M apples N plates problem

We want M apples to be placed in N plates. It allows empty plates. How many choices to distribute apples to plates? And it does not allow repeatable combinations.

For example, M=4, N=3. The combinations are (0, 0, 4), (0, 1, 3), (0, 2, 2), (1, 1, 2). So 4 choices are acceptable. (1, 1, 2), (1, 2, 1) and (2, 1, 1) are repeatable combinations, they are regarded as one choice.

Define f(m, n) as the number of choices.

1) if n>m, there must exist n-m empty plates. f(m, n) = f(m, m).

2) if n<=m,

a) if there exists at least one empty plates, f(m, n) = f(m, n-1)

b) if all plates are placed with apples, suppose we remove one apple from N plates, m-n apples now are placed into N plates. So f(m, n) = f(m-n, n).

So in summary, f(m, n) = f(m, n-1) + f(m-n, n).

The initial condition: n=1, f(m, 1) = 1; m=0 or 1, f(m, n) =1

```
int func(int m, int n)
{
    if (n == 1 || m == 0 || m == 1)
        return 1;
    if (m < n)
        return func(m, m);
    else
        return func(m - n, n) + func(m, n - 1);
}

int main()
{
    int m, n;
    while (~scanf("%d%d", &m, &n))
    {
        printf("%d\n", func(m, n));
    }
    return 0;
}
```

To go further, if it allows repeatable combinations, (1, 1, 2), (1, 2, 1) and (2, 1, 1) are repeatable combinations, they are regarded as three choices.

Same, we can use iterations to solve it.

```
int fun(int m, int n){
    if(m <= 0)
        return 1;
    if(n <= 0)
        return 0;
    int num=0;
    for(int i=0;i<=m; ++i){
        num += fun(m-i, n-1);
    }
    return num;
}
```