1. 순환복잡도(CC)

순환복잡도는 McCabe가 정의한 **원시코드 복잡도를 정량적**으로 평가하는 방법이다.

〈McCabe의 순환복잡도〉

- McCabe의 순환복잡도(cyclometic complexity)는 CC라고도 한다.
- 일반적으로, 순환복잡도는 원시코드에 존재하는 **분기문 수로 결정**된다.
- 기본적으로 순환복잡도는 최소값 1을 가지며 분기유발 구문마다 1씩 증가한다.
- 원시코드에 1개의 if문이 있으면 순환복잡도는 1 + 1 = **2**가 되고,
- 원시코드에 **if**문 1개, **for**문 2개가 있으면 순환복잡도는 1 + 2 + 1 = **4**가 된다.
- MaCabe는 CC를 10이하로 유지할 것을 권고하였다.(1976년)
- 현재, 프로그래밍 환경은 많이 변했다. C++ 표준에는 CC를 20이하로 유지할 것을 명시

↓

↓순환복잡도 계산

↓

〈순환복잡도 계산 공식 - 3가지〉

① **그래프에서 영역 수를 이용** : CC = 그래프의 폐쇄 영역 수 + 1

② **그래프에서 노드와 간선수를 이용** : CC = 간선수(화살표수) - 노드수 + 2

③ **원시코드의 결정문 수를 이용** : CC = 원자적 조건의 수 + 1

- 원자적 조건은 분기문(결정문)에서 참과 거짓으로 판별되는 조건을 의미한다.

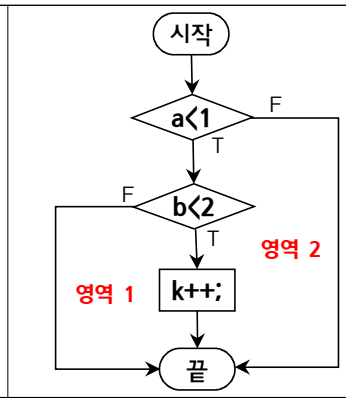
// 분기문에 사용되는 것

조건연산자	and, or, not 등
의사결정문	if-then-else, for, while, do-while, do-until, case 등

2 http://cafe.daum.net/pass365(홍재연)

[예제 1] 다음 원시코드에 대한 McCabe의 순환복잡도는?

```
void foo(int a, int b)
{
    if(a>1)
    {
        if(b<2) k++;
    }
}
```



↓ 풀이(3가지)

① **영역** 수를 이용 : CC = 그래프의 폐쇄 영역 수 + 1 = 2 + 1 = **3**

② **노드**와 **간선수**를 이용 : CC = 간선수(화살표수) – 노드수 + 2 = 6 – 5 + 2 = **3**

③ 원시코드의 **결정문** 수를 이용 : CC = 원자적 조건의 수 + 1 = 2 + 1 = **3**

[예제 2] 다음 원시코드에 대한 McCabe의 순환복잡도(cyclomatic complexity)는?

```
-----------------------------------
 int TestMe(int x, int y, int z)
 {
   if (a=1 && b=2 || c=3) k++;
 }
-----------------------------------
```

① 1　　② 2

③ 3　　④ 4

♤ McCabe의 순환복잡도

- 원시코드의 결정문 수를 이용 : CC = 원자적 조건의 수 + 1 = 1 + 1 = **2 //정답은 2**
- 문제 분석(현재까지는 이런 유형으로 출제되지 않음)
- CC = 원자적 조건의 수 + 1 = 3 + 1 = 4 //조건이 3개 있으므로 **엄격하게는 4**라고도 한다.

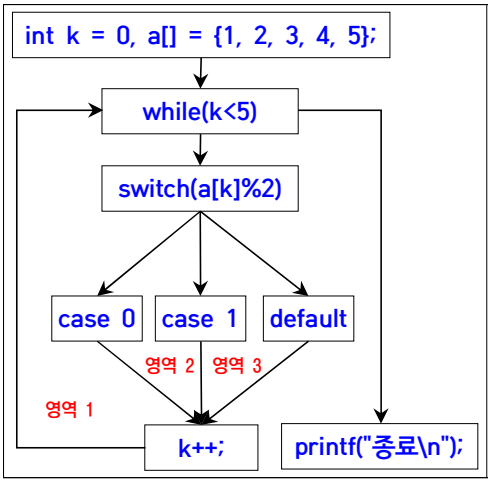
정답 : ②

[예제 3] 다음 원시코드에 대한 McCabe의 순환복잡도(cyclomatic complexity)는?

```
int main(void) {
  int k = 0, a[] = {1, 2, 3, 4, 5};
  while(k<5)
  {
    switch(a[k]%2)
    {
      case 0 : printf("짝수 : %d \n", a[k]);
               break;
      case 1 : printf("홀수 : %d \n", a[k]);
               break;
      default : printf("오류 : %d \n", a[k]);
    }
    k++;
  }
  printf("종료\n");
}
```

① 2 ② 3 ③ 4 ④ 5

♤ McCabe의 순환복잡도



① 영역 수를 이용
• CC = 그래프의 폐쇄 영역 수 + 1 = 3 + 1 = **4**

② 노드와 간선수를 이용
• CC = 간선수 − 노드수 + 2 = 10 − 8 + 2 = **4**

③ 원시코드의 결정문 수를 이용
• CC = 원자적 조건의 수 + 1 = 3 + 1 = **4**

• 결정문 : if, for, while, do-while, case 등
• else, default는 제외

정답 : ③

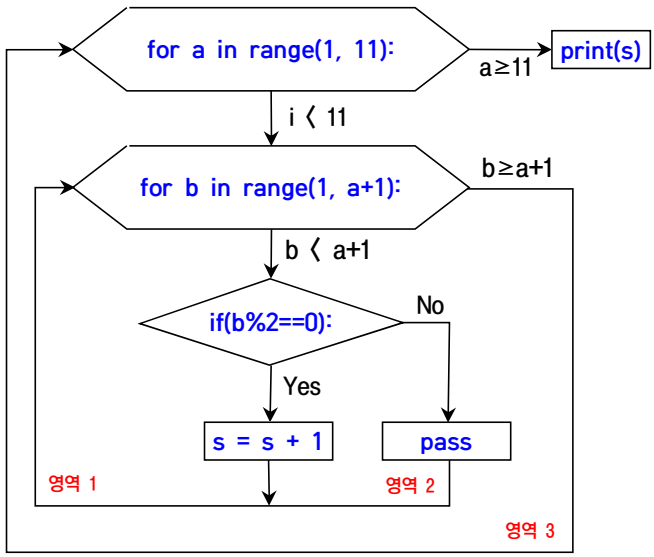
[예제 4] 다음 원시코드에 대한 McCabe의 순환복잡도(cyclomatic complexity)는?

```
----<python>-----------------------------------------------------------
 s = 0
 for a in range(1, 11):          # 1에서 10까지 반복
     for b in range(1, a+1):     # 1에서 a까지 반복
         if(b%2==0):             # b가 짝수이면 참
             s = s + 1           # 짝수 개수를 누적
         else:
             pass                # pass는 특별하게 기술할 것이 없을 때
 print(s)                        # 출력 : 25
-----------------------------------------------------------------------
```

① 2 ② 3 ③ 4 ④ 5

♤ McCabe의 순환복잡도

// 먼저, **그래프**(순서도)를 그리면 다음과 같다.



① 영역 수를 이용 : CC = 그래프의 폐쇄 영역 수 + 1 = 3 + 1 = **4**

② 노드와 간선수를 이용 : CC = 간선수(화살표수) − 노드수 + 2 = 8 − 6 + 2 = **4**

③ 원시코드의 결정문 수를 이용 : CC = 원자적 조건의 수 + 1 = 3 + 1 = **4**

• 결정문 : if, while, for, do-while, do-until, case 등

• else, default는 제외

정답 : ③

〈McCabe의 순환복잡도와 기본경로 테스트〉

- **기본경로 테스트**는 **McCabe의 순환복잡도**를 이용한 테스트이다.
- 즉, 순환복잡도를 통해 테스트할 독립적인 기본경로 수를 구한다.
- 기본경로 테스팅의 최소 테스트 케이스 수는 McCabe의 순환복잡도와 같다.
- 만약, CC = 3이면, 최소 테스트 케이스 수 = 3이다.(최소 테스트)

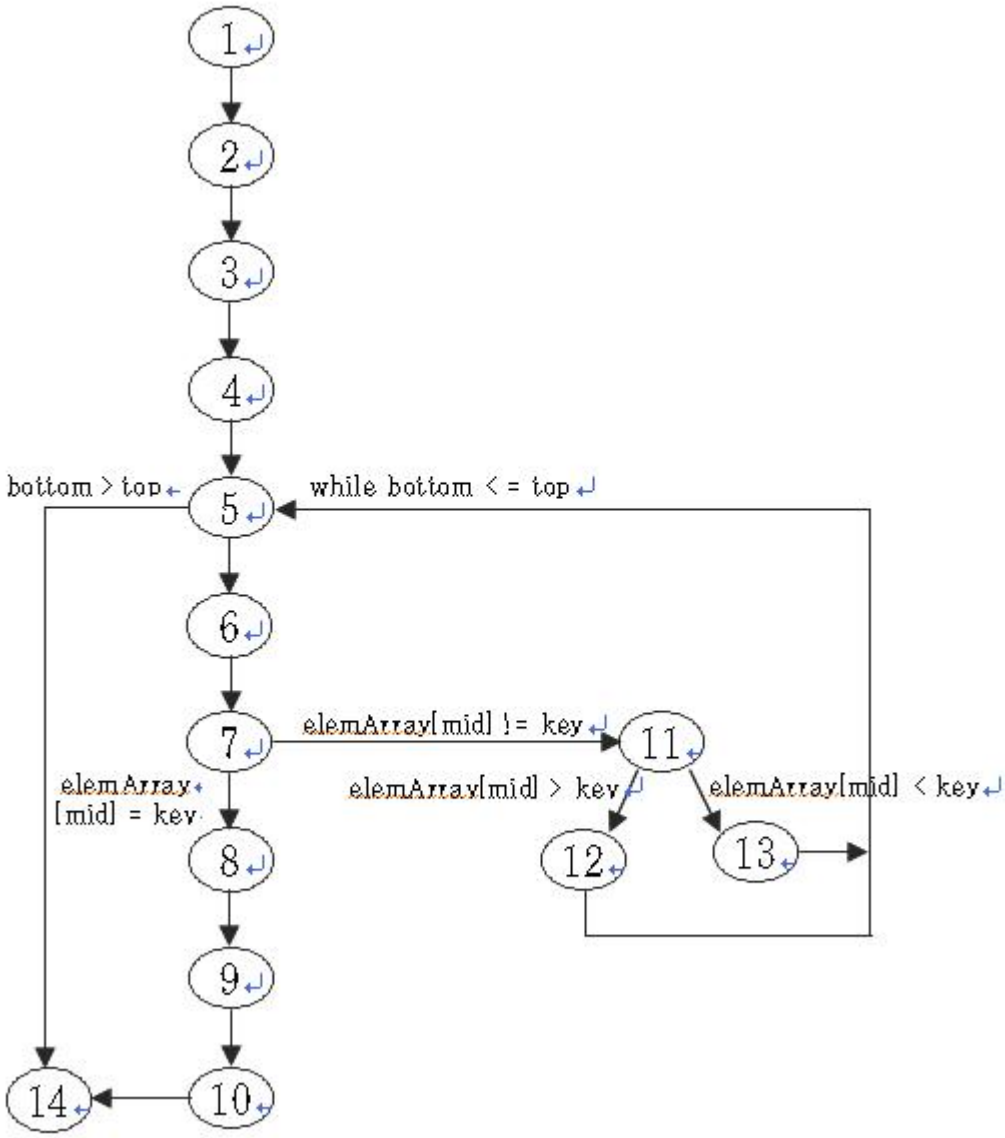
- **기본경로 테스트**에 대해서는 뒤에서 상세하게 다룬다.

- 여기서는, **기본경로 테스트**에 **McCabe의 순환복잡도**가 이용된다는 것을 명심하자!

6 http://cafe.daum.net/pass365(홍재연)

기출문제 분석

1. 다음 프로그램 흐름 그래프의 사이클로메틱(cyclomatic) 복잡도를 계산하면? [2007년 국가 7급]



① 3 ② 4
③ 5 ④ 6

♤ McCabe의 순환복잡도(CC, cyclomatic complexity)

• CC는 McCabe가 정의한 원시코드 복잡도를 정량적으로 평가하는 방법이다.

// McCabe의 순환복잡도(CC) 계산

① 원시코드상에서 계산

• CC = 분기문 수 + 1

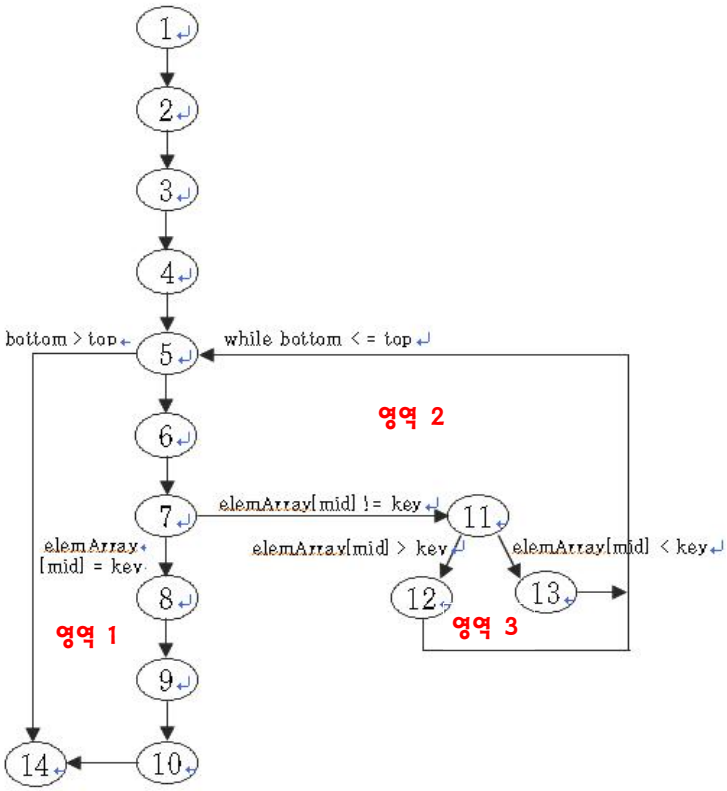
• 분기문에 사용되는 것은 다음과 같다.

조건연산자	and, or, not 등
의사결정문	if-then-else, for, while, do-while, do-until, case 등

② 순서도(제어흐름 그래프)를 이용한 계산(2가지 방식)

• CC = 화살표수(간선수) - 노드수 + 2

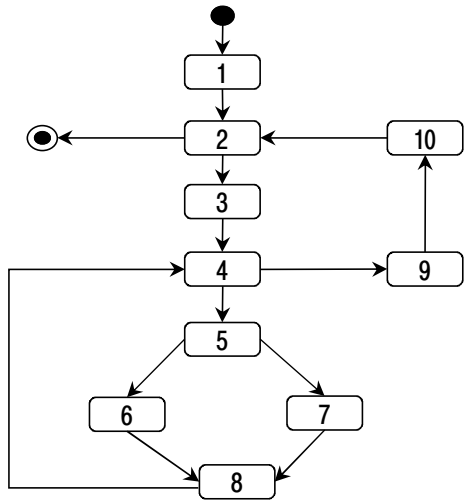
• CC = 그래프 영역 수 + 1 → 가장 간단



• 순환복잡도 = 그래프 영역 개수 + 1 = 3 + 1 = **4**

정답 : ②

2. 다음은 특정 함수의 제어흐름 그래프이다. 이 함수의 사이클로매틱 복잡도는 얼마인가? [2022년 군무원 7급]



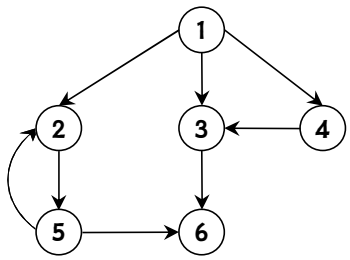
① 3 ② 4 ③ 5 ④ 6

♤ 사이클로매틱 복잡도 - 순환복잡도(CC)

• **그래프에서 영역 수를 이용 :** CC = 그래프의 폐쇄 영역 수 + 1 = **3** + 1 = **4**

정답 : ②

3. 다음 제어흐름 그래프에 대한 순환복잡도(cyclomatic complexity)는? [2021년 국가 7급]



① 2 ② 3 ③ 4 ④ 5

♤ 순환복잡도

• **그래프에서 영역 수를 이용 :** CC = 그래프의 폐쇄 영역 수 + 1 = 3 + 1 = **4**

정답 : ③

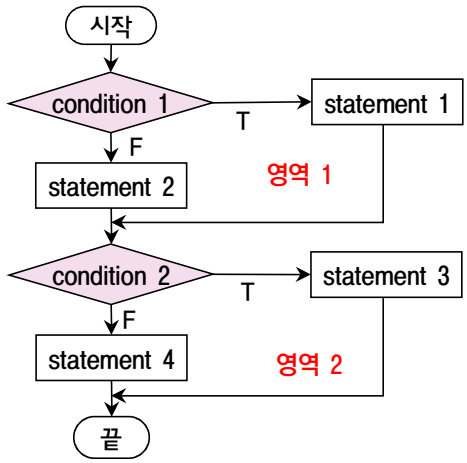
4. 다음 지문에 나타난 코드의 순환복잡도(mccabe cyclomatic complexity)로 가장 적절한 것은? [2023년 군무 7급]

```
------------------
if (condition 1)
  statement 1
else
  statement 2
if (condition 2)
  statement 3
else
  statement 4
------------------
```

① 2 ② 3 ③ 4 ④ 5

♤ **순환복잡도**

- 순환복잡도는 McCabe가 정의한 **원시코드 복잡도를 정량적**으로 평가하는 방법이다.
- 일반적으로, 순환복잡도는 원시코드에 존재하는 **분기문 수로 결정**된다.
- 기본적으로 순환복잡도는 최소값 1을 가지며 분기유발 구문마다 1씩 증가한다.



① **영역** 수를 이용 : CC = 그래프의 폐쇄 영역 수 + 1 = 2 + 1 = **3**
② **노드**와 **간선수**를 이용 : CC = 간선수(화살표수) − 노드수 + 2 = 9 − 8 + 2 = **3**
③ 원시코드의 **결정문** 수를 이용 : CC = 원자적 조건의 수 + 1 = 2 + 1 = **3**

정답 : ②

5. <보기>에 제시된 코드에 대한 순환복잡도를 계산하여 기본경로 테스팅을 수행하고자 한다. 이 코드의 순환복잡도 수치는? [2020년 서울 7급]

```
----<보기>-----------------------------------------------
while(True):
    code = ''
    for a in range(12):
        code+=str(chr(random.randint(65,90)))
    if len(Model.objects.filter(name__contains=code)) > 0:
        continue
    else :
        break
    print(code)
-------------------------------------------------------
```

① 3　　② 4
③ 5　　④ 6

♤ **순환복잡도 계산 공식 - 3가지**

① **그래프에서 영역 수를 이용** : CC = 그래프의 폐쇄 영역 수 + 1
② **그래프에서 노드와 간선수를 이용** : CC = 간선수(화살표수) − 노드수 + 2
③ **원시코드의 결정문 수를 이용** : CC = 원자적 조건의 수 + 1

- 일반적으로, 순환복잡도는 원시코드에 존재하는 분기문 수로 결정된다.
- 기본적으로 순환복잡도는 최소값 1을 가지며 분기유발 구문마다 1씩 증가한다.
- 원자적 조건은 분기문(결정문)에서 참과 거짓으로 판별되는 조건을 의미한다.
- 분기문에 사용되는 것은 다음과 같은 것이 있다.

조건연산자	and, or, not 등
의사결정문	if-then-else, for, while, do-while, do-until, case 등

↓
↓ **결정문 수를 이용**하여 순환복잡도 수치를 구하면
↓

- 순환복잡도 수치 = 의사결정문 수 + 1 = 3 + 1 = **4**
  ↓**3개**

```
while(True):, for a in range(12):, if len(Model.objects.filter(name__contains=code)) > 0:
```

정답 : ②

6. 다음 코드에서 McCabe의 순환복잡도(cyclomatic complexity)는? [2013년 국가 7급]

```
-----------------------------------
int TestMe(int x, int y, int z) {
  if (x >= y) {
    if (x >= z)
      result = x;
    else
      result = z;
  }
  else
    result = y;
}
-----------------------------------
```

① 1 ② 2 ③ 3 ④ 4

♤ **McCabe의 순환복잡도**

- McCabe의 순환복잡도 = 의사결정수 + 조건수 + 1 = 2 + 0 + 1 = 3
- 의사결정수 : If-Then-Else, Do-While, Do-Until, Case 등의 의사결정수
- 조건수 : AND, OR, NOT 등의 조건수

정답 : ③

7. McCabe의 순환복잡도(CC : cyclometic complexity)에 대한 설명으로 옳지 않은 것은? [2016년 국가 7급]

① 프로그램의 논리적인 복잡도를 정량적으로 측정하기 위한 척도(metric)이다.
② 프로그램의 분기 노드의 수보다 항상 큰 값을 갖는다.
③ 프로그램의 모든 경로를 최소 한 번 이상 실행하도록 테스트하기 위한 입력의 개수에 해당한다.
④ 플로우 그래프(flow graph)에서 간선(edge)의 수를 E, 노드의 수를 N이라고 하면 CC = E - N + 2이다.

♤ **McCabe의 순환복잡도**

- 프로그램의 **모든 경로**를 최소 한 번 이상 실행하도록 테스트하기 위한 **입력의 개수**에 해당한다.(×) ⇢ **순환복잡도**는 **원시코드 복잡도를 정량적**으로 평가하는 방법이다.
  ⇢ **순환복잡도**는 **기본경로 테스트**에 이용하는 것이다.

정답 : ③

8. T. McCabe의 순환복잡도(cyclomatic complexity)에 대한 설명으로 옳지 않은 것은? [2020년 국가 7급]

① 사이클로매틱 수는 각 모듈에 대한 제어도(fan-out)를 이용하여 측정한다.
② 사이클로매틱 수는 코드 전체에서 독립적인 경로의 수를 선형적으로 측정한다.
③ 사이클로매틱 수는 그래프 이론을 기반으로 하여, 코드를 동등한 제어흐름 그래프로 변환한 다음, 메트릭을 결정하기 위한 그래프의 속성을 이용하여 계산한다.
④ 원시코드의 구조적인 복잡성을 알아내는 측도이다.

♤ **McCabe의 순환복잡도**

• **사이클로매틱 수**는 각 모듈에 대한 **제어도(fan-out)**를 이용하여 측정한다.**(×)**
→ **사이클로매틱 수(순환복잡도)**는 원시코드에 존재하는 **분기문 수**로 결정된다.
→ **제어도(fan-out)**는 한 모듈이 직접 호출하는 **하위모듈 수**

• 순환복잡도는 McCabe가 정의한 원시코드 복잡도를 정량적으로 평가하는 방법이다.
• **기본경로 테스트**는 **McCabe의 순환복잡도**를 이용한 테스트이다.
• 즉, 순환복잡도를 통해 테스트할 독립적인 기본경로 수를 구한다.

〈McCabe의 순환복잡도〉
• McCabe의 순환복잡도(cyclometic complexity)는 CC라고도 한다.
• 일반적으로, 순환복잡도는 원시코드에 존재하는 분기문 수로 결정된다.
• 기본적으로 순환복잡도는 최소값 1을 가지며 분기 유발구문마다 1씩 증가한다.
• 원시코드에 1개의 if문이 있으면 순환복잡도는 1 + 1 = 2가 되고,
• 원시코드에 if문 1개, for문 2개가 있으면 순환복잡도는 1 + 2 + 1 = 4가 된다.
• MaCabe는 CC를 10이하로 유지할 것을 권고하였다.(1976년)
• 현재, 프로그래밍 환경은 많이 변했다. C++ 표준에는 CC를 20이하로 유지할 것을 명시

↓
↓순환복잡도 계산
↓

〈순환복잡도 계산 공식 - 3가지〉

① **그래프에서 영역 수를 이용** : CC = 그래프의 폐쇄 영역 수 + 1
② **그래프에서 노드와 간선수를 이용** : CC = 간선수(화살표수) - 노드수 + 2
③ **원시코드의 결정문 수를 이용** : CC = 원자적 조건의 수 + 1

정답 : ①

9. 다음 용어에 대한 설명으로 옳은 것은? [2019년 국가 7급]

① 사용자 스토리(user story)는 개발자 관점에서 프로젝트를 통해 구현하고 싶은 기능을 상세하게 표현한 알고리즘이다.

② 베이스라인(baseline)은 공식적인 변경절차 없이 언제라도 개발자가 변경할 수 있는 상태에 있는 산출물의 집합이다.

③ 기능점수(function point)는 시스템을 구현한 기술에 의존적이고 개발자에 의해 식별되는 기능에 기반하여 시스템의 크기를 측정하는 척도이다.

④ 순환복잡도(cyclomatic complexity)는 원시코드의 복잡도를 정량적으로 평가하는 척도이며 기본경로(basis path) 테스팅과 밀접한 관련이 있다.

♤ **용어**

① **사용자 스토리**(user story)는 **개발자 관점**에서 프로젝트를 통해 구현하고 싶은 기능을 상세하게 표현한 알고리즘이다.**(×)**

→ **사용자 스토리**는 실제 **사용자들에게 가치 있는 기능**을 간단명료하게 기술한 것이다.

→ 대부분 애자일 방법론들은 사용자 스토리를 사용한다.

→ 사용자 스토리는 일상생활에서 사용하는 용어를 통해 시스템과 사용자의 상호작용을 기술하여 사용자의 요구사항을 발췌한다.

② **베이스라인**(baseline)은 공식적인 **변경절차 없이** 언제라도 개발자가 변경할 수 있는 상태에 있는 산출물의 집합이다.**(×)**

→ **베이스라인**은 **공식적인 변경 제어 프로세스**를 통해서만 수정될 수 있다.

→ 베이스라인은 개발의 기초가 된다.

③ **기능점수**(function point)는 시스템을 구현한 **기술에 의존적**이고 **개발자에 의해 식별**되는 기능에 기반하여 시스템의 크기를 측정하는 척도이다.**(×)**

→ **기능점수**는 시스템이 가지는 **기능 개수를 기준**으로 시스템 규모를 측정하는 기법이다.

→ 기능점수는 시스템 개발비용을 산정하는데 이용된다.

→ 기능점수 분석은 사용자 관점에서 시스템 개발 규모를 측정하기 위한 표준 기법이다.

④ **순환복잡도**(cyclomatic complexity)는 **원시코드의 복잡도를 정량적으로 평가**하는 척도이며 기본경로(basis path) 테스팅과 밀접한 관련이 있다.**(○)**

→ 기본경로 테스트는 McCabe의 순환복잡도를 이용한 테스트이다.

→ 순환복잡도를 통해 테스트할 독립적인 기본경로의 수를 구한다.

→ 기본경로 테스트의 최소 테스트 케이스 수는 McCabe의 순환복잡도와 같다.

정답 : ④