Digit DP

Q. You are given three integers L, R and m. Find the number of integers in the range [L, R] such that the sum of their digits is divisible by m.

Constraints:
1 <= L, R <= 10^18
1 <= m <= 100

```
#include<bits/stdc++.h>
using namespace std;

#define int long long

int L, R, m;

int dp[20][2][200];

string bound;

void updateBound(int n){
    bound = to_string(n);
    while(bound.size()<20) bound = "0"+bound;
}

int solve(int pos, int eq, int sm){
    // Base Case
    if(pos>=20) return (sm==0);

    int& val = dp[pos][eq][sm];
    // Memoization condition
    if(val!=-1) return val;

    val=0;
    if(eq){
        int curr_digit = bound[pos]-'0';
        for(int i=0;i<curr_digit;i++){
            val += solve(pos+1, 0, ((sm+i)%m + m)%m);
        }
        val += solve(pos+1, 1, ((sm+curr_digit)%m + m)%m);
    }else{
        for(int i=0;i<10;i++){
            val += solve(pos+1, 0, ((sm+i)%m + m)%m);
        }
    }
```

```
    return val;
}

int32_t main(){
    cin>>L>>R>>m;

    // Add count of numbers from [0,R] which are divisible by m
    memset(dp, -1, sizeof dp);
    updateBound(R);
    int ans = solve(0, 1, 0);

    // Subtract count of numbers from [0,L-1] which are divisible by m
    memset(dp, -1, sizeof dp);
    updateBound(L-1);
    ans-=solve(0, 1, 0);

    cout<<ans<<"\n";
    return 0;
}
```