```
// LightOJ 1032 - Fast Bit Calculations
// Bit DP (Almost same as Digit DP)
// Complexity O(2*pos*total_bits*tights*number_of_bits)
// Initial Params : (MostSignificantOnBitPos, N, 0, 0, 1)
// Call as : bitDP(SigOnBitPos, N, 0, 0, 1)   N is the Max Value, calculating [0 - N]
// Tight is initially on, it turns off when the mask contains no maximum digits
// pairs are number of paired bits, prevOn shows if previous bit was on (it is for this problem)

int N, lastBit;
ll dp[33][33][2][2];
ll bitDP(int pos, int mask, int pairs, bool prevOn, bool tight) {
   if(pos < 0)
      return pairs;
   if(dp[pos][pairs][prevOn][tight] != -1)
      return dp[pos][pairs][prevOn][tight];
   bool newTight = tight & !isOn(mask, pos);          // Turn off tight when we are turning off a bit which was initially on
   ll ans = bitDP(pos-1, Off(mask, pos), pairs, 0, newTight);

   if(On(mask, pos) <= N)
      ans += bitDP(pos-1, On(mask, pos), pairs + prevOn, 1, tight && isOn(mask, pos));

   return dp[pos][pairs][prevOn][tight] = ans;
}

// LightOJ 1068 - Investigation (Digit DP)
// Complexity : O(10*idx*sum*tight)
// Tight contains if there is any restriction to number (Tight is initially 1)
// Initial Params: (MaxDigitSize-1, 0, 0, 1, modVal, digitvector)

ll dp[15][100][100][2];

ll digitSum(int idx, int sum, ll value, bool tight, int mod, vector<int>&MaxDigit) {
   if (idx == -1)
      return ((value == 0) && (sum == 0));
   if (dp[idx][sum][value][tight] != -1)
      return dp[idx][sum][value][tight];
   ll ret = 0;
   int lim = (tight)? MaxDigit[idx] : 9;                  // Numbers are genereated in reverse order
   for (int i = 0; i <= lim; i++) {
      bool newTight = (MaxDigit[idx] == i)? tight : 0;    // Caclulating newTight value for next state
      ll newValue = value ? ((value*10) % mod)+i : i;
      ret += digitSum(idx-1, (sum+i)%mod, newValue%mod, newTight, mod, MaxDigit);
   }
   return dp[idx][sum][value][tight] = ret;
}

int main() {
   vector<int>mx;                                         // Contains the digits of the number (The Upper Bound)
   while(Val) {                                           // Val is the input value
      mx.push_back(Val%10);                               // The numbers are placed in reversed order
      Val /= 10;
   }
   ll ans = digitSum((int)mx.size()-1, 0, 0, 1, ModVal, mx);
}
```