9184 - 신나는 함수 실행

난이도	실버2

문제

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	19162	8287	6313	42.078%

문제

재귀 호출만 생각하면 신이 난다! 아닌가요?

다음과 같은 재귀함수 w(a, b, c)가 있다.

```
if a <= 0 or b <= 0 or c <= 0, then w(a, b, c) returns:
    1

if a > 20 or b > 20 or c > 20, then w(a, b, c) returns:
    w(20, 20, 20)

if a < b and b < c, then w(a, b, c) returns:
    w(a, b, c-1) + w(a, b-1, c-1) - w(a, b-1, c)

otherwise it returns:
    w(a-1, b, c) + w(a-1, b-1, c) + w(a-1, b, c-1) - w(a-1, b-1, c-1)
```

위의 함수를 구현하는 것은 매우 쉽다. 하지만, 그대로 구현하면 값을 구하는데 매우 오랜 시간이 걸린다. (예를 들면, a=15, b=15, c=15)

a, b, c가 주어졌을 때, w(a, b, c)를 출력하는 프로그램을 작성하시오.

입력

입력은 세 정수 a, b, c로 이루어져 있으며, 한 줄에 하나씩 주어진다. 입력의 마지막은 -1 -1 -1로 나타내며, 세 정수가 모두 -1인 경우는 입력의 마지막을 제외하면 없다.

출력

입력으로 주어진 각각의 a, b, c에 대해서, w(a, b, c)를 출력한다.

제한

- -50 ≤ a, b, c ≤ 50

예제 입력 1 복사

```
1 1 1
2 2 2
10 4 6
50 50 50
-1 7 18
-1 -1 -1
```

예제 출력 1 복사

```
w(1, 1, 1) = 2
w(2, 2, 2) = 4
w(10, 4, 6) = 523
w(50, 50, 50) = 1048576
w(-1, 7, 18) = 1
```

풀이과정

DP의 메모이제이션 특성을 이용해 풀이했다.

💡 주요 포인트

1. a, b, c 중 하나라도 0 이하일 경우 w(a,b,c) = 1

문제에서 주어진 로직대로 캐시 배열에 저장할 때, 세 개의 인덱스 중 하나라도 0이 있다면 1로 초기화해주자.

2. a, b, c 중 하나라도 20보다 클 경우 w(a,b,c) = w(20,20,20)

따라서 해당 문제에 대한 캐시 배열은 3차원으로 만들되 각 크기를 21로 만들어주면 된다.

> **왜 21이지?** 인덱스가 0인 부분은 제외하고 1부터 시작하기 위해서 크기를 21로 만들어주었다.

소스코드

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class Main {
    static int a,b,c;
    static int[][][] cache;

    public static void main(String[] args) throws Exception{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st;
        while(true) {
            String str = br.readLine();
            if(str.equals("-1 -1 -1")) break;
            st = new StringTokenizer(str);
            a = Integer.parseInt(st.nextToken());
            b = Integer.parseInt(st.nextToken());
            c = Integer.parseInt(st.nextToken());

            int tmp = dp(a, b, c);
            System.out.println("w(" + a + ", " + b + ", " + c + ") = " + tmp);
        }

        br.close();
    }

    static int dp(int a, int b, int c){
        if(a <= 0 || b <= 0 || c <= 0) return 1;
        if(a > 20 || b > 20 || c > 20) return dp(20, 20, 20);

        int answer = 0;

        // a,b,c 중 하나라도 0 이하면 1 리턴 -> 여기서 계산할 a,b,c는 양수
        // a,b,c 중 하나라도 20보다 크다면 a=b=c=20 -> a,b,c는 최대 20까지
        cache = new int[21][21][21];

        // 인덱스 중 하나라도 0일 경우 값을 1로 다 초기화
        for(int i=0 ; i<21 ; i++){
            for(int j=0 ; j<21 ; j++) {
                cache[0][i][j] = 1;
                cache[i][0][j] = 1;
                cache[i][j][0] = 1;
            }
        }

        for(int i=1 ; i<=a ; i++){
            for(int j=1 ; j<=b ; j++){
                for(int k=1 ; k<=c  ;k++){

                    // 인덱스 검사 -> 계산 결과 중 하나라도 0 이하일 경우
                    int tmpI, tmpJ, tmpK;
                    if(i-1<=0) tmpI = 0; else tmpI = i-1;
                    if(j-1<=0) tmpJ = 0; else tmpJ = j-1;
                    if(k-1<=0) tmpK = 0; else tmpK = k-1;

                    if(i<j && j<k)
                        cache[i][j][k] = cache[i][j][tmpK] + cache[i][tmpJ][tmpK] - cache[i][tmpJ][k];
                    else
                        cache[i][j][k] = cache[tmpI][j][k] + cache[tmpI][tmpJ][k] + cache[tmpI][j][tmpK] - cache[tmpI][tmpJ][tmpK];

                    if(i == a && j == b && k == c) {
                        return cache[i][j][k];
                    }
                }
            }
        }
        return answer;
    }
}
```

사담

처음에는 캐시 배열을 3차원으로 하기엔 메모리 초과나 시간초과가 날 것만 같아서 다른 방법을 찾아보려 했으나 쓸데없이 복잡해졌었다.

설마하는 마음으로 캐시 배열을 3차원 배열으로 만들어주고 실행해보니 이게 됐다. 아무래도 넉넉한 편이었나보다.