이다.
  → 조건-결정 커버리지는 모든 조건과 결정에 True/False가 모두 있어야 한다.
- **결정** 커버리지는 각 분기(참 또는 거짓)를 적어도 한번 이상 실행시키는 시험이다.
  → 각 분기의 내부 조건 자체가 아닌 해당 조건으로 인한 전체 결과가 참/거짓이면 된다.
- **조건** 커버리지는 분기문 전체의 결과(참, 거짓)와 무관하게, 연산자 &&, || 전후에 사용된 각 조건식이 적어도 한 번은 참, 한 번은 거짓이 되도록 시험한다.

조건 커버리지	①	②	③	④
a > 0	F / T	F / T	T / T	T / F
b <= 15	F / T	T / F	F / F	T / T
c > 65	T / F	F / T	T / F	T / T
만족여부	O	O	×	×

결정 커버리지	①	②	③	④
(a > 0 && b <= 15) || c > 65	T / T	F / T	T / F	T / F
만족여부	×	O	O	O

- 조건과 결정, 2개를 모두 만족하는 케이스는 2번이다.

정답 : ②