이름 : 유형곤
학번 : 201902722
사용 언어 : Java
3주차 과제

<점수표>

9	201902722	10	1640	1302 13 tries	0 1 try	0 1 try	38 4 tries

문제 1 : w03-fibonacci
문제 : n번째 피보나치 수 구하기
해결 방법 :
저번 주와 마찬가지로 fibo[n] = fibo[n-1] + fibo[n-2]라는 점화식을 사용했는데,
테스트 케이스에 입력되는 n의 범위가 증가했으므로, BigInteger를 사용하였습니다.
재귀함수이므로 종료조건과 점화식을 잘 활용하여 작성했습니다. 또한 메모이제이션(memoization)을 통해서 함수의 실행시간을 개선했습니다.

```
public static BigInteger fibo(int n) {
    if(fibo[n] != null) {
        return fibo[n];
    }
    fibo[n] = fibo(n-1).add(fibo(n-2));
    return fibo[n];
}
```

```
<terminated> Fibonacci (1) [Java Application] C:₩Program Files₩Java₩jre1.8.0_211₩bin₩javaw.exe (2019. 9. 19. 오후 8:46:35)
10
55
```

문제 2 : w03-factorial

문제 : n! 구하기

해결 방법 :

factorial[n] = n * factorial[n-1]식을 사용했고, 팩토리얼은 결괏값이 크므로 BigInteger를 사용하였습니다. 재귀함수이므로 종료조건과 점화식을 잘 활용하여 작성했습니다. 또한 메모이제이션(memoization)을 통해서 함수의 실행시간을 개선했습니다.

```
public static BigInteger factorial(int n) {
    if(factorial[n] != null) {
        return factorial[n];
    }
    factorial[n] = factorial(n-1).multiply(new BigInteger(Integer.toString(n)));
    return factorial[n];
}
```

```
<terminated> Factorial (1) [Java Application] C:₩Program Files₩Java₩jre1.8.0_211₩bin₩javaw.exe (2019. 9. 19. 오후 8:44:56)
10
3628800
```

문제 3 : w03-stair
문제 : 계단 수가 n, 최대 도약 수가 m일 때 계단 n개를 오르는 경우의 수 구하기
해결 방법 :

우선 계단 n개를 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 n이 m보다 작은 경우는 k-m이 음수가 되는데, 이 경우에는
(k-c)가 0이 될 때 까지만 더해주면 됩니다. 이 내용을 top-down 방식 (재귀)으로 구현했습니다.

저번주와 변경된 점은 메모이제이션을 적용했다는 것과, 변수를 BigInteger 타입으로 변경했다는 점입니다.

```
public static BigInteger stairs(int n, int max) {
    if(!stairs[n].equals(BigInteger.ZERO)) {
        return stairs[n];
    }
    for(int i = 1; i <= n; i++) {
        if(i < max) {
            for(int j = 1; j <= i; j++) {
                stairs[i] = stairs[i].add(stairs(i-j, max));
            }
        } else {
            for(int j = 1; j <= max; j++) {
                stairs[i] = stairs[i].add(stairs(i-j, max));
            }
        }
    }
    return stairs[n];
}
```

```
<terminated> Superman (1) [Java Application] C:₩Program Files₩Java₩jre1.8.0_211₩bin₩javaw.exe (2019. 9. 19. 오후 8:52:00)
50 10
551742150354112
```

문제 4 : w03-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 최솟값과 비교해서 작은 경우 업데이트했습니다. (물론 현재 가진 돈보다 동전이 더 가치가 큰 경우는 스킵 합니다.)

저번 주와 바뀐 점은, 저번 주 같은 경우에는 메모이제이션을 하지 않아서 중복된 연산을 반복적으로 수행했습니다. 그래서 cache 배열을 만들어 cache[돈] = (돈을 만드는 데 최소 동전 수)를 저장했습니다.

```
private static int count(int money, int cnt) {
    if(money == 0) {
        return cnt;
    }
    if(cache[money] > 0) {
        return cache[money];
    }
    int min = INF;
    for(int i = 0; i < coins.length; i++) {
        int coin = coins[i];
        if(coin > money) {
            continue;
        }
        int result = count(money - coin, 0) + 1;
        if(result < min) {
            min = result;
        }
    }
    cache[money] = min;
    return min;
}
```

```
<terminated> Change2 [Java Application] C:₩Program Files₩Java₩jre1.8.0_211₩bin₩javaw.exe (2019. 9. 19. 오후 9:00:33)
5
21 10 5 1 25
2019
82
```

느낀 점 :
저번 주에 작성했던 코드를 바탕으로 코드를 작성했는데, 처음 작성한 코드의 구조가 지저분하다보니 코드를 고치기가 어려웠습니다. 그래서 코드를 짜려면 처음부터 잘 짜야겠다고 생각했습니다.