



# SW 역량 테스트 대비반 - 6회차

***2020년 08월 19일***

# 오늘 할 내용

구분		<div>동적표 활용 알고리즘 설계</div>	
1주차 (07/08)			
2주차 (07/15)			
3주차 (07/22)			
4주차 (07/29)			
5주차 (08/05)			
6주차 (08/12)	분할과 정복 강의 및 실습		
7주차 (08/19)	동적 계획법 강의 및 실습		
8주차 (08/26)	그래프 알고리즘 강의 및 실습		
(옵션) 9주차 ~	실전 감각 기르기 (실전 기출 기반 문제풀이 집중반)		

# 동적 표

동적표란 먼저 표의 일부를 어떤 값으로 채운 후, 나머지 칸들은 주변의 값들을 참조하여 동적으로 채워지는 표

① 첫 번째 칸을 1로 채운다.

$D$	1									
	1	2	3	4	5	6	7	8	9	10

②  $i$  번째 칸은  $(i-1)+i$  로 채운다. (진하게 표시된 부분)

$D$	1	3	6	10	15	21	28	36	45	55
	1	2	3	4	5	6	7	8	9	10

1~N까지의 합

# 동적 표

## 피보나치 수열의 재귀 함수 구현

$$f(n) = \begin{cases} 1 & (n \leq 2) \\ f(n-1) + f(n-2) & (n > 2) \end{cases}$$

입력값	실행시간(초)	입력값	실행시간(초)
40	0.435	48	19.163
41	0.685	49	31.064
42	1.083	50	50.210
43	1.739	51	80.838
44	2.827	52	129.340
45	4.551	53	209.532
46	7.272	:	
47	11.888	100	대략 26,241 년

```
#include <stdio>
int f(int n)
{
    if (n <= 2)
        return 1;
    return f(n - 1) + f(n - 2);
}
int main()
{
    int n;
    scanf("%d", &n);
    printf("%d", f(n));
    return 0;
}
```

# 동적표

## 피보나치 수열의 동적표 구현 (하향식)

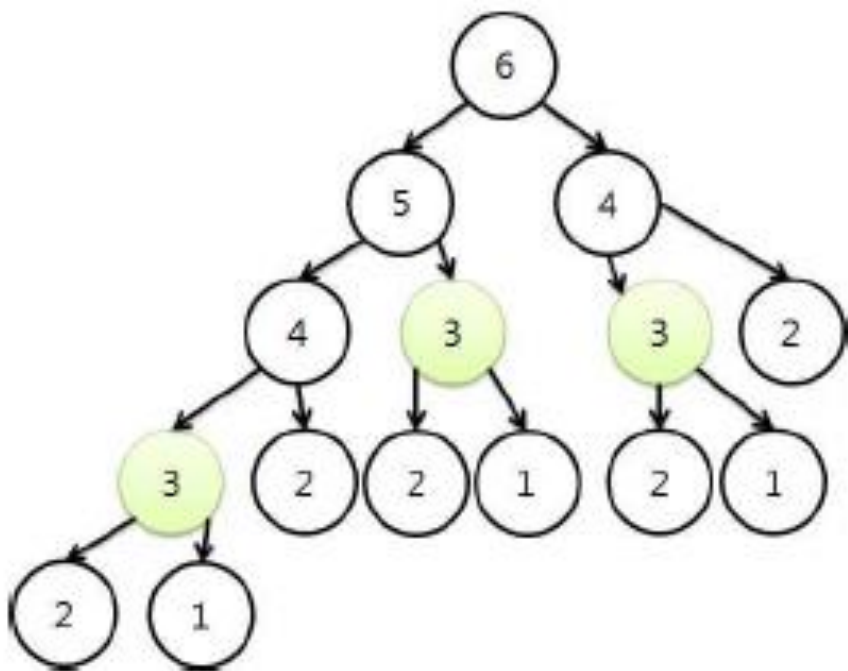
$DT[n]$  =  $n$ 번째 피보나치 수

입력값	실행시간(초)	입력값	실행시간(초)
40	0.000	48	0.000
41	0.000	49	0.000
42	0.000	50	0.000
43	0.000	51	0.000
44	0.000		
45	0.000	100	0.000
46	0.000	:	
47	0.000	50,000	0.002

```
#include <stdio>
int DT[100001];
int f(int n)
{
    if (n <= 2)
        return 1;
    if (!DT[n]) DT[n] = f(n - 1) +
f(n - 2);
    return DT[n];
}
int main()
{
    int n;
    scanf("%d", &n);
    printf("%d\n", f(n));
    return 0;
}
```

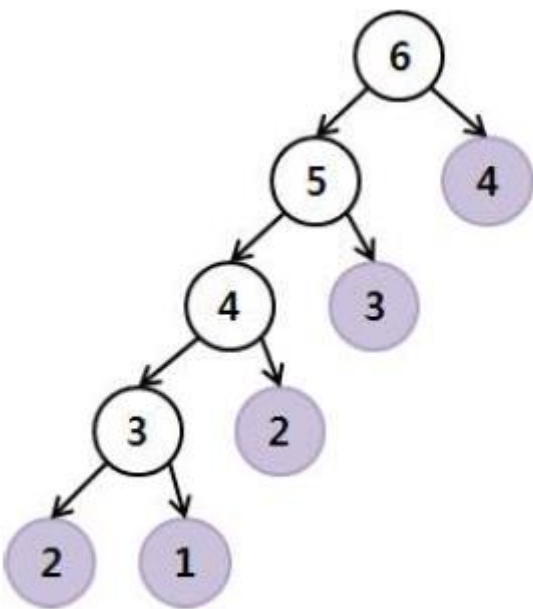
# 동적표

재귀함수 이용



VS

동적표 이용



$f(6)$	$f(5)$	$f(4)$	$f(3)$	$f(2)$	...
1회	1회	2회	3회	5회	...

$f(6)$	$f(5)$	$f(4)$	$f(3)$	$f(2)$	...
1회	1회	1회	1회	1회	...

# 동적표 - 하향식 설계 (메모이제이션)

하향식 설계의 기본은 위에서 아래로 내려가는 것이다. 즉,  $n$  부터 1 의 방향으로 진행해 가면서 해를 구하는 설계방법을 말한다

문) 1부터 N까지의 합을 하향식 설계 동적표를 이용하여 구현하라

100

5050

[illegible]

# 동적표 - 하향식 설계 (메모이제이션)

답안

```
#include <iostream>
using namespace std;
```

```
int DT[101];
```

```
int f(int n)
```

```
{
    if (n == 1) return 1;
    if (!DT[n]) DT[n] = f(n - 1) + n;
    return DT[n];
}
```

```
int main()
```

```
{
    int n;
    cin >> n;
    cout << f(n) << endl;

    return 0;
}
```

기존 재귀함수 호출에 중간값  
저장을 위한 동적표를 추가





# 동적표 - 상향식 설계(동적계획법)

상향식 설계는 하향식 설계와는 반대로 1부터  $n$ 의 방향으로 해를 구해가는 알고리즘 설계방법을 말한다.

일반적으로 상향식 설계는 반복문으로 구현할 수 있으며, 재귀함수로는 전체탐색법에서와 같은 방법의 상향식으로 구현할 수 있다. 재귀호출을 상향식으로 작성할 때에는  $f(0)$  또는  $f(1)$ 의 형태로 호출한다.

**문) 1부터  $N$ 까지의 합을 상향식 설계 동적표를 이용하여 구현하라**

# 동적표 - 상향식 설계(동적계획법)

답안

```
#include <iostream>
using namespace std;

int D[101], n;
int main()
{
    cin >> n;

    for (int i = 1; i <= n; i++)
        if (i == 1)
            D[i] = 1;
        else
            D[i] = D[i - 1] + i;

    cout << D[n] << endl;

    return 0;
}
```

# 동적표 실습

## combination(L)

$nCk$ 는  $n$ 개 중에서  $k$ 개를 고르는 방법의 수이다.

$nCk$ 를 구하기 위한 일반식은 다음과 같다.

$$\binom{n}{k} = \frac{n(n-1)\dots(n-k+1)}{k(k-1)\dots 1} = \frac{n!}{k!(n-k)!}$$

위 식은  $n!$ 을 이용하기 때문에  $n$ 이 커지면 overflow가 발생하여 정확한 값을 구할 수 없다.

물론 위 방법 이외에도 다양한 점화식으로도 구할 수 있다.

$nCk$ 를 정확하게 구하는 프로그램을 작성하시오.

### 입력

첫 번째 줄에  $n$ 과  $k$ 가 공백으로 구분되어 입력된다.  
( 단,  $1 \leq n, k < 30$  )

### 출력

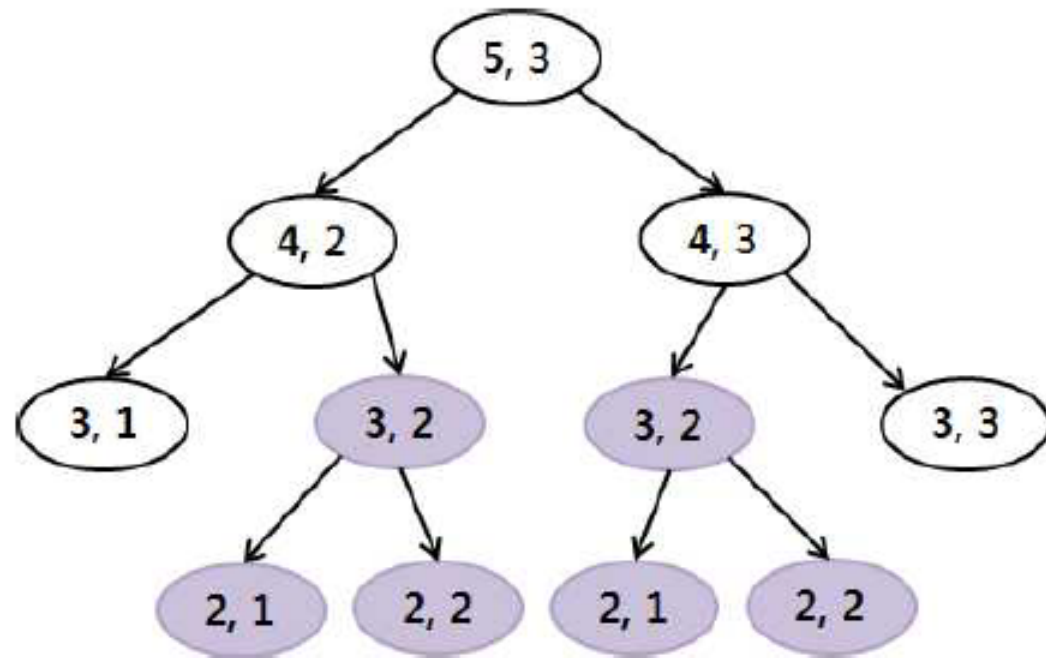
구한 답을 첫 번째 줄에 출력한다.

입력 예	출력 예
5 1	5
10 5	252

# 동적표 실습

## 아이디어

<하향식 설계>



위의 구조에서 보면 진하게 표시된 정점들은 모두 중복호출이 발생하는 정점들이다.  $n$ 과  $k$ 의 값이 커질수록 중복호출의 수는 기하급수적으로 늘어난다. 따라서 동적표를 활용하면 효율이 매우 좋아질 수 있다.

따라서 다음 관계식을 이용하여 소스코드를 작성해보자.

$$f(n, k) = \begin{cases} 1 & (k = n) \\ n & (k = 1) \\ f(n-1, k-1) + f(n-1, k) & (1 < k < n) \end{cases}$$

위 관계식을 다음과 같이 동적표에 적용시키자. 이번의 경우는 2차원이라 다음 2가지 방법으로 적용할 수 있다.

$$\begin{aligned} DT[n][k] &= f(n, k) \text{의 값} \\ DT[nK + k] &= f(n, k) \text{의 값 (단, } K \text{는 최대 열의 크기)} \end{aligned}$$

# 동적표 실습

답안

<하향식 설계>

```
#include <iostream>
using namespace std;

int DT[31][31];
int f(int n, int k)
{
    if (k == n) DT[n][k] = 1;
    else if (k == 1) DT[n][k] = n;
    else
    {
        if (!DT[n][k])
            DT[n][k] = f(n - 1, k - 1) + f(n - 1, k);
    }
    return DT[n][k];
}

int main()
{
    int n, k;
    cin >> n >> k;
    cout << f(n, k) << endl;

    return 0;
}
```

# 동적표 실습

## 아이디어

### <상향식 설계>

먼저 동적표를 다음과 같이 정의하자.

$$DT[n][k] = \text{"n 개 중에 k 개를 고르는 경우의 수"}$$

따라서 다음과 같은 관계를 얻을 수 있다.

$$\begin{aligned} DT[n][k] &= \text{마지막 물체를 고른 경우의 수} + \text{마지막 물체를 고르지 않은 경우의 수} \\ \text{마지막 물체를 고른 경우의 수} &= DT[n-1][k-1] \\ \text{마지막 물체를 고르지 않은 경우의 수} &= DT[n-1][k] \end{aligned}$$

그림을 관계식으로 나타낸 것이다. 이를 종합하면 다음과 같은 관계식을 얻는다.

$$DT[n][k] = DT[n-1][k-1] + DT[n-1][k]$$

그리고 직접 계산하는 상태는 'n개 중 n개를 고르는 경우의 수'와 'n개 중 1개를 고르는 경우의 수'로 이 값은 각각 1가지와 n가지이므로 다음과 같이 정의할 수 있다.

$$DT[1][1] = 1, \quad DT[n][n] = 1, \quad DT[n][1] = n$$

위의 관계를 정리하여 관계식을 구하면 다음과 같다.

$$DT[n][k] = \begin{cases} 1 & (k = n) \\ n & (k = 1) \\ DT[n-1][k-1] + DT[n-1][k] & (1 < k < n) \end{cases}$$

# 동적표 실습

답안

<상향식 설계>

```
#include <iostream>
using namespace std;

int DT[31][31];
int main()
{
    int n, k;
    cin >> n >> k;

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= k && j <= i; j++)
        {
            if (i == j) DT[i][j] = 1;
            else if (j == 1) DT[i][j] = i;
            else DT[i][j] = DT[i - 1][j - 1] + DT[i - 1][j];
        }
    }
    cout << DT[n][k] << endl;

    return 0;
}
```

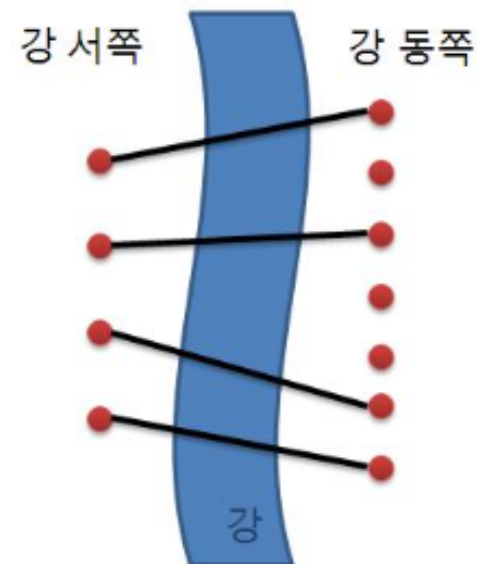
# 동적표 문제 1

<https://www.acmicpc.net/problem/1010>

## 문제

재원이 한 도시의 시장이 되었다. 이 도시에는 도시를 동쪽과 서쪽으로 나누는 큰 강이 흐르고 있다. 하지만 재원은 다리가 없어서 시민들이 강을 건너는데 큰 불편을 겪고 있음을 알고 다리를 짓기로 결심하였다. 강 주변에서 다리를 짓기에 적합한 곳을 사이트라고 한다. 재원은 강 주변을 면밀히 조사해 본 결과 강의 서쪽에는  $N$ 개의 사이트가 있고 동쪽에는  $M$ 개의 사이트가 있다는 것을 알았다. ( $N \leq M$ )

재원은 서쪽의 사이트와 동쪽의 사이트를 다리로 연결하려고 한다. (이때 한 사이트에는 최대 한 개의 다리만 연결될 수 있다.) 재원은 다리를 최대한 많이 지으려고 하기 때문에 서쪽의 사이트 개수만큼 ( $N$ 개) 다리를 지으려고 한다. 다리끼리는 서로 겹쳐질 수 없다고 할 때 다리를 지을 수 있는 경우의 수를 구하는 프로그램을 작성하라.



## 입력

입력의 첫 줄에는 테스트 케이스의 개수  $T$ 가 주어진다. 그 다음 줄부터 각각의 테스트케이스에 대해 강의 서쪽과 동쪽에 있는 사이트의 개수 정수  $N, M$  ( $0 < N \leq M < 30$ )이 주어진다.

## 출력

각 테스트 케이스에 대해 주어진 조건하에 다리를 지을 수 있는 경우의 수를 출력한다.

### 예제 입력 1 복사

```
3
2 2
1 5
13 29
```

### 예제 출력 1 복사

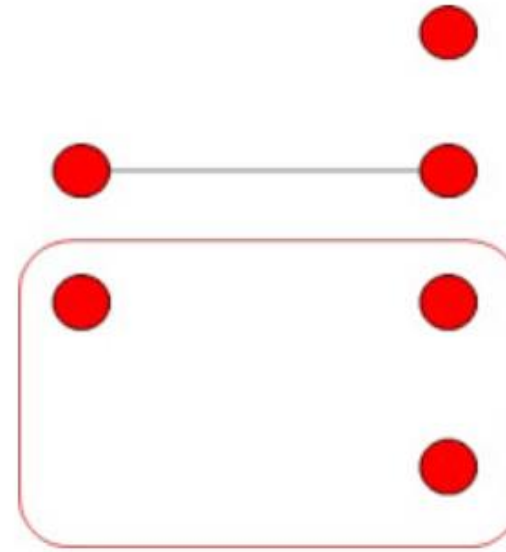
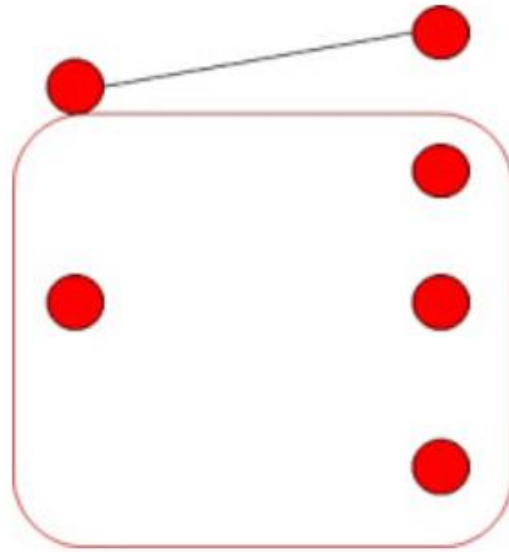
```
1
5
67863915
```



# 동적표 문제 1

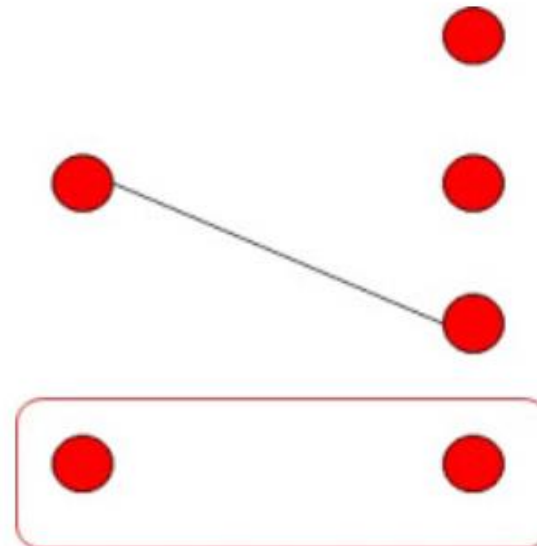
## 아이디어

그림을 자세히 보시면 첫번째 다리를 연결 한 후 남은 다리의 그림은  $N=1, M=3$  일 때 경우와 동일 하며 2번의 경우  $N=1, M=2$  때, 3번은  $N=1, M=1$  일 경우와 똑같습니다.



$N=2, M=4$  는?

$$\text{dp}[2][4] = \text{dp}[1][3] + \text{dp}[1][2] + \text{dp}[1][1]$$



# 동적표 문제 1

## 아이디어

<점화식으로 표현>

NA (N > M)

$dp[N][M] = M$  (N = 1)

$dp[N][M] = dp[N-1][M-1] + dp[N-1][M-2] + \dots + dp[N-1][N-1]$  (Others)

# 동적표 문제 1

답안

<https://onlinegdb.com/H1Pkt9zw>

# 머리 식히는 타임 : 코딩 컨벤션

- 구글 C++ 코딩 스타일 가이드
- <https://valueelectronic.tistory.com/161>
- 입사 하시면 일반적으로 각 회사마다 코딩 컨벤션 가이드가 따로 있습니다.

# 동적표 문제 2

## 숙직 선생님

학교에 몰래 누군가 침투를 했다. 그래서 학교 숙직 선생님이 누군가를 잡기 위해 찾으러 다닌다. 이때 숙직 선생님의 현재 위치와 침투한 누군가의 위치가 주어질 때, 숙직 선생님은 모두 다른 특별한 능력을 3가지 받게 된다.

이 능력은 1초에 이동할 수 있는 거리를 이야기 한다. 1초에 하나의 능력만 사용이 가능하다. 능력을 이용하여 침투한 누군가를 가장 빨리 찾을 수 있는 최소 시간을 출력하라.

경우에 따라서는 찾지 못할 수도 있다. 찾지 못할 경우 -1을 출력한다. 단, 겁이 많은 누군가는 움직이지 않고 숨어만 있다. 이동은 위치가 커지는 방향으로 이동한다.

### 입력

첫 번째 줄에 숙직 선생님의 현재위치 a 누군가의 위치 b(  $1 \leq a \leq b \leq 1000$  인 정수)

두 번째 줄에 3개의 숙직 선생님이 사용할 수 있는 능력이 입력된다(  $1 \leq \text{능력} \leq 100$  인 정수).

### 출력

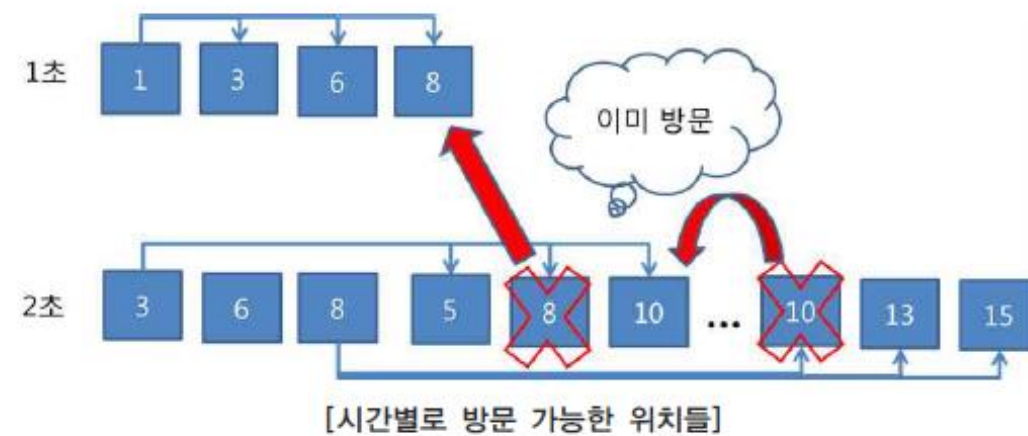
누군가를 찾는 데 걸리는 시간(초)를 출력한다.

단, 찾지 못할 경우는 -1을 출력한다.

입력 예	출력 예
1 15 2 5 7	2

# 동적표 문제 2

## 아이디어



특정 위치까지 소요시간은 “(특정 위치 - 세 가지 능력)의 소요시간 중 최소 + 1”

$D(n)$  :  $n$ 까지 오는 최소 소요시간

$D(a) = 0$  ( $a$ 는 시작위치)

$D(n) = \text{Min}(D(n - \text{able}[i])) + 1$  (단,  $D(n - \text{able}[i]) > a$ )

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	$\infty$	1	$\infty$	2	1	3																		

[8번 위치에서 동적표 갱신 방법]

1번 위치에서 3번째 능력(+7)으로 오는 경우가 최소 + 1 = 1

# 동적표 문제 2

답안

<https://onlinegdb.com/Byjb2kSh8>

# 동적표 문제 3

## 문제 설명

증가하는 수란 숫자의 가장 왼쪽부터 오른쪽으로 읽을 때, 작거나 같은 경우가 없이 항상 증가하는 수를 의미합니다.

즉, 0, 05, 47, 123, 1789 등은 증가하는 수이지만, 455, 790 등은 증가하는 수가 아닙니다.

단 숫자는 0에서 시작할 수 있고, 증가하는 기준은 길이 기준과 숫자 기준이 있습니다. 예를 들어 길이가 4인 6789보다 길이가 5인 01234가 더 큰 수입니다.

또한 길이가 같을 때는, 05보다 12가 더 큰 수입니다.

숫자 K가 주어졌을 때, K번째 증가하는 수를 반환하는 함수를 완성해 주세요.

이 때, K번째로 증가하는 수가 존재하지 **않는**다면 -1을 반환하세요. (반환형은 문자열입니다.)

## 제한사항

K : 1,000,000,000 이하인 자연수

## 입출력 예

입력(K)	출력
1	0
10	9
11	01
999999999	-1

## 입출력 예 설명

입출력 예 #1

규칙에 따른 1번째 숫자는 0입니다.

입출력 예 #2

규칙에 따른 10번째 숫자는 9입니다.

입출력 예 #3

규칙에 따른 11번째 숫자는 01입니다.

왜냐하면 길이가 1인 마지막 숫자는 9, 즉 10번째 숫자이므로 11번째 숫자는 길이가 2인 첫번째 숫자인 01이 됩니다.

입출력 예 #4

규칙에 따른 999999999번째 숫자는 존재하지 않습니다.









# 동적표 문제 3

## 아이디어

0	1	2	3	4	5	6	7	8	9
01	02	03	04	05	06	07	08	09	
	12	13	14	15	16	17	18	19	
		23	.....						29
			.....						
									89

1. d[n][k] k로 시작하는 n자리 증가하는 수  
: k+1부터 시작하는 n-1 자리수의 증가하는 수 총합

 d	0x00a142e8 {0x00a142e8 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}
▶  [0]	0x00a142e8 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
▶  [1]	0x00a14338 {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
▶  [2]	0x00a14388 {9, 8, 7, 6, 5, 4, 3, 2, 1, 0}
▶  [3]	0x00a143d8 {36, 28, 21, 15, 10, 6, 3, 1, 0, 0}
▶  [4]	0x00a14428 {84, 56, 35, 20, 10, 4, 1, 0, 0, 0}

2. input(K번째 수)의 값에 따라 자리수(digit), 시작값(start), offset을 구한다.
3. 증가하는 수를 찾기 위해 문자열 변환 후, 자리수에 맞는 시작값을 세팅한다.
4. cnt까지 1씩 증가시키면서 증가하는 수를 찾는다.
5. 이때, 0부터 시작하는 경우를 처리해준다.

# 동적표 문제 3

답안

<https://onlinegdb.com/By9LikS3I>

1022 => 123456789

1023 => 0123456789