Sum of Digits

Majid is a 3rd-grade elementary student and quite well in mathematics.
Once, Majid's teacher asked him to calculate the sum of numbers 1 through n.

Majid quickly answered, and his teacher made him another challenge.
He asked Majid to calculate the sum of the digits of numbers 1 through n.

Majid did finally find the solution. Now it is your turn, can you find a solution?

Input

Two space-separated integers 0 ≤ a ≤ b ≤ 10^9.

Program terminates if a and b are -1.

Output

The sum of the digits of numbers a through b.

Example

```
Input:
1 10
100 777
-1 -1

Output:
46
8655
```

Problem Breakdown

Lets understand the problem first

Given a and b, compute:

    sum_{x = a..b} digitSum(x)

We do:

    ans = f(b) − f(a-1)

Where f(n) = sum of digits of all numbers from 0 to n.

DIGIT DP to compute sum(0..n)

This DP will return two things:

- how many numbers we can build
- the total digit sum across those numbers

So our DP returns a pair<long long count, long long sum>

DP State

We define:

```
pair<long long, long long> dfs(pos, tight)
```

Meaning:

- count = how many numbers we can build from pos to end
- sum = total sum of digits of *ALL* those numbers

We compute:

For each digit d allowed:

```
child = dfs(nextPos, nextTight)
count += child.count
sum += child.sum + d * child.count
```

The term:

```
d * child.count
```

represents:

- every number formed from child contributes digit d at THIS position
- so their total contribution = d multiplied by how many numbers child generates

This is the magical trick for sum-of-digits DP.

Code

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

vector<int> digits;

// dp[pos][tight] -> pair(count_of_numbers, sum_of_digits_of_those_numbers)
pair<ll,ll> dp[20][2];
bool vis[20][2];

pair<ll,ll> dfs(int pos, int tight) {
    if (pos == (int)digits.size()) {
        return {1, 0};   // 1 number: the empty suffix; sum=0
    }
    if (vis[pos][tight]) return dp[pos][tight];

    ll cnt = 0, sum = 0;
    int limit = tight ? digits[pos] : 9;

    for (int d = 0; d <= limit; d++) {
        int nt = (tight && d == limit);
        auto child = dfs(pos + 1, nt);

        cnt += child.first;
        sum += child.second + d * child.first;
    }

    vis[pos][tight] = true;
    return dp[pos][tight] = {cnt, sum};
}
ll solve(ll n) {
    if (n < 0) return 0;

    digits.clear();
    if (n == 0) digits.push_back(0);
    while (n > 0) {
        digits.push_back(n % 10);
        n /= 10;
    }
    reverse(digits.begin(), digits.end());

    memset(vis, 0, sizeof(vis));
    return dfs(0, 1).second;  // we want the sum part
}

int main() {
    while (true) {
        ll a, b;
        cin >> a >> b;
        if (a == -1 && b == -1) return 0;

        cout << solve(b) - solve(a - 1) << "\n";
    }
    return 0;
}
```