9. 재귀함수(recursive function)

재귀함수는 함수 내에서 **자신**을 다시 호출하는 것이다.**(되부름, 재귀호출, 자기호출, 순환호출)**
재귀함수를 설명하기 전에 **반복함수**와 **재귀함수(순환함수)**를 비교해 본다.
다음은 **계승함수** n!을 구하는 과정을 반복함수와 재귀함수로 나타낸 것이다.

```
int factorial(int n)        //반복함수
{
    int i, f;

    f = 1;
    for(i=2; i<=n; i++)
      f*=i;

    return f;
}
void main()
{
    int y;
    y = factorial(4);
    printf("%d\n", y);
}
```

```
int factorial(int n)        //재귀함수
{
    int i, f;

    if(n<=1)                //완료조건(중요!)
      f = 1;
    else
      f = n * factorial(n-1);
    return f;
}
void main()
{
    int y;
    y = factorial(4);
    printf("%d\n", y);
}
```

◇ f = n * factorial(n-1) //재귀함수

↓ n = 4이면

```
f = 4 * factorial(3)
  = 4 * 3 * factorial(2)
  = 4 * 3 * 2 * factorial(1)      //완료조건 : factorial(1) = 1
  = 4 * 3 * 2 * 1                 //factorial(1)이면 f = 1이므로 factorial(1) = 1이다.
  = 24
```

◈ **반복함수와 순환함수 정리**

① 순환함수로 구현된 **모든 알고리즘**은 순환을 사용하지 않는 방법으로 표현할 수 있다.
② 순환 알고리즘은 반복에 비해 명확하고 간결하게 표현된다.(세상 원리)
③ 순환 알고리즘은 반복 알고리즘에 비해 일반적으로 수행시간이 느리다.
④ 순환 알고리즘은 반복 알고리즘에 비해 기억공간이 더 많이 필요하다.(별도 Stack을 사용하므로)
⑤ 문제 자체가 순환적으로 정의되어 있으면, 순환함수로 처리하는 것이 명확하고 간단해 진다.
예를 들면, 2개의 이진트리에 대한 동일성 비교 같은 문제이다.
⑥ 순환함수에는 **완료조건**이 필요하다.(그렇지 않으면 Stack overflow가 발생하여 시스템은 정지)

예제 1	순환(recursive; 되부름, 재귀호출)으로 처리된 함수

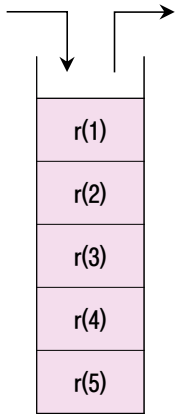
```
int r(int   n){
    int h;

    if ( n == 1 )              //완료조건
        h = 1;
    else
        h = n + r(n - 1);      //되부름

    return   h;
}

void main(void){
    int d, k = 5;

    d = r(k);
    printf("실행결과 : %d\n", d);
}
실행결과 : 15
```



- 함수가 되부름되면 함수의 상태는 순차적으로 스택에 적재되고
- 함수 종료시 return문이 실행되면서 스택의 내용은 하나씩 제거된다.
- 이때 되부름 순간에 적용된 **연산**이 **수행**된다.(여기서는 **+ 연산**이 수행된다)

```
h = n + r(n - 1);
        ↓n = 5일 때,
h = 5 + r(4)
  = 5 + 4 + r(3)
  = 5 + 4 + 3 + r(2)
  = 5 + 4 + 3 + 2 + r(1) 처럼 내부적으로 진행된다.
                    └→ r(1)의 값은 1이다.
                       마지막, r(1)인 경우에 h = 1;로 함수값을 반환하므로
h = 5 + 4 + 3 + 2 + 1 = 15
```

예제 2	다음 C 프로그램 출력 결과는? – 되부름 수식에 음수가 포함된 경우

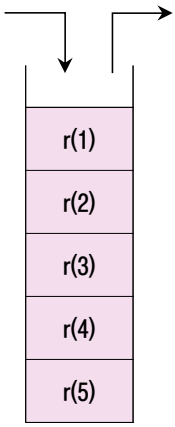
```
int r(int n)
{
    int h;

    if ( n == 1 )              //완료조건
        h = 1;
    else
        h = n - r(n - 1);      //되부름

    return  h;
}

void main(void)
{
    int d, k = 5;

    d = r(k);
    printf("실행결과 : %d\n", d);
}
실행결과 : 3
```



- 함수가 되부름되면 함수의 상태는 순차적으로 스택에 적재되고
- 함수 종료시 return문이 실행되면서 스택의 내용은 하나씩 제거된다.
- 이때 되부름 순간에 적용된 **연산**이 **수행**된다.(여기서는 **– 연산**이 수행된다)

// 되부름 수식에 음수가 포함된 경우 – 완료조건을 구한 후에 수식의 값을 찾는다.

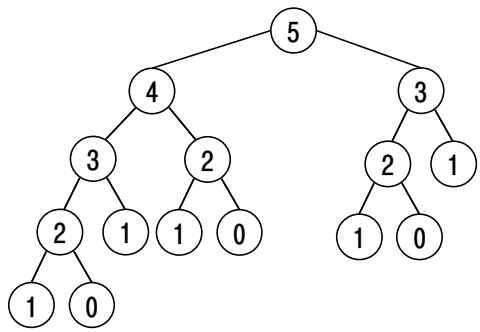
- r(1) = 1 //완료조건
- r(2) = 2 - r(1) = 2 - 1 = 1
- r(3) = 3 - r(2) = 3 - 1 = 2
- r(4) = 4 - r(3) = 4 - 2 = 2
- r(5) = 5 - r(4) = 5 - 2 = 3 //출력결과 : 3

예제 3	다음 함수 fib()를 사용하여 fib(5)를 실행했을 때, fib(5)를 포함한 fib() 함수의 총 재귀호출 횟수와 최종 리턴 값은?

```
int fib(int n)
{
  if(n<=0) return 0;    //완료조건
  if(n==1) return 1;    //완료조건
  else return fib(n-1) + fib(n-2);
}
```

◈ 함수 fib()가 호출되는 횟수 : 15번

함수 fib()가 호출되는 횟수는 다음처럼 재귀트리를 그려서 풀면 된다.



• 동그라미 수가 **재귀호출 횟수**이다.(15번)

◈ 함수 fib()의 리턴 값 : 5

• fib(0) = 0 **//완료조건**
• fib(1) = 1 **//완료조건**
• fib(2) = fib(1) + fib(0) = 1 + 0 = 1
• fib(3) = fib(2) + fib(1) = 1 + 1 = 2
• fib(4) = fib(3) + fib(2) = 2 + 1 = 3
• fib(5) = fib(4) + fib(3) = 3 + 2 = **5** → 리턴 값

[Tip] 국가 7급 시험에서 재귀호출 문제 유형(출제빈도가 매우 높음)

• 재귀함수가 호출되는 횟수 구하기 : 재귀트리로 해결
• 재귀함수의 리턴 값 구하기 : 완료조건으로 해결
• 재귀함수의 출력 결과 값 구하기 : 가장 어려운 부분(뒤에서 다룸)

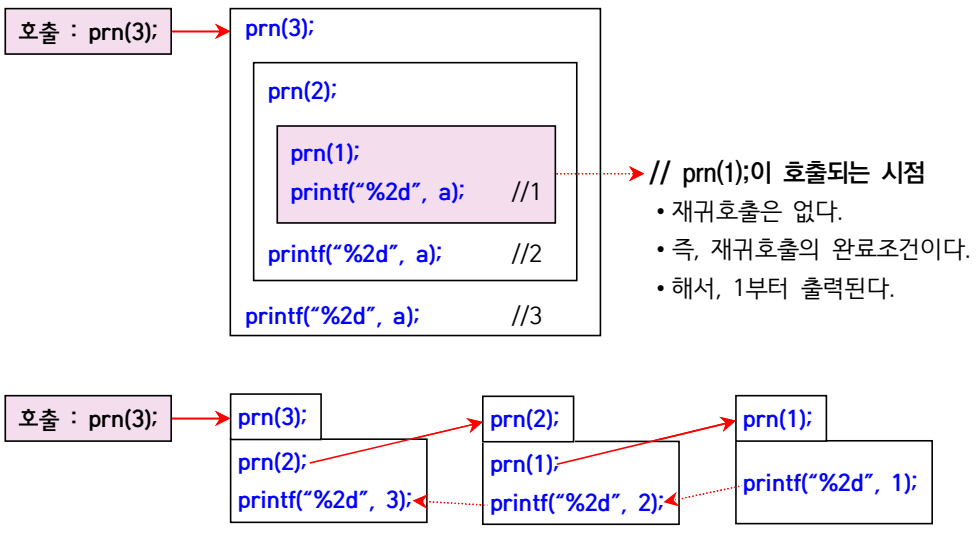
예제 4	다음 C 프로그램 출력 결과는?

```
-----------------------
#include <stdio.h>
void prn(int a){
   if(a>1)        //완료조건
      prn(a-1);
   printf("%2d", a);
}
void main(){ prn(3); }
-----------------------
```

① 1 2 3　　② 2 3
③ 3 2 1　　④ 3 2

♤ **재귀호출**

- 재귀호출은 자신을 그대로 호출하지만, 내부적으로 변수 값은 변할 수 있다.(당연!)
- 재귀호출은 변수 값이 어떻게 변하느냐? 에 따라서 실행 결과는 다르게 된다.(당연!)
- 다음은 재귀호출 수행 과정을 2종류의 그림으로 나타낸 것이다.**(이해하기 쉬운 것 택일!)**



- 주어진 그림을 보면, 출력 결과가 **1 2 3**라는 것을 알 수 있을 것이다.
- 재귀호출문과 출력문 위치에 따라서 실행 결과는 다르게 된다.

정답 : ①

예제 5	다음 C 프로그램 출력 결과는?

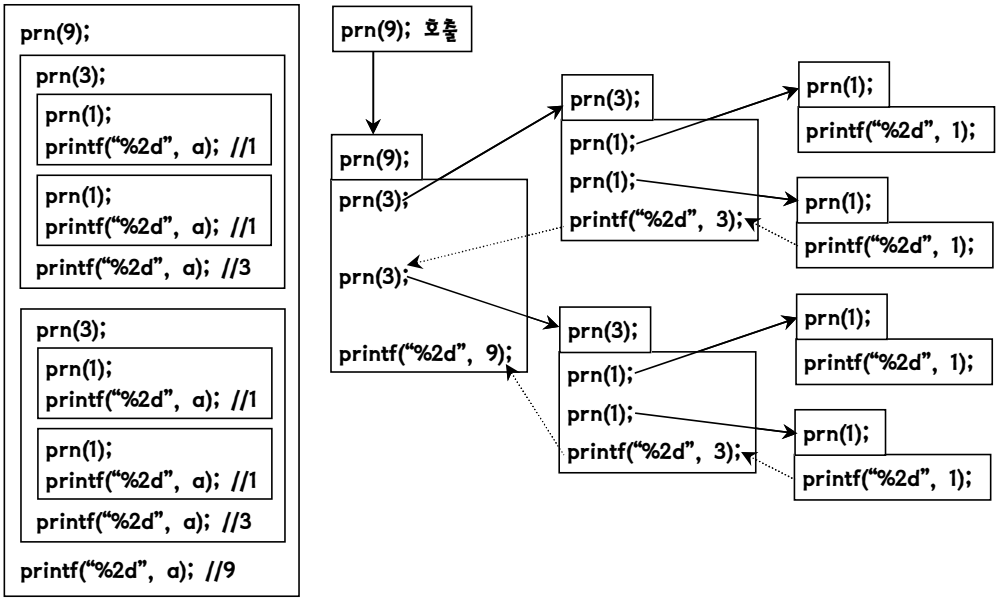
```
-------------------------
#include <stdio.h>
void prn(int a) {
  if(a>1){
    prn(a/3);
    prn(a/3);
  }
  printf("%2d", a);
}
void main(){ prn(9); }
-------------------------
```

① 1　　② 1 1 3 1 1 3 9

③ 9　　④ 9 3 1 1 3 1 1

♤ 재귀호출

• 주어진 재귀호출 수행 과정을 2종류의 그림으로 나타낸 것이다.(이해하기 쉬운 것 택일!)



• 주어진 그림을 보면, 출력 결과가 1 1 3 1 1 3 9라는 것을 알 수 있을 것이다.

• 재귀호출은 자신을 그대로 호출하지만, 내부적으로 변수 값은 변할 수 있다.(당연!)

정답 : ②

예제 6	다음 C 프로그램 출력 결과는?

```
-------------------------
#include <stdio.h>
void prn(int a) {
   printf("%2d", a);
   if(a>1){
      prn(a/3);
      prn(a/3);
   }
}
void main(){ prn(9); }
-------------------------
```

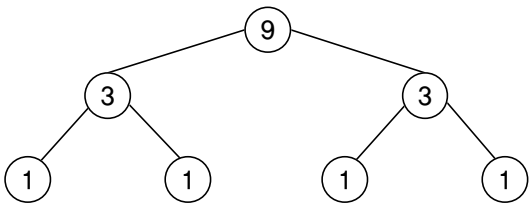
① 1　　② 1 1 3 1 1 3 9

③ 9　　④ 9 3 1 1 3 1 1

♤ **재귀호출 - 재귀트리 이용**

• 먼저, 주어진 재귀호출 문제를 재귀트리로 나타내면 다음과 같다.

• prn(9);를 수행



• 전위운행(중,좌,우) : 9 3 1 1 3 1 1

• 후위운행(좌,우,중) : 1 1 3 1 1 3 9

void prn(int a)
{
  printf("%2d", a); //출력문이 앞에 있음
  if(a>1){
  prn(a/3);
  prn(a/3);
  }
}
	void prn(int a)
	{
	  if(a>1){
	  prn(a/3);
	  prn(a/3);
	  }
	  printf("%2d", a); //출력문이 뒤에 있음
	}

출력 결과 : 9 3 1 1 3 1 1	출력 결과 : 1 1 3 1 1 3 9

• 출력문이 **앞**에 있는 경우 : 트리를 **전위**운행한 결과와 같다.

• 출력문이 **뒤**에 있는 경우 : 트리를 **후위**운행한 결과와 같다.

정답 : ④

기출문제 분석

1. 다음 C 함수를 사용하여 function(9)를 수행했을 때, 반환되는 값은? [2017년 국가 7급]

```
--------------------------------
int function(int n)
{
  if (n <= 1) return 1;
  if (n % 2 == 0)
    return n * function(n-1);
  else
    return n * function(n-3);
}
--------------------------------
```

① 60 ② 84
③ 540 ④ 672

♤ **재귀호출**

풀이 1	• function(9) = 9 * function(6) = 9 * 6 * function(5) = 9 * 6 * 5 * function(2) = 9 * 6 * 5 * 2 * function(1) = 9 * 6 * 5 * 2 * 1 = **540**

풀이 2	• function(0) = 1 **//완료조건** • function(1) = 1 **//완료조건** • function(2) = 2 * function(1) = 2 * 1 = 2 • function(5) = 5 * function(2) = 5 * 2 = 10 • function(6) = 6 * function(5) = 6 * 10 = 60 • function(9) = 9 * function(6) = 9 * 60 = **540**

• 수식에 연산우선순위가 서로 다른 연산자가 섞여 있을 때는 **풀이 2**를 이용하는 것이 좋다.

정답 : ③

2. 다음 C 프로그램의 실행 결과로 옳은 것은? [2020년 군무 7급]

```
---------------------------------
int fun(int n)
{
  if(n<1) return 1;
  return fun(n-1) + fun(n/2);
}
int main()
{
  int k;
  k = fun(5);
  printf("%d\n", k);
  return 0;
}
---------------------------------
```

① 7　　② 12
③ 14　　④ 16

♤ **재귀호출 – 반환값 구하기**

```
if(n<1) return 1;          // 완료조건
return fun(n-1) + fun(n/2);
```

↓

↓ fun(7) = ?

↓

- fun(0) = 1
- fun(1) = fun(0) + fun(1/2) = fun(0) + fun(0) = 1 + 1 = 2
- fun(2) = fun(1) + fun(2/2) = fun(1) + fun(1) = 2 + 2 = 4
- fun(3) = fun(2) + fun(3/2) = fun(2) + fun(1) = 4 + 2 = 6
- fun(4) = fun(3) + fun(4/2) = fun(3) + fun(2) = 6 + 4 = 10
- fun(5) = fun(4) + fun(5/2) = fun(4) + fun(2) = 10 + 4 = **14 ← 반환값**

// C에서 정수와 정수의 연산 결과는 정수이다.

- fun(1/2) = fun(0) = 1
- fun(3/2) = fun(1) = 2

정답 : ③

3. 다음 C 코드의 실행 결과는? [2021년 국가 7급]

```
-----------------------------------
#include <stdio.h>
int func (int n)
{
  if(n <= 1) return 2;
  return func (n-1) + n;
}
int main (int argc, char* argv[])
{
  printf("%d\n", func (10) );
  return 0;
}
-----------------------------------
```

① 55 ② 56
③ 58 ④ 60

♤ **재귀호출**

```
if(n <= 1) return 2;      // 완료조건, n=1일 때, 반환값이 2인 것에 주의!
return func (n-1) + n;
```

↓
↓ func(10) = ?
↓

- func(1) = 2
- func(2) = func(1) + 2 = 2 + 2 = 4
- func(3) = func(2) + 3 = 4 + 3 = 7
- func(4) = func(3) + 4 = 7 + 4 = 11
- func(5) = func(4) + 5 = 11 + 5 = 16
- func(6) = func(5) + 6 = 16 + 6 = 22
- func(7) = func(6) + 7 = 22 + 7 = 29
- func(8) = func(7) + 8 = 29 + 8 = 37
- func(9) = func(8) + 9 = 37 + 9 = 46
- func(10) = func(9) + 10 = 46 + 10 = **56**

정답 : ②

4. 관계식 T(n)이 <보기>와 같을 때, 옳지 않은 것은? [2022년 서울 7급]

----<보기>----------
T(n) = 2T(n/2) + n, T(0) = T(1) = 1
---

① $T(n) = \Omega(n^2)$ ② $T(n) = \Theta(nlogn)$

③ $T(n) = O(n^2)$ ④ $T(n) = O(nlogn)$

♤ **점화식 – 신중하게 재귀호출되는 그대로 적용해야 함**

$T(n) = 2T(n/2) + n$

$= 2[2T(\frac{n}{2^2}) + \frac{n}{2}] + n = [2^2T(\frac{n}{2^2}) + n] + n = 2^2T(\frac{n}{2^2}) + n + n$

$= 2^2[2T(\frac{n}{2^3}) + \frac{n}{2^2}] + n + n = 2^3T(\frac{n}{2^3}) + n + n + n$

:

$= 2^kT(\frac{n}{2^k}) + kn$

↓ 여기서, $\frac{n}{2^k} = 1$이고 2^k=n

$= nT(1) + kn$

↓ T(1)=1

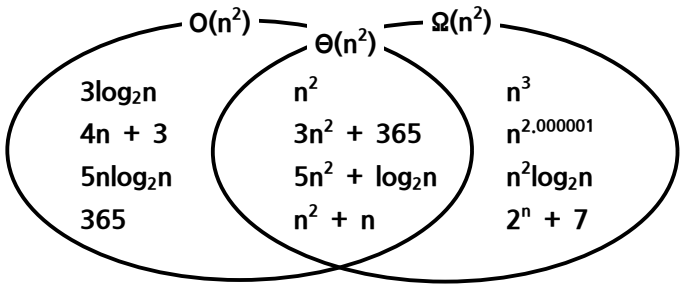
$= n + kn$

↓ k=$\log_2 n$

$= n + n\log_2 n$

$= \Theta(n\log_2 n) = O(n\log_2 n) = O(n^2)$

// **O(n^2), Ω(n^2), Θ(n^2)의 관계**



정답 : ①

5. 다음 C 언어 프로그램의 출력 결과는? [2018년 국가 7급]

```
-----------------------------------------------
int i;
int RecFunc(int n) {
  i++;
  if (n <= 1) return 1;
  else  return RecFunc(n-1) + RecFunc(n-2);
}
int main() {
  int result;
  i = 0;
  result = RecFunc(5);
  printf("%d, %d \n", i, result);
}
-----------------------------------------------
```

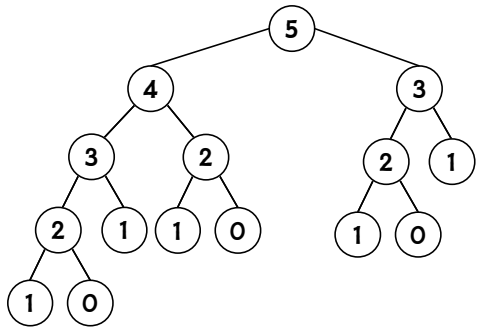
① 13, 5 ② 13, 8 ③ 15, 5 ④ 15, 8

♤ C 언어 프로그램 – 재귀호출

- RecFunc(0) = 1 **//완료조건**
- RecFunc(1) = 1 **//완료조건**
- RecFunc(2) = RecFunc(1) + RecFunc(0) = 1 + 1 = 2
- RecFunc(3) = RecFunc(2) + RecFunc(1) = 2 + 1 = 3
- RecFunc(4) = RecFunc(3) + RecFunc(2) = 3 + 2 = 5
- RecFunc(5) = RecFunc(4) + RecFunc(3) = 5 + 3 = **8**

◈ **외부변수 i = ?**

- 외부변수 i의 값은 재귀호출 되는 횟수이다.(재귀트리로 구함)



- 동그라미 수가 재귀호출 횟수이다.(15번)

정답 : ④

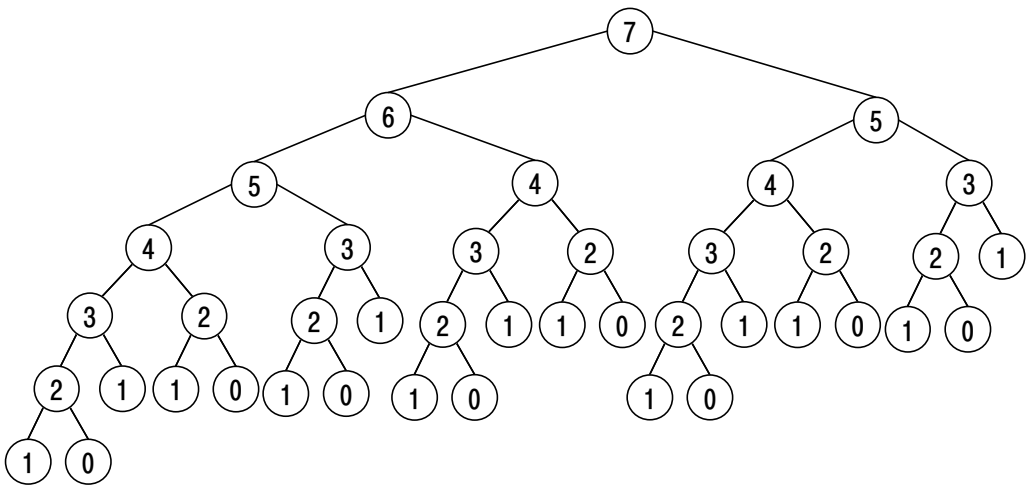
6. 순환(recursive) 함수 f()에 대한 첫 번째 호출이 f(7)일 때, f(7)을 포함하여 함수 f()가 호출되는 총 횟수는? [2014년 국가 7급]

```
int f(int n)
{
    if(n == 0 || n == 1)
        return n;
    return f(n-1) + f(n-2);
}
```

① 25　　② 35
③ 41　　④ 51

♤ **피보나치수열과 재귀트리**

- f(7) = f(6) + f(5) → f(7)을 구하기 위해서는 f(6)과 f(5)가 필요하고,
- f(6) = f(5) + f(4) → f(6)을 구하기 위해서는 f(5)와 f(4)가 필요로 한다.
- f(5) = f(4) + f(3)
- f(4) = f(3) + f(2)
- f(3) = f(2) + f(1)
- f(2) = f(1) + f(0)
- f(1) = 1 **//완료조건**
- f(0) = 0 **//완료조건**



- 동그라미 수가 호출되는 수이다.(41번)

정답 : ③

7. 다음 C 함수를 이용하여 asterisk(9)를 수행할 때 출력되는 '∗'의 개수는? [2016년 국가 7급]

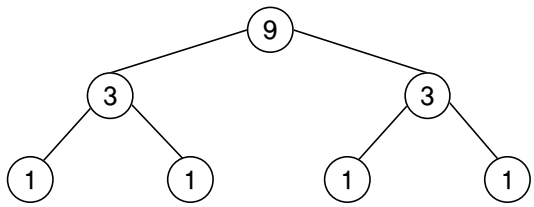
```
-------------------------
#include <stdio.h>
void asterisk(int I)
{
  if(i>1)
  {
    asterisk(i/3);
    asterisk(i/3);
  }
  printf("*");
}
-------------------------
```

① 6　　② 7
③ 8　　④ 9

♤ **재귀호출 - 재귀트리 이용**

• 먼저, 주어진 재귀호출 문제를 재귀트리로 나타내면 다음과 같다.

// asterisk(9)를 수행



• 간단하게 답을 구하면, asterisk(9)가 7번 호출되므로 '∗'가 **7**개 출력된다.

• 국가 7급에서 재귀호출 문제는 거의 매년 출제되고 있다.
• 주어진 문제처럼
재귀호출에서 같은 것을 단순하게 출력하는 것을 묻는 문제는
재귀트리를 그려서 푸는 것이 가장 빠르다.

정답 : ②

8. 다음 C 언어 프로그램의 출력 결과는? [2020년 국가 7급]

```
#include <stdio.h>
void func(char s[], int size)
{
  if (size <= 0) return;
  s[size-1] = s[size-1] + 1;
  func(s, size-2);
}
int main()
{
  char str[] = "window";
  func(str, 6);
  printf("%c", str[3]);
}
```

① d　　② e
③ n　　④ o

♤ 재귀호출

```
// 프로그램 분석
  void func(char s[], int size)    //size = 6
  {
    if (size <= 0) return;         //완료조건
    s[size-1] = s[size-1] + 1;     //해당 인덱스 위치의 문자를 1 증가시킴
    func(s, size-2);               //인덱스 감소 : -2
  }
```

↓메모리 구조

0	1	2	3	4	5	→ 인덱스(첨자)
w	i	n	d	o	w	
	↓ j		↓ e		↓ x	→ 인덱스 위치의 문자를 1 증가

• str[3] = e

정답 : ②

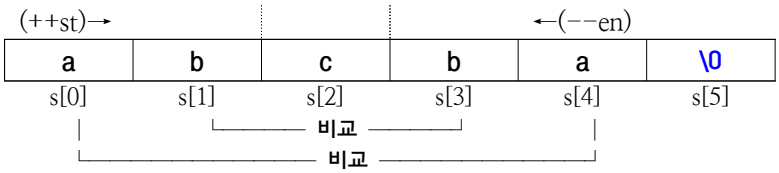
9. 〈보기〉의 C언어 재귀함수는 크기가 n인 영문과 숫자로 구성된 문자 배열 str[]이 회문(palindrome)이면 1을 반환하고, 그렇지 않으면 0을 반환한다. ㉠과 ㉡에 들어갈 내용을 옳게 짝지은 것은? (단, 회문(palindrome)이란 문자열을 거꾸로 해도 원래의 문자열과 동일한 문자열을 말한다. 예시: 1234321, abcba, a) [2021년 서울 7급]

```
----<보기>-------------------------------------------------------
int is_palin(char str[], int n)
{
  if ( n <= 1 ) return 1;
  if ( str[0] == str[n-1] ) return is_palin (____㉠____, ____㉡____);
  else return 0;
}
-----------------------------------------------------------------
```

	㉠	㉡
①	str	n-1
②	str	n-2
③	str+1	n-1
④	str+1	n-2

♤ 재귀함수 – 회문

// 회문을 판별하는 원리는 다음과 같다.



- 양 끝에서 안쪽으로 한 글자씩 좁혔을 때 글자가 서로 같으면 회문이다.
- 인덱스 st, en이 단어 중간에 위치하면, en <= st가 성립하게 된다.

// 프로그램 분석

```
int is_palin(char str[], int n)
{
  if(n<=1) return 1;                                  //문자열이 회문이면 1을 리턴
  if(str[0]==str[n-1]) return is_palin(㉠ str+1, ㉡ n-2);  //재귀호출
  else return 0;                                      //문자열이 회문이 아니면 0을 리턴
}
```

㉡ n-2에서 **n-2**인 이유는 한번 비교할 때마다 **문자 2개씩 감소**하므로

정답 : ④

10. 다음 함수를 이용하여 f(4)를 수행한 결과 값은? [2015년 국가 7급]

```
int f(int n)
{
  if (n <= 0)
    return 0;
  else if (n <= 2)
    return n;
  else
    return f(n-3) + f(n-2) + f(n-1);
}
```

① 3　　② 4
③ 5　　④ 6

♤ **재귀호출**

- f(0) = 0 **//완료조건**
- f(1) = 1 **//완료조건**
- f(2) = 2 **//완료조건**

- f(n) = f(n-3) + f(n-2) + f(n-1);
  ↓
  ↓ n=4 이면
  ↓
- f(4) = f(1) + f(2) + f(3)
  = 1 + 2 + f(3)
  = 1 + 2 + f(0) + f(1) + f(2)
  = 1 + 2 + 0 + 1 + 2
  = **6**

정답 : ④

11. 다음 코드의 출력으로 가장 옳은 것은? [2022년 군무원 7급]

```
1. int A(int a, int b) {
2.       if (b==1) return a;
3.       else return a + A(a, b-1);
4. }
5. main() {
6.       A(3, 4);
7. }
```

① 12　　② 9
③ 6　　④ 10

♤ **재귀호출**

```
if (b==1) return a;          // 완료조건, b=1일 때, 반환값은 a
else return a + A(a, b-1);
```

↓

- A(a, b) = a + A(a, b-1)

↓ A(3, 4)

- A(3, 4) = 3 + A(3, 3)
- A(3, 4) = 3 + 3 + A(3, 2)
- A(3, 4) = 3 + 3 + 3 + A(3, 1)

↳ A(3, 1) = 3

- A(3, 4) = 3 + 3 + 3 + 3
- A(3, 4) = 12

정답 : ①

12. 다음은 x^n을 구하는 알고리즘이다. 이 알고리즘의 시간복잡도는? (단, n>0이며, even(m)은 m이 짝수일 때 참을 반환하고 그렇지 않을 때 거짓을 반환하는 함수이다) [2014년 국가 7급]

```
int PowersRec(int x, int n)
{
    int Pow;
    if (n == 1)
        Pow = x;
    else
    {
        if (even(n))
            Pow = PowersRec(x*x, n/2);
        else
            Pow = x * PowersRec(x*x, (n-1)/2);
    }
    return (Pow);
}
```

① $\Theta(1)$ ② $\Theta(\log n)$
③ $\Theta(n)$ ④ $\Theta(n\log n)$

♤ 거듭제곱(x^n) 구하는 알고리즘

반복(iterative) 알고리즘	재귀(recursive) 알고리즘

double iter_pow(double x, int n)
{
int i;
double r = 1.0;
for(i=0; i<n; i++)
r = r * x;
return r;
}
	double pow(double x, int n)
	{
	if(n==0) return 1;
	else if(n%2==0) //n이 짝수일 때
	return pow(x*x, n/2);
	else //n이 홀수일 때
	return x*pow(x*x, (n-1)/2);
	}

• 연산시간 : O(n)	• 연산시간 : O(log n)

• 반복 알고리즘은 for를 사용하여, x를 n번 곱한다.
• 해서, 연산시간은 O(n)이다.

◈ **재귀 알고리즘**

- 2의 4승은 2를 4번 곱하는 것보다는 4를 2번 곱하는 것이 더 효율적이다.
- 2의 8승은 4^4으로 계산하고
- 2의 9승은 2 * 4^4으로 계산하고
- 4의 5승은 4 * 16^2으로 계산하고

예	• x = 2, n = 10일 때, pow(x, n) = ? ↓다음처럼 호출된다. • pow(2,10) = pow(4,5) = 4*pow(16,2) = pow(256,1) = 256*pow(256*256, 0)

- 호출을 10번하지 않고, 5번 호출하게 된다. 1/2씩 줄어든다.
- 해서, 연산시간은 O(log n)이 된다.
- n이 커지면, 호출은 엄청나게 줄어든다.

정답 : ②

13. 알고리즘의 표현에서 반복(iteration)과 순환(recursion)에 관한 설명 중 옳은 것은? [2004년 국가 기술고시]

① 반복을 순환으로 대체하게 되면 수행시간이 짧아진다.
② 순환은 선택구조의 표현에 적합하다.
③ 순환을 사용하여 표현된 모든 알고리즘은 순환을 사용하지 않는 방법으로도 표현 가능하다.
④ 동일한 작업을 반복을 사용하여 표현할 경우에 비해 순환을 사용하여 표현하면 알고리즘이 길어진다.
⑤ 순환 알고리즘을 구현하는 함수의 인수, 지역변수 등의 관리를 위해 일반적으로 이용되는 자료구조는 큐이다.

♤ **알고리즘 – 반복과 순환**

- **모든** 알고리즘은 **순환** 또는 **반복** 표현이 가능하다. 여기서 '모든'이라는 것이 중요하다.
- 예를 들면, 이진검색 알고리즘은 '반복 또는 순환' 표현 방식으로 작성할 수 있다.

- 순환은 **선택**구조의 표현에 **적합**하다.**(×)**
  → 순환 알고리즘에서 선택구조를 잘못 표현하면 완료조건에 문제가 발생된다.**(무한루프 발생)**
- 순환 알고리즘에서 선택구조는 반드시 필요한 부분이지만, 그 표현은 매우 신중해야 한다.
- 해서, 순환 알고리즘은 선택구조의 표현에 적합한 것은 아니다.

정답 : ③

14. 재귀(recursion) 알고리즘과 for문 또는 while문을 사용하는 반복(iteration) 알고리즘에 대한 설명 중 가장 적절한 것은? [2021년 군무원 7급]

① 반복 알고리즘은 대체로 재귀 알고리즘에 비해 시간적으로나 공간적으로 매우 비효율적이다.

② 재귀 알고리즘은 반드시 큐(queue)의 개념이 필요하다.

③ 이진탐색을 하는 반복 알고리즘의 최악의 경우 수행시간은 $O(\log_2 n)$이고, 재귀 알고리즘의 최악의 경우 수행시간은 O(n)이다.

④ 재귀 알고리즘은 재귀호출(recursive call)을 사용한 알고리즘으로서 오버헤드(overhead), 즉 부가적인 기억공간이 필요할 수 있다.

♤ **재귀 알고리즘과 반복 알고리즘**

---

① **반복** 알고리즘은 대체로 **재귀** 알고리즘에 비해 시간적으로나 공간적으로 매우 비효율적이다.**(×)**
→ **재귀** 알고리즘은 대체로 **반복** 알고리즘에 비해 시간적으로나 공간적으로 매우 비효율적이다.

② **재귀** 알고리즘은 반드시 **큐**(queue)의 개념이 필요하다.**(×)**
→ **재귀** 알고리즘은 반드시 **스택**의 개념이 필요하다.

③ **이진탐색**을 하는 반복 알고리즘의 최악의 경우 수행시간은 $O(\log_2 n)$이고, **재귀** 알고리즘의 최악의 경우 **수행시간은 O(n)**이다.**(×)**
→ **이진탐색** : **재귀** 알고리즘의 최악의 경우 **수행시간은 O(**$\log_2 n$**)**이다.

정답 : ④

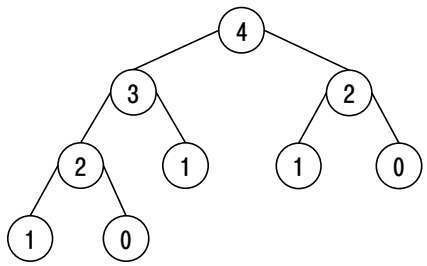
15. 다음은 Fibonacci 수열을 계산하는 C 함수이다. 함수 fibonacci(4)를 호출하였을 때 화면에 출력되는 숫자의 순서로 옳은 것은? [2019년 국가 7급]

```
int fibonacci(int n) {
  printf("%3d", n);
  if(n==0) return 0;
  else if(n==1) return 1;
  else return fibonacci(n-1) + fibonacci(n-2);
}
```

① 4 3 2 2 1 1 0 1 0　　② 4 3 2 1 0 2 1 1 0
③ 4 3 2 1 0 1 2 1 0　　④ 4 3 2 1 0 2 1 0 1

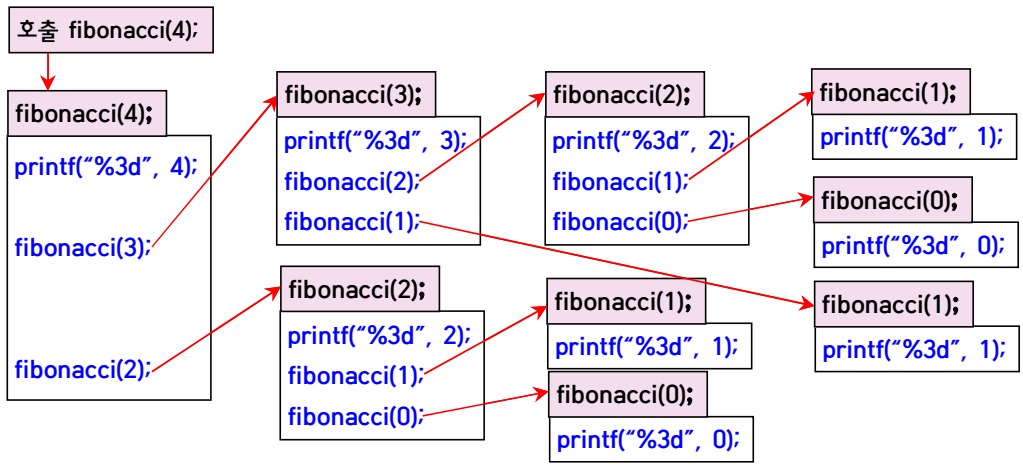
♤ Fibonacci 수열

• 먼저, 출력되는 숫자는 다음 재귀트리를 전위운행하면 된다.(출력문이 먼저 있는 경우)



전위운행(중,좌,우) : 4 3 2 1 0 1 2 1 0

// 재귀호출 수행 과정을 그림으로 나타내면 다음과 같다.



정답 : ③

16. 다음 C 언어 프로그램의 출력 결과는? [2020년 국가 7급]

```
#include <stdio.h>
#define SWAP(x, y, t) (t)=(x), (x)=(y), (y)=(t)
void recursive(char *list, int i, int n)
{
  int j, temp;
  if (i == n)
  {
    for (j = 0; j <= n; j++)
      printf("%c", list[j]);
    printf(" ");
  }
  else
  {
    for (j = i; j <= n; j++)
    {
      SWAP(list[i], list[j], temp);
      recursive(list, i + 1, n);
      SWAP(list[i], list[j], temp);
    }
  }
}
int main()
{
  char list[3] = {'a', 'b', 'c'};
  recursive(list, 0, 2);
}
```

① ab ac bc
② ab bc ca
③ abc acb bac bca cba cab
④ abc acb bca bac cab cba

♤ **재귀호출을 이용한 순열(permutation)을 구하기**

- 먼저, 주어진 프로그램은 문자열 **abc**에 대한 **순열(permutation)**을 구하는 문제이다.
- **abc = {abc, acb, bac, bca, cba, cab}**

// 문자열 abc에 대한 순열 알고리즘

① 인덱스 0번째 원소를 0번째부터 n-1번째까지 위치를 바꾼다.

• 문자열 abc에 대해 이 과정을 진행하면 3가지 종류 **abc, bac, cba**가 된다.

↓

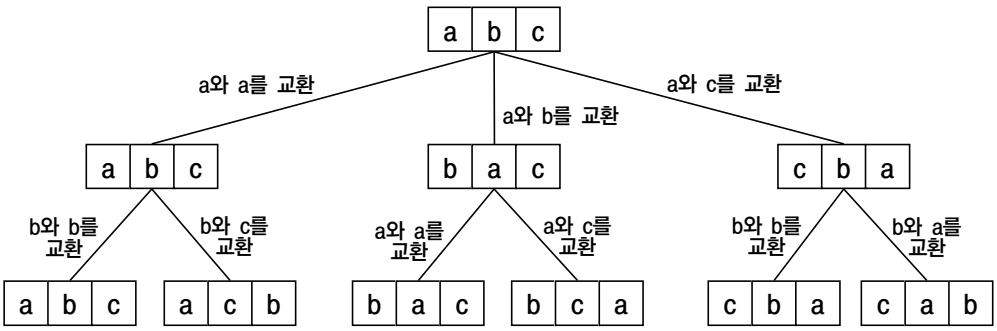
• abc : 첫 번째[0] 원소 a를 첫 번째[0] 원소 a와 바꾼 것**(실제로는 바뀐 것이 없음)**
• bac : 첫 번째[0] 원소 a를 두 번째[1] 원소 b와 바꾼 것
• cba : 첫 번째[0] 원소 a를 세 번째[2] 원소 c와 바꾼 것

② 위의 ①번 과정의 결과에서 인덱스 1번째 원소를 1번째부터 n-1번째까지 위치를 바꾼다.

• abc : 두 번째[1] 원소 b를 두 번째[1] 원소 b와 바꾼 것(실제로는 바뀐 것이 없음) • acb : 두 번째[1] 원소 b를 세 번째[2] 원소 c와 바꾼 것
• bac : 두 번째[1] 원소 a를 두 번째[1] 원소 a와 바꾼 것(실제로는 바뀐 것이 없음) • bca : 두 번째[1] 원소 a를 세 번째[2] 원소 c와 바꾼 것
• cba : 두 번째[1] 원소 b를 두 번째[1] 원소 b와 바꾼 것(실제로는 바뀐 것이 없음) • cab : 두 번째[1] 원소 b를 세 번째[2] 원소 a와 바꾼 것

③ 이러한 과정을 n-1번 진행합니다.

↓
↓ 트리구조로 나타내면 다음과 같다.
↓



↓ 원시코드(핵심부분)

```
for (int i = start; i <= end; i++)          //start=0, end=2인 경우
{
   swap(array[start], array[i]);            //시작부터 끝까지 모든 문자 교환
   permutation(array, start+1, end);        //시작 인덱스만 1 증가
   swap(array[start], array[i]);            //시작부터 끝까지 모든 문자 교환
}
```

• 주어진 순열 알고리즘의 시간복잡도는 **O(n!)**이다.

정답 : ③