자료구조 연습문제 2장.

01. 팩토리얼을 계산하는 순환호출함수 factorial에 매개변수로 5를 주었다면, 최대 몇 개의 factorial 함수의 활성레코드가 동시에 존재할 수 있는가??

factorial(5)는 factorial(4)를
factorial(4)는 factorial(3)을
factorial(3)은 factorial(2)를
factorial(2)는 factorial(1)을 호출하고
factorial(1)부터 재귀호출 함수를 종료하므로
최대 5개가 동시에 존재한다.
∴ 5.

02. 순환호출을 하였을 경우에 활성레코드들이 생성되는 위치는 어디인가?

스택에 생성된다.
∴ (4)

03. 다음 중 활성레코드에 생성되지 않는 것은 무엇인가?

복귀주소가 시스템 스택에 생성되고,
호출되는 함수를 위한 매개변수 값과 지역변수를
할당한다.
∴ (4)

04. 재귀적 함수가 호출할 수 있는 순환호출의 개수는?

스택이 허용하는 한도까지 활성레코드가
존재할 수 있다.
∴ (3)

05. 다음의 순환호출 함수에서 출력되는 값은 무엇인가?

```
int recursive(int n) {
  if (n==1) return 0;
  return n+ recursive(n);
}
```

종료조건 1이 들어오지 않는다면,
계속 자기자신을 다시 호출하므로
함수가 끝나지 않는다.

06. 다음의 순환호출 함수에서 출력되는 값은 무엇인가?

```
int recursive(int n) {
  printf("recursive(%d)\n", n);
  return n* recursive(n-1);
}
```

함수가 끝나지 않고, 계속 새로운 함수를
호출하므로 무한히 반복된다.

07. 다음함수를 sum(5)로 호출하였을 때,
화면에 출력되는 내용과 함수의 반환값을 구하라.

```
int sum(int n) {
  printf("%d\n", n);
  if (n<1) return 1;
  else return (n+ sum(n-1));
}
```

[출력]
5
4
3
2
1
0

[반환값]
16

08. 다음 함수를 recursive(5)로 호출하였을 때, 화면에 출력되는 내용과 함수의 반환값을 구하여라.

```
int recursive(int n) {
  printf("%d\n", n);
  if(n<1) return 2;
  else return (2* recursive(n-1)+1);
}
```

[출력]

5
4
3
2
1
0

[반환값]

95

09. 다음 함수를 recursive(10)으로 호출하였을 때, 화면에 출력되는 내용과 함수의 반환값을 구하라.

```
int recursive(int n){
  printf("%d\n",n);
  if(n<1) return -1;
  else return (recursive(n-3)+1);
}
```

[출력]

10
7
4
1
-2

[반환값]

3

10. 다음 함수를 recursive(5)로 호출하였을 때, 화면에 출력되는 내용을 쓰시오.

오류난듯 !! 'void'로 해야함.

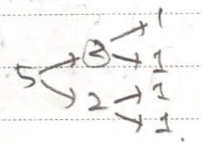
```
int recursive(int n){
  if(n != 1) recursive(n-1);
  printf("%d\n", n);
}
```

[출력]

1
2
3
4
5.

11. 다음 함수에서 asterisk(5)를 호출하였을 때 출력되는 *의 갯수는 ??

```
void asterisk(int i){
  if(i>1){
    asterisk(i/2);
    asterisk(i/2);
  }
  printf("*");
}
```



[출력]

****

∴ 7개.

12. 다음과 같은 함수를 만들고 "recursive"를 입력한 다음, 엔터키를 눌렀다면 화면에 출력되는 것은?

```
unknown() {
  int ch;
  if((ch=getchar()) != '\n')
    unknown();
  putchar(ch);
}
```

[출력]

eviscrucer

13. 다음을 계산하는 순환적인 프로그램을 작성하시오.

$1+2+\cdots+n$

```
int sum(int n){
  if(n==1) return 1;
  return n+sum(n-1);
}
```

14. 다음을 계산하는 순환적인 프로그램을 작성하시오.

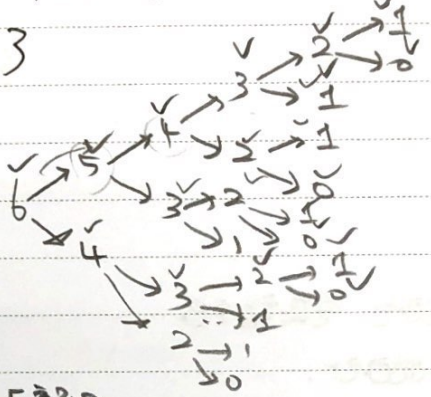
$1+\frac{1}{2}+\cdots+\frac{1}{n}$

```
double sum(int n){
    if(n==1) return 1;
    return (double)1/n + sum(n-1);
}
```

15. 순환호출되는 것을 이해하기 위하여 fib함수를 다음과 같이 바꾸어서 실행하여 보라.

fib(6)을 호출할 때 화면에 출력되는 내용을 쓰시오.

```
int fib(int n){
  printf("fib(%d) is called\n",n);
  if(n==0) return 0;
  if(n==1) return 1;
  return (fib(n-1)+fib(n-2));
}
```



[출력]

fib(6) is called
fib(5) is called
fib(4) is called
fib(3) is called
fib(2) is called
fib(1) is called
fib(0) is called
fib(1) is called
fib(2) is called
fib(1) is called
fib(0) is called
fib(3) is called
fib(2) is called
fib(1) is called
fib(0) is called
fib(1) is called
fib(4) is called
fib(3) is called
fib(2) is called
fib(1) is called

→ fib(0) is called
fib(1) is called
fib(2) is called
fib(1) is called
fib(0) is called.

16. 다음의 순환적인 프로그램을 반복 구조를 사용한 비순환적 프로그램으로 바꾸시오.

```
int sum(int n) {
  if (n == 1) return 1;
  else return (n + sum(n-1));
}
```

```
int sum(int n) {
  int tmp = 0;
  for(int i = 1; i <= n; i++)
    tmp += i;
  return tmp;
}
```

17. 이항계수를 계산하는 순환함수와 반복함수를 구현하라.

$$nCk = \begin{cases} n-1Ck-1 + n-1Ck & \text{if } 0<k<n \\ 1 & \text{if } k=0 \text{ or } k=n \end{cases}$$

[순환함수]

```
int Combination(int n, int k) {
  if (k == 0 || k == n)
    return 1;
  return Combination(n-1, k-1) + Combination(n-1, k);
}
```

[반복함수]

```
int Combination(int n, int k) {
  int t1 = 1, t2 = 1, t3 = 1;
  for(int i = 1; i <= n; i++)
    t1 *= i;
  for(int i = 1; i <= k; i++)
    t2 *= i;
  for(int i = 1; i <= n-k; i++)
    t3 *= i;
  return t1 / (t2 * t3);
}
```

18. Ackerman 함수는 다음과 같이 순환으로 정의된다.

$A(0,n) = n+1$;

$A(m,0) = A(m-1, 1)$

$A(m,n) = A(m-2, A(m, n-1))$ $m, n \geq 1$

"m-1"로 바꾸어야 한다.

(a) A(3,2)와 A(2,3)의 값을 구하시오.

① $A(3,2) = 29$

② $A(2,3) = 9$.

(b) Ackerman 함수를 구하는 순환적인 프로그램을 작성하시오.

```
int Ackermann(int m, int n) {
  if(m == 0)
    return n+1;
  else if(n == 0)
    return Ackermann(m-1, 1);
  else
    return Ackerman(m-1, Ackerman(m, n-1));
}
```

(c) 위 순환적인 프로그램을 for, while, do-while 중 한 반복문을

19.

반복문을 피보나치 수열 프로그램은
n 번째 피보나치 값을 구한다면,
(n-1)번의 연산이 필요하다.
하지만, 순환적인 피보나치 수열 프로그램은
n번째 피보나치 값을 구하기 위해
(n-1)번째, (n-2)번째 피보나치 함수를
다시 호출하고, 또 이 호출된 함수가
2개씩 함수를 더 호출하여
기하급수로 호출 횟수가 증가한다.
따라서, 훨씬 더 많이 걸린다.

20.

(1) 크기 n인 문제가 크기가 1씩 감소한다.

(2). 크기 n인 문제가 크기가 (n-1)인 문제
2개로 한 번의 연산으로 나뉘어진다.

21.

```
void flood-fill (int x, int y) {
  if (read-pixel (x,y) == WHITE) {
    write-pixel (x,y, BLACK);
    flood-fill (x+1, y);
    flood-fill (x-1, y);
    flood-fill (x, y+1);
    flood-fill (x, y-1);
  }
}
```