ch2

1. 최대 5개의 활성 레코드가 존재할 수 있다.
2. (4)
3. (4)
4. (3)
5. (3)
6. 순환호출을 할때 파라미터의 값이 줄어들지 않았다.

return n*recursive(n);-> return n*recursive(n-1)

7. 순환호출을 끝내는 문장이 빠져있다.

```
if( n==1 ) return 0;
```

8.

```
5
4
3
2
1
0
함수의 반환값=15
```

9.

```
5
4
3
2
1
0
함수의 반환값=95
```

10.

```
10
7
4
1
-2
함수의 반환값=3
```

11.

```
1
2
3
4
5
```

12. 7개

13. 오타: putchar()->putchar(ch)
evisrucer

14.

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

15.

```
double recursive1(int n)
{
        if( n==1 ) return 1.0;
        else return (1.0/n)+recursive1(n-1);
}
```

16.

```
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
```

17.

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

18. 이항계수

```
int bin(int n, int k)
{
        if( k==0 || n==0 )
                return 1;
        else
```

```
        return bin(n-1,k-1)+bin(n-1,k);
}
```

19. Ackermann 함수는 다음과 같이 순환적으로 정의된다.

$$A(0,n) = n+1;$$
$$A(m,0) = A(m-1,1)$$
$$A(m,n) = A(m-1,A(m,n-1)) \qquad m,n \geq 1$$

(a) $A(3,2)$와 $A(2,3)$의 값을 구하시오.

A(3,2)= 29

A(2,3)=9

(b) Ackermann 함수를 구하는 순환적인 프로그램을 작성하시오.

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

(c) 위의 순환적인 프로그램을 for, while, do와 같은 반복구조를 사용한 비순환적 프로그램으로 바꾸시오.

```
int ack2(int m, int n)
{
    while (m != 0){
        if (n == 0)
            n = 1;
        else
            n = ack2(m, n-1);
        m = m - 1;
    }
    return n+1;
}
```

20. 본문의 순환적인 피보나치 수열 프로그램과 반복적인 피보나치 수열 프로그램의 수행 시간을 측정하여 비교하라. 어떤 결론을 내릴 수 있는가?

-> 순환적인 피보나치 프로그램은 거의 사용하지 못할 정도로 많은 수행시간을 요구한다.

21. 순환호출에서는 순환호출을 할때마다 문제의 크기가 작아져야 한다.

(1) 팩토리얼 계산 문제에서 순환호출이 일어날 때마다 문제가 어떻게 작아지는가?

-> n! = n * (n-1)! 으로 n을 곱한 다음 (n-1)!을 구한다.

(2) 하노이의 탑에서 순환호출이 일어날 때마다 문제의 어떻게 작아지는가?

-> 하나의 원판은 이동시킨 다음에 나머지 원판에 대하여 순환호출을 한다.