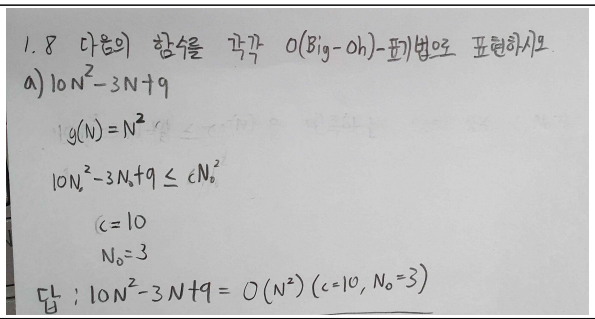
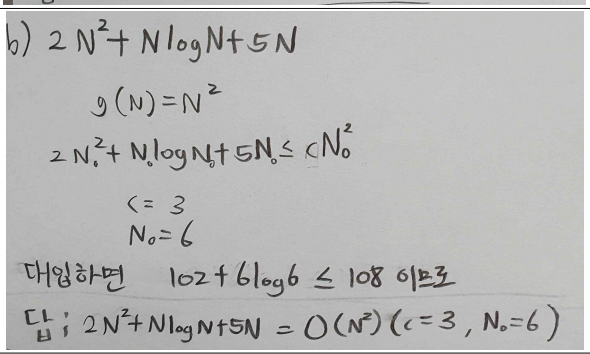
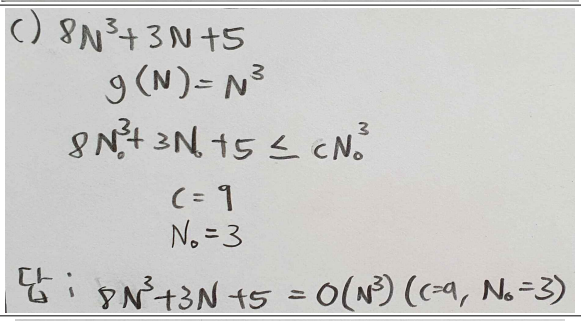
자료구조				과제 1	
학년	2	학번	20180283	이름	김주호

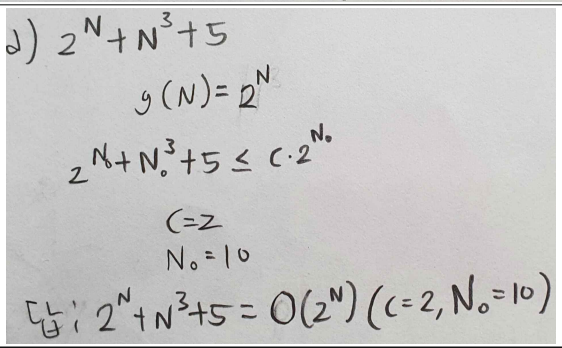
(1) 연습문제 풀이 사진 촬영

- 1.8









- 1.10

1.10 다음의 루프의 수행시간을 θ-표기법으로 표현하시오.

```
01  int s =0;
02  for(int i=0; i < N;i++)
03      for(int j=0;j < N;j++)
04          s += N;
```

행	단계수
1	1
2	N+1
3	N(N+1)
4	N^2
총합	$2N^2+2N+2$

θ-표기법이므로 모든 $N \geq N_0$에 대해 $c_1 g(N) \geq f(N) \geq c_2 g(N)$이 성립하여야 하고 O-표기와 Ω-표기가 같아야하기때문에 O-표기와 Ω표기를 구하면 O-표기에서는 정의를 만족하는 가장낮은 차수 N^2이 $g(N)$이

O-표기; $2N^2+2N+2 \leq cN^2$ 이므로 이를 만족하는 $c=3$, $N_0=3$이 존재하므로

O-표기는 $2N^2+2N+2 = O(N^2)$ $(c=3, N_0=3)$.

Ω-표기에서도 정의를 만족하는 가장 높은 차수가 N^2이므로 N^2가 $g(N)$이 된다.

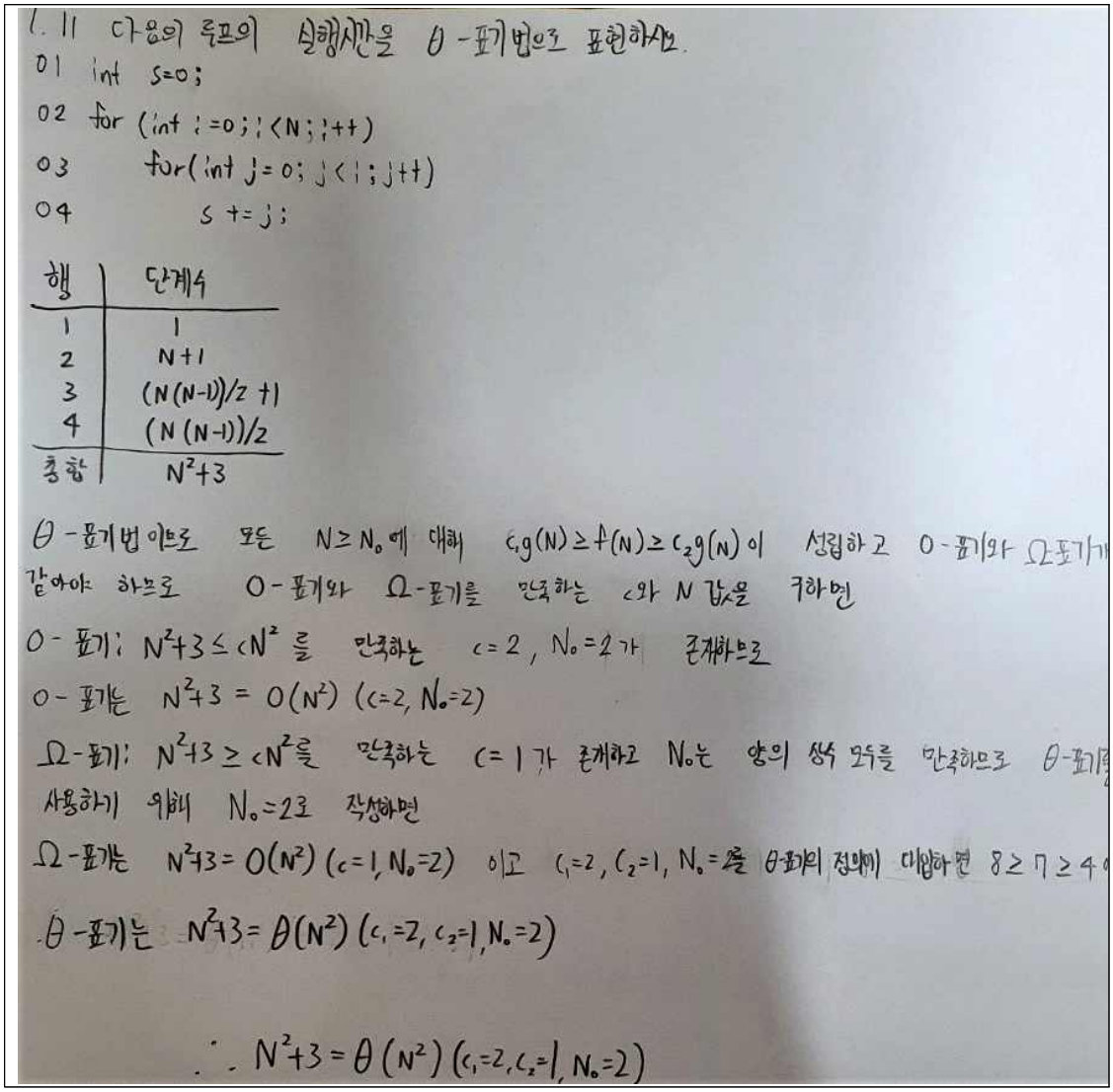
Ω-표기; $2N^2+2N+2 \geq cN^2$을 만족해야 하고 이를 만족하는 $c=2$가 존재하고 N_0는 0보다 큰 양의 상수 모두를 만족하지만 θ-표기 시 N_0의 값이 O-표기, Ω-표기에서 같아야 하므로 $N_0=3$이 되어

Ω-표기는 $2N^2+2N+2 = \Omega(N^2)$ $(c=2, N_0=3)$ 이고 $c_1=3$, $c_2=2$, $N_0=3$을 θ-표기 정의에 대입하면 $2n \geq 2 \geq 18$이

θ-표기는 $2N^2+2N+2 = \theta(N^2)$ $(c_1=3, c_2=2, N_0=3)$

$$\therefore 2N^2+2N+2 = \theta(N^2) \quad (c_1=3, c_2=2, N_0=3)$$

- 1.11



1.11 다음의 루프의 실행시간을 θ-표기법으로 표현하시오.

```
01 int s=0;
02 for (int i=0; i<N; i++)
03     for(int j=0; j<i; j++)
04         s += j;
```

행	단계수
1	1
2	$N+1$
3	$(N(N-1))/2+1$
4	$(N(N-1))/2$
총합	N^2+3

θ-표기법이므로 모든 $N \geq N_0$에 대해 $c_1 g(N) \geq f(N) \geq c_2 g(N)$이 성립하고 O-표기와 Ω-표기가 같아야 하므로 O-표기와 Ω-표기를 만족하는 c와 N 값을 구하면

O-표기: $N^2+3 \leq cN^2$를 만족하는 $c=2$, $N_0=2$가 존재하므로

O-표기는 $N^2+3 = O(N^2)$ $(c=2, N_0=2)$

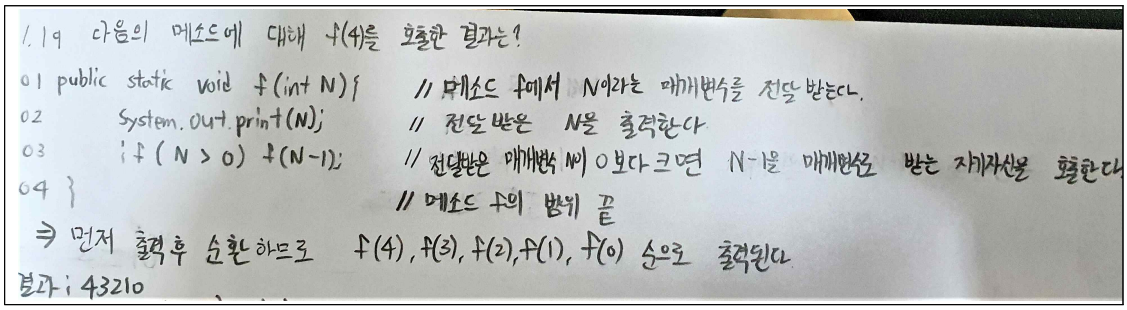
Ω-표기: $N^2+3 \geq cN^2$를 만족하는 $c=1$가 존재하고 N_0는 양의 정수 모두를 만족하므로 θ-표기를 사용하기 위해 $N_0=2$로 작성하면

Ω-표기는 $N^2+3 = O(N^2)$ $(c=1, N_0=2)$ 이고 $c_1=2$, $c_2=1$, $N_0=2$를 θ표기의 정의에 대입하면 $8 \geq 7 \geq 4$

θ-표기는 $N^2+3 = \theta(N^2)$ $(c_1=2, c_2=1, N_0=2)$

$\therefore N^2+3 = \theta(N^2)$ $(c_1=2, c_2=1, N_0=2)$

- 1.19



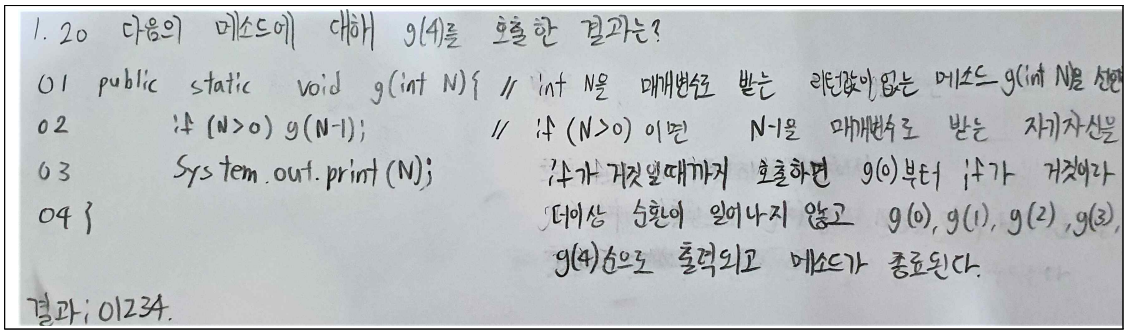
1.19 다음의 메소드에 대해 f(4)를 호출한 결과는?

```
01 public static void f(int N){   // 메소드 f에서 N이라는 매개변수를 전달받는다.
02     System.out.print(N);       // 전달받은 N을 출력한다
03     if ( N > 0) f(N-1);        // 전달받은 매개변수 N이 0보다 크면 N-1을 매개변수로 받는 자기자신을 호출한다
04 }                              // 메소드 f의 범위 끝
```

⇒ 먼저 출력 후 순환하므로 f(4), f(3), f(2), f(1), f(0) 순으로 출력된다

결과 : 43210

- 1.20

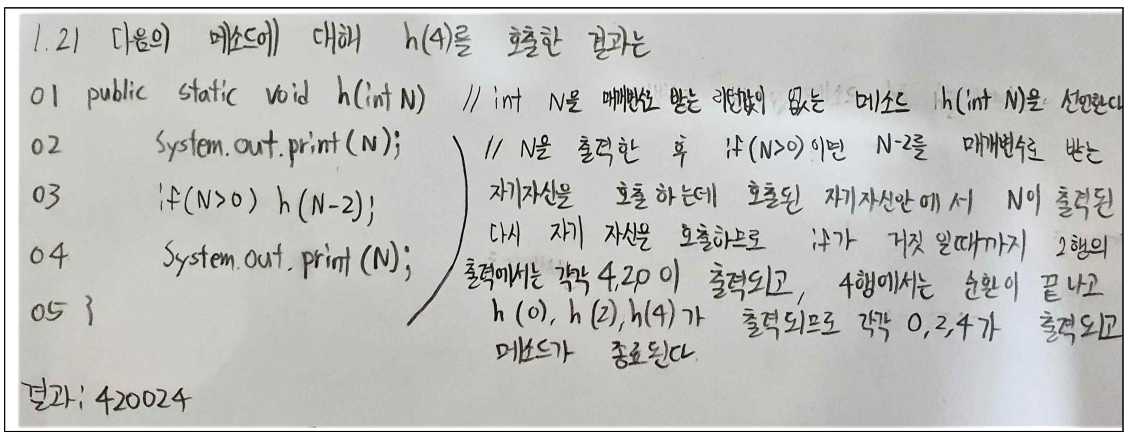


1.20 다음의 메소드에 대해 g(4)를 호출한 결과는?

```
01 public static void g(int N){ // int N을 매개변수로 받는 리턴값이 없는 메소드 g(int N)을 선언
02     if (N>0) g(N-1);         // if (N>0) 이면 N-1을 매개변수로 받는 자기자신을
03     System.out.print(N);     // if가 거짓일때까지 호출하면 g(0)부터 if가 거짓이라
04 }                            // 더이상 순환이 일어나지 않고 g(0), g(1), g(2), g(3),
                                // g(4)순으로 출력되고 메소드가 종료된다.
```

결과: 01234.

- 1.21

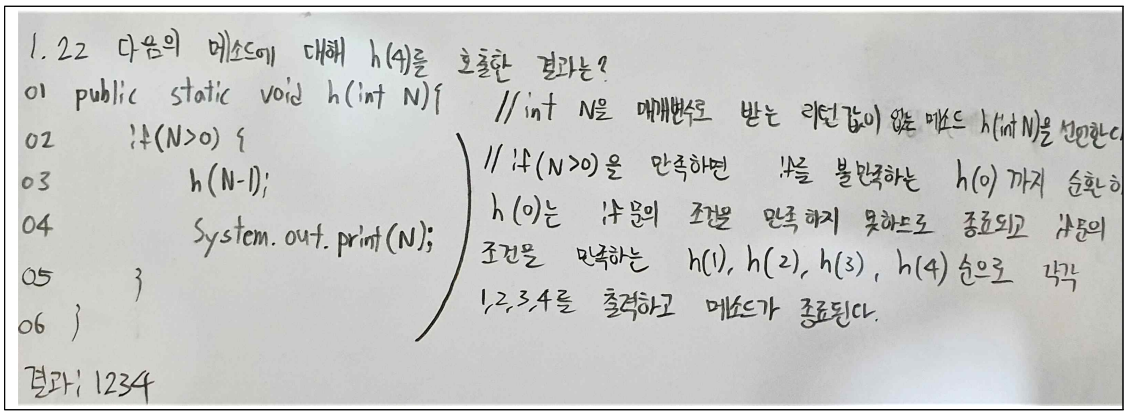


1.21 다음의 메소드에 대해 h(4)를 호출한 결과는

```
01 public static void h(int N) // int N을 매개변수로 받는 리턴값이 없는 메소드 h(int N)을 선언한다
02     System.out.print(N);    // N을 출력한 후 if (N>0) 이면 N-2를 매개변수로 받는
03     if(N>0) h(N-2);         // 자기자신을 호출하는데 호출된 자기자신안에서 N이 출력된
04     System.out.print(N);    // 다시 자기 자신을 호출하므로 if가 거짓 일때까지 2행의
05 }                           // 출력에서는 각각 4,2,0 이 출력되고, 4행에서는 순환이 끝나고
                               // h(0), h(2), h(4)가 출력되므로 각각 0,2,4가 출력되고
                               // 메소드가 종료된다.
```

결과: 420024

- 1.22

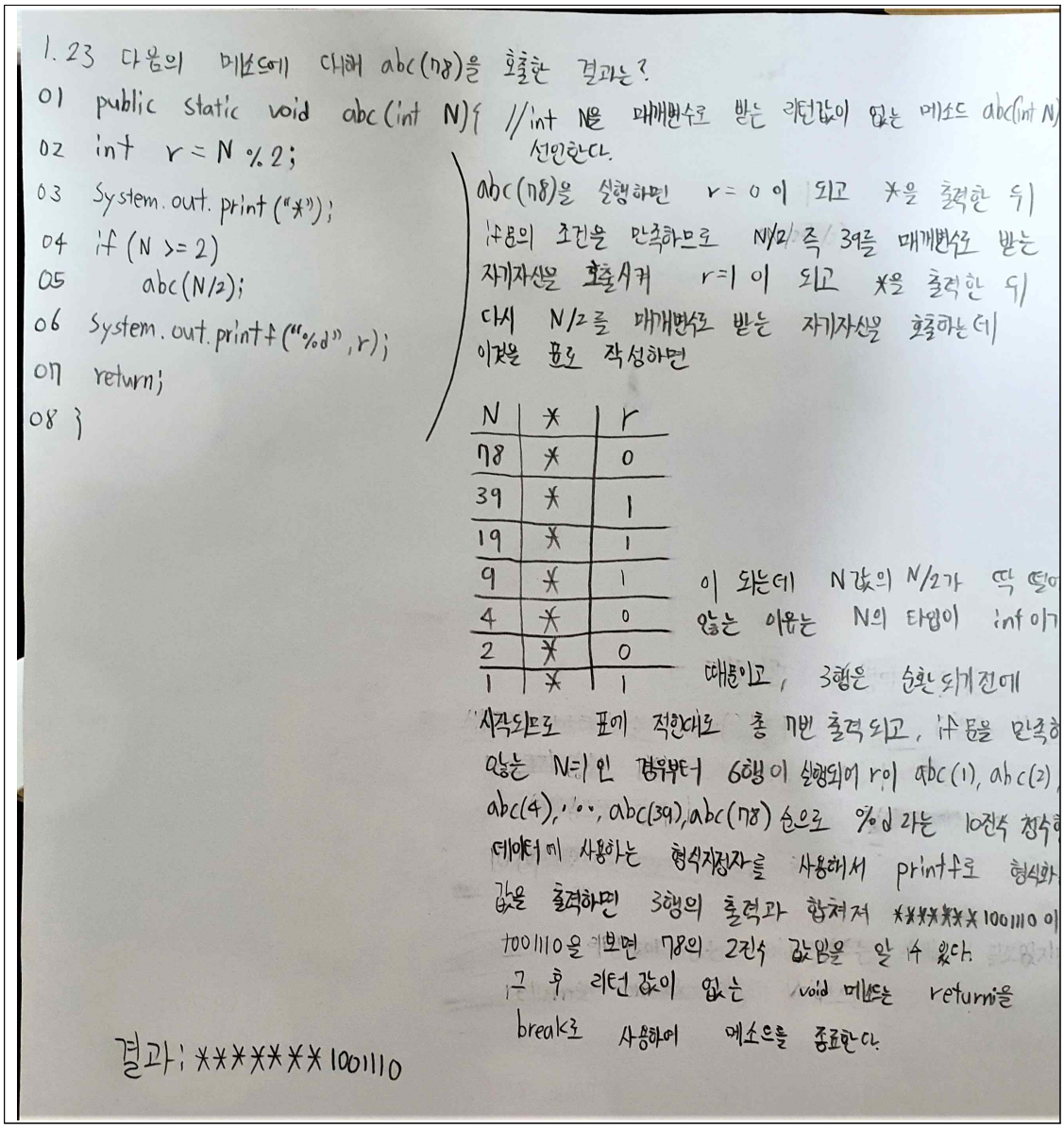


1.22 다음의 메소드에 대해 h(4)를 호출한 결과는?

```
01 public static void h(int N){ // int N을 매개변수로 받는 리턴값이 없는 메소드 h(int N)을 선언한다
02     if(N>0) {                // if (N>0)을 만족하면 if를 불만족하는 h(0) 까지 순환하
03         h(N-1);              // h(0)는 if문의 조건을 만족하지 못하므로 종료되고 if문의
04         System.out.print(N); // 조건을 만족하는 h(1), h(2), h(3), h(4) 순으로 각각
05     }                        // 1,2,3,4를 출력하고 메소드가 종료된다.
06 }
```

결과: 1234

- 1.23



1.23 다음의 메소드에 대해 abc(78)을 호출한 결과는?

```
01 public static void abc(int N){ //int N을 매개변수로 받는 리턴값이 없는 메소드 abc(int N)
02 int r = N % 2;
03 System.out.print("*");
04 if (N >= 2)
05     abc(N/2);
06 System.out.printf("%d", r);
07 return;
08 }
```

선언한다.

abc(78)을 실행하면 r = 0 이 되고 *을 출력한 뒤 if문의 조건을 만족하므로 N/2 즉 39를 매개변수로 받는 자기자신을 호출시켜 r=1 이 되고 *을 출력한 뒤 다시 N/2를 매개변수로 받는 자기자신을 호출하는데 이것을 표로 작성하면

N	*	r
78	*	0
39	*	1
19	*	1
9	*	1
4	*	0
2	*	0
1	*	1

이 되는데 N값의 N/2가 딱 떨어지지 않는 이유는 N의 타입이 int이기 때문이고, 3행은 순환되기전에 시작되므로 표에 적힌대로 총 7번 출력되고, if문을 만족하지 않는 N=1인 경우부터 6행이 실행되어 r이 abc(1), abc(2), abc(4), ..., abc(39), abc(78) 순으로 %d라는 10진수 정수형 데이터에 사용하는 형식지정자를 사용해서 printf로 형식화된 값을 출력하면 3행의 출력과 합쳐져 *******1001110 이 되고 1001110을 계산하면 78의 2진수 값임을 알 수 있다.

그 후 리턴 값이 없는 void 메소드는 return을 break로 사용하여 메소드를 종료한다.

결과: *******1001110

-1.24

1.24 다음의 메소드에 대해 test("11010011", 4)가 리턴하는 값은 ?

```
01 public static int test(String s, int last) {
02     if (last < 0) {
03         return 0;
04     }
05     if (s.charAt(last) == '0') {
06         return 2 * test(s, last-1);
07     }
08     return 1+2*test(s, last-1);
09 }
```

// 반환값이 int인 test 메소드의 리턴값은 매개변수로 ("11010011", 4)를 전달받았을 때의 결과값

우선 2행~4행의 구문은 last 값이 0보다 작으면 0을 리턴하는 구문이고, 5행~7행의 구문은 전달받은 s의 last번째의 문자가 '0'일 경우 return 2*test(s, last-1); 을 실행하여 자기자신을 호출하는 구문이고, 8행은 위 조건에 속하지 않을 경우 실행되는 반환값에 재귀가 있는 구문이다.

처음에 주어진 ("11010011", 4)를 집어 넣으면 문자열 "11010011"의 4번째 문자가 0이므로 5행 구문이 실행되는데 이 값을 도출하기 위해서는 test("11010011", 3)의 값이 필요하게 되어 return 값이 0일 경우까지 자기자신을 호출하는 것을 표로 표현하면

last	s.charAt(last)	실행문	결과
4	0	5~7행실행	26 ↑
3	1	8행실행	13 ↑
2	0	5~7행 실행	6 ↑
1	1	8행 실행	3 ↑
0	1	8행 실행	1 ↑
-1	X	2행실행	0

이므로 다시 재귀된 값을 집어 넣어주면서 나오는 결과는 26이고 이 결과는 0~4번째 문자열을 2진수에서 10진수로 변환한 값이라는 것을 알 수 있다.

결과: 26

-1.25

1.25 다음의 메소드가 무엇을 계산하는지 설명하시오

```
01  public  static  void  t(int  N){
02      if(N>0){
03          t(N/2);
04          System.out.print(N%2);
05      }
06  }
```

// N을 전달받아 if문을 만족하는 가장작은 N까지 자기 자신을 호출하고 가장작은 N에 대한 출력부터 처음 전달한 N값까지 출력하는 메소드이다. 예를 N=10일 경우와 N이 17일 경우로 들어서 표로 표현하면

N=10인 경우

N	t(N/2)	N%2
10	5	0
5	2	1
2	1	0
1	0	1

N=17인 경우

N	t(N/2)	N%2
17	8	1
8	4	0
4	2	0
2	1	0
1	0	1

위와 같이 N=10인경우는 가장 마지막에 순환된 N=1인경우부터 N=10인경우까지의 출력이 1010이고 N=17인 경우는 같은 원리로 10001인 것을 볼 수 있다. 지금 문제의 메소드는 결국 전달받은 수를 2로 나누고, 나머지를 구하면서 가장 마지막에 순환된 메소드 부터 출력되므로 입력받은 N을 2진수로 변환해주는 10진수 → 2진수 변환 메소드임을 알 수 있다.

결과; 전달 받은 10진수 N을 2진수로 변환해주는 메소드

(2) Hanoi tower 구동 감상문

하노이 탑에 대한 코드와 여러 정의를 보면서 하노이 탑은 1번 막대에 있는 n개의 원판을 3번 막대로 1번 막대에 원판이 꽂혀있던 순서대로 옮기기 위해 원판을 총 2^n-1번 이동하는 대표적인 재귀 함수라는 것을 알게 되었으며, 만약 재귀 함수를 모르는 상태에서 원판 이동 수가 2^n-1인 하노이 탑을 코딩했더라면 어느 위치로 어느 원판이 이동하는지를 작성하고 이동한 값을 저장해주는 변수들을 각각 추가로 작성하고 코드가 복잡해지는 것을 생각해봤더니 재귀 함수가 중복되는 코딩의 작성을 줄여주어 프로그래머가 더욱 편하고 간결하게 코드를 작성할 수 있게 해주는 것을 알게 되었고 이 방법을 알아낸 분들에게 감사함이 느껴졌다. 솔직히 말해 이전에는 코딩 시 재귀 함수가 꼭 필요하다고 생각하고 있지 않았지만, 하노이 탑 같은 경우에는 재귀 함수가 없으면 굉장히 힘들다는 것을 알게 되었고 이번 기회를 통해 재귀 함수에 대해서 조금 더 알게 되어 혼자 코딩할 때도 사용할 수 있을 것 같다.

(3) 피보나치 50번째 값이 이상하게 나오는 이유

입력이 10, 20, 30인 경우까지는 출력 값이 계산되는 속도가 크게 다르지 않다고 느껴졌다. 40부터는 재귀가 반복보다 살짝 느리다는 정도의 차이가 느껴졌고, 50을 입력했을 시에는 실행 시간이 확연히 다르게 느껴져 출력까지의 시간을 System.currentTimeMillis() 함수를 구해서 구해보았더니 재귀 함수는 약 61초, 반복 함수는 약 2초의 시간이 걸렸다. 이 결과로 보아 피보나치 수열에서 재귀 함수를 사용하는 경우에 속도 감소 및 실행 시간의 증가는 반복문처럼 반복문 내부 코드를 입력값까지 반복 실행하는 것이 아닌 자기 자신을 입력값만큼 다시 호출하고 계산해주어야 하기에 실행 시간이 입력값의 크기에 대해 반복 함수보다 더 큰 영향을 받는다는 것을 알 수 있다. 그리고 n의 값에 50을 입력하면 -298632863이라는 음수가 출력되는 것을 볼 수 있다. 양의 피보나치 수를 구하는 코드인데 음수의 출력이 나오는 이유는 n=47부터는 int의 표현 범위인 -2147483648~2147483647을 초과하므로 오버플로우가 발생하기 때문이다. 결론적으로 int의 범위를 초과하여 발생한 오버플로우에 의해 50에 대한 피보나치 값은 예상하지 못한 결과가 출력된다.