

Rozważmy chyba najbardziej znany problem ze zliczaniem. Na taras wieży widokowej prowadzą schody o n stopniach. Turysta wchodzi na taras wykonując kroki co 1 lub co 2 stopnie. Na ile sposobów turysta może przejść schody?

```
int fib(int n) {
    int DP[n + 1];
    DP[0] = DP[1] = 1;
    for (int i = 2; i <= n; i++) {
        DP[i] = DP[i-1] + DP[i-2];
    }
    return DP[n];
}
```

Mamy monety o nominałach $1, k, k^2, k^3, \dots$ (k jest stałe). Na ile sposobów możemy rozmiąć nimi kwotę n ?

$$k + n \Rightarrow \nearrow$$

$$k, k^2, k^3, \dots$$

$$k \mid n$$



$$1, k, k^2, \dots$$

$$\frac{n}{k}$$

```
int money (int n, int k) {
    if (k <= 1)
        return 1;
    DP[0] = DP[1] = 1;
    for (int i = 2; i <= n; i++) {
        if (i % k != 0)
            DP[i] = DP[i-1];
        else
            DP[i] = DP[i-1] + DP[i/k];
    }
    return DP[n];
}
```

Tym razem zliczanie użyjemy zliczania jako pomocniczego narzędzia. Pozwolę sobie na użycie bardzo prostego przykładu: chcemy wypisać k -tą liczbę podzielną przez 3, gdzie $k \leq 10^{18}$. Wiem, że możemy wypisać $3 \cdot k$ - chcę tylko pokazać ogólne podejście :)

$$dp[n][c][s] \quad \square _ _ _$$

```
int ustal(int x) {
```



```
int ustal(int x) {
    //upraszcza kod - zwraca reszta z dzielenia przez 3
    x += 33333333; //nie chcemy modulowac liczb ujemnych
    return x % 3;
}
```

$c \cdot 10^k + \dots$

```
DP[0][0] = 1;
for (int i = 0; i <= 9; i++)
    DP[1][i % 3]++;
for (int i = 2; i <= MAX_N; i++)
    for (int j = 0; j < 3; j++)
        for (int pierwsza = 0; pierwsza < 10; pierwsza++)
            DP[i][j] += DP[i-1][ustal(j - pierwsza)];
```

```
vector<int> wynik;
int reszta = 0;
for (int i = MAX_N; i > 0; i--) {
    for (int c = 0; c < 10; c++) {
        long long dokoncz = DP[i-1][ustal(3 - reszta - c)];
        if (dokoncz >= k) {
            wynik.push_back(c);
            reszta += c;
            break;
        }
        else k -= dokoncz;
    }
}
int pocz = 0;
while (wynik[pocz] == 0)
    pocz++;
for (int i = pocz; i < wynik.size(); i++)
    cout << wynik[i];
```

0 - - -

1
- 2

```
///Zliczanie
dp[0][0][0] = 1;
for(int i = 0; i <= D; ++i) {
    for(int j = 0; j <= S; ++j) {
        for(int r = 0; r < m; ++r) {
            for(int d = 0; d <= 9; ++d) {
                //25
                add(dp[i+1][j+d][(r+d*powBase[i]) % m], dp[i][j][r]); // [ilosc cyfr][suma cyfr][modul]
            }
        }
    }
}
```

```

///Zliczanie
dp[0][0][0] = 1;
for(int i = 0; i <= D; ++i) {
    for(int j = 0; j <= S; ++j) {
        for(int r = 0; r < m; ++r) {
            for(int d = 0; d <= 9; ++d) {
                //25
                add(dp[i+1][j+d][(r+d*powBase[i]) % m], dp[i][j][r]); // [ilosc cyfr][suma cyfr][modul]
            }
        }
    }
}

```

```

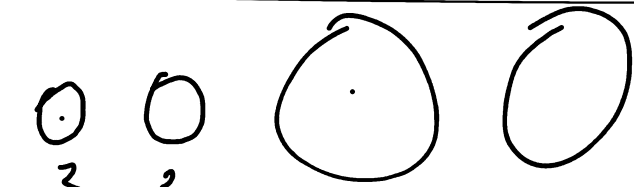
powBase[0] = 1;
for(int i = 1; i <= D; ++i) {
    powBase[i] = powBase[i-1]*10 % m;
}

```

```

for(int i = D; i > 0; --i) {
    for(int d = 0; d <= 9; ++d) {
        //25 (7) - 2 = 5
        int newSum = sum - d;
        //1 - 2 = 2 (mod 3)
        int newMod = ((mod - d*powBase[i-1]) % m + m) % m; // mod - d * powBase[i-1] === x
        if(dp[i-1][newSum][newMod] < k)
            k -= dp[i-1][newSum][newMod];
        else {
            res += '0'+d;
            mod = newMod;
            sum = newSum;
            break;
        }
    }
}

```



 $i = n, \text{wyn} = 1$

 while $i > 0$

 $il += \text{wys}[i]$

 $\text{wyn} = \text{wyn} \cdot il \% m$

 $i--$

 $il--$