

이름 : 유형곤  
 학번 : 201902722  
 사용 언어 : Java  
 2주차 과제

# <점수표>

RANK	TEAM	SCORE	01-FIBONACCI [1 POINT]	02-GCD [2 POINTS]	03-STAIR [3 POINTS]	04-CHANGE [4 POINTS]
1	201902722	10 139	4 1 try	9 1 try	51 1 try	35 3 tries

문제 1 : w02-fibonacci

문제 : n번째 피보나치 수 구하기

해결 방법 :

피보나치 수 같은 경우는

$fib(n) = fib(n-1) + fib(n-2)$ 라는 점화식이 있습니다.

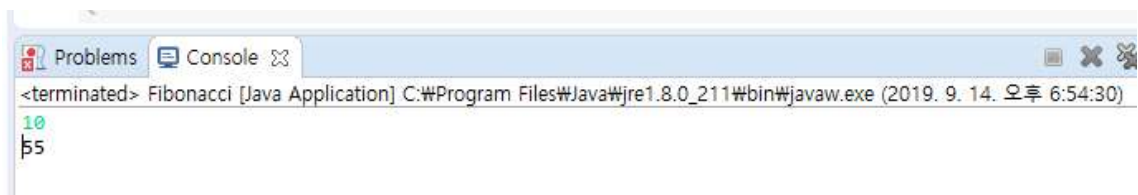
그래서 기본단계인  $n == 1$ ,  $n == 0$ 인 경우만 미리 정의해준 후

점화식에 따라서  $fib(n)$ 의 값을 구했습니다. 그리고 이미 구한 값을 다시 구하지 않도록 long fibo[]라는 배열에 저장했습니다. (memoization) 그 후 원하는 값인 fibo[n]을 출력했습니다. 그런데 그 뒤 문제에선 마음이 급해서 memoization을 하진 않았습니다.

```
public static long fibo(int n) {
    if(n == 0 | n == 1) {
        return n;
    }

    if(fibo[n] != 0) {
        return fibo[n];
    }

    return fibo(n-1) + fibo(n-2);
}
```



문제 2 : w02-gcd

문제 : 최대공약수를 구하기

해결 방법 :

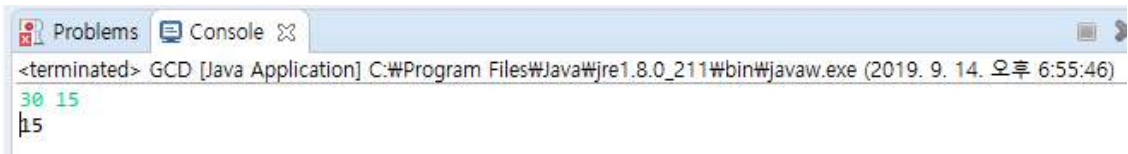
유클리드 호제법에 따라서 숫자 a, b가 있다고 할 때,

a가 몫이 되고 ( $a = b$ , 그런데 a의 값이 다시 필요하므로 tmp에 저장합니다.)

b는 a (tmp)를 b로 나눈 나머지( $a \text{ (tmp)} \% b$ )가 됩니다.

이것을 b가 0이 될 때까지 반복(여기선 재귀로)하면 최대공약수 a를 얻을 수 있습니다.

```
public static int gcd(int a, int b) {  
    if(b == 0) {  
        return a;  
    } else {  
        int tmp = a;  
        a = b;  
        b = tmp % b;  
        return gcd(a, b);  
    }  
}
```



The screenshot shows a Java IDE window with a 'Console' tab. The console output displays the result of the gcd function: '30 15' on the first line and '15' on the second line. The title bar of the window indicates the application is 'GCD [Java Application]' and the path is 'C:\Program Files\Java\jre1.8.0\_211\bin\javaw.exe'.

```
<terminated> GCD [Java Application] C:\Program Files\Java\jre1.8.0_211\bin\javaw.exe (2019. 9. 14. 오후 6:55:46)  
30 15  
15
```

문제 3 : w02-stair

문제 : 계단 수가  $n$ , 최대 도약 수가  $m$ 일 때 계단  $n$ 개를 오르는 경우의 수 구하기

해결 방법 :

우선 계단  $n$ 개를 오르는 경우의 수는

$(k-1)$ 개를 오르는 수

+  $(k-2)$ 개를 오르는 수

+ ...  $(k-c)$ 개를 오르는 수

+  $(k-m)$ 개를 오르는 수

이걸 모두 더하면 됩니다.

그런데  $n$ 이  $m$ 보다 작은 경우는  $k-m$ 이 음수가 되는데, 이 경우에는

$(k-c)$ 가 0이 될 때 까지만 더해주면 됩니다. 이 내용을 top-down 방식 (재귀)으로 구현했습니다.

```
static int stairs[];
static int n, m;
static int sum = 0;

public static void stairs(int k, int cnt) {

    if(k == 0) {
        sum++;
    }

    if(k < m) {
        for(int i = 1; i <= k; i++) {
            stairs(k - i, cnt+1);
        }
    } else {
        for(int i = 1; i <= m; i++) {
            stairs(k - i, cnt+1);
        }
    }
}
```



문제 4 : w02-change

문제 : 동전 n개로 돈을 거슬러줄 때 가장 작은 동전의 개수

해결 방법 :

돈을 거슬러주는 가장 작은 경우를 고르는 것이므로, 모든 경우의 수 중에서 가장 작은 것을 반환하면 됩니다. 그래서 저는 가진 돈이 0이 될 때까지 카운트를 더해서, 이 값을 기존의 최소값과 비교해서 작은 경우 업데이트했습니다. (물론 현재 가진 돈보다 동전이 더 가치가 큰 경우는 스킵 합니다.)

```
public class POS {  
  
    static int min = Integer.MAX_VALUE;  
  
    public static void count(int coins[], int charge, int cnt) {  
  
        if(charge == 0) {  
            if(cnt < min) {  
                min = cnt;  
            }  
        }  
  
        for(int i = 0; i < coins.length; i++) {  
            int d = coins[i];  
            if(d > charge) {  
                continue;  
            }  
            count(coins, charge - d, cnt+1);  
        }  
    }  
}
```



Problems Console

<terminated> POS [Java Application] C:\Program Files\Java\jre1.8.0\_211\bin\javaw.exe (2019. 9. 14. 오후 6:57:38)

5  
21 10 5 1 25  
14  
|

느낀 점 :

제한시간 없이 공부하다보니, 시간이 촉박해서 평소에 하던 걸 안 하거나(memoization을 안 하거나) 안 하던 짓을 하거나(전역변수 남발해서 코드가 지저분해지는 것) 하는 게 있었는데, 제한시간이 있어도 긴장하지 않고 코딩할 수 있게 많이 연습해야겠다고 생각했습니다.