Rekurentne relacije

Zeka mora da pređe n metara skačući i to skokovima dužine 3, 2 ili 1 metar. Na koliko načina to može da uradi zeka, ako dužine uspešnih skokova formiraju neopadajući niz? Napišite program koji računa broj načina. Vrednost n se unosi sa standardnog ulaza ($1 \leq n \leq 10^9$). Program treba da na standardni izlaz ispiše jedan ceo broj jednak ostatku pri deljenju sa 1000000.
Test primer

Input	Output
6	7

Objašnjenje: Broj različitih načina je 7, i njegov ostatak pri deljenju sa 1000000 je 7. Nizovi skokova su:
1) 3+3
2) 1+2+3
3) 1+1+1+3
4) 2+2+2
5) 1+1+2+2
6) 1+1+1+1+2
7) 1+1+1+1+1+1

Rešenje 1
Označimo sa x, y, z broj skokova dužine 3, 2, 1, redom.
Neka M=1000000
Mi zapravo treba da izračunamo broj različitih nenegativnih celobrojnih rešenja jednačine $3x + 2y + z = n$**.**

Jasno da $x \leq n/3$, i da za fiksiranu vrednost x, moguće vrednosti za y su 0, 1, …, $(n – 3x)/2$.

```
int cnt=0;
for(int x=0; x<= n/3; x++)
for(int y=0; y<=(n-3*x)/2; y++)
{ cnt++;
if(cnt>=M) cnt = cnt - M;
}
cout << cnt << endl;
```

Mana ovog rešenja je kvadratna složenost.
Na primer, ako je n= 987654321, prevelika je složenost da bi dobili rešenje 324748.

Testirajmo program:

```
#include <iostream>
typedef long long ll;
const ll M = 1000000;
using namespace std;
int main() {
 int n;
 cin >> n;
  long long cnt=0;  //resenje, broj nacina
for(int x=0; x<= n/3; x++)
for(int y=0; y<=(n-3*x)/2; y++)
{ cnt++;
if(cnt>=M) cnt = cnt - M;
}
cout << cnt << endl;
 return 0;
```

}

Rešenje 2
Možemo ukloniti unutrašnji ciklus u rešenju 1. Za fiksiranu vrednost za x postoji tačno $1+ (n -3x) / 2$ mogućnosti za y.
Dodatno da bi izbegli množenja, označimo da $a = 3x$. Novo rešenje:

```
int cnt=0;
for(int a=0; a<=n; a=a+3)
{ cnt = cnt + (n-a)/2 + 1;
if(cnt>M) cnt = cnt%M;
}
cout << cnt << endl;
```

Složenost O(n)

Rešenje 3
Koristeći prethodna dva načina za rešavanje ovog problema, možemo uočiti sledeće karakteristike konačnog rešenja:
Uočimo broj izračunatih načina za $n = 1, 7, 13, 19, \ldots, 6k + 1, \ldots$

n	1	7	13	19	25
cnt	1	8 = 1 + 7	21 = 8 + 13	40 = 21 + 19	65 = 40 + 25

Na primer za n=7, razbijanja broja 7 su
1) 3+3+**1**
2) 3+2+1+**1**
3) 3+1+1+1+**1**
4) **3+2+2**
5) 2+2+2 **+1**
6) 2+2+1+1 **+1**
7) 2+1+1+1+1 **+1**
8) 1+1+1+1+1+1+**1**

Dakle, uočimo da rešenje za $x \geq 7$ zadovoljava rekurentnu relaciju $cnt(x) = cnt(x – 6) + x$.
Na primer, za x=13, 21=cnt(x)=cnt(7)+13=8+13
Možemo pretpostaviti da $cnt(0)=1$

```
int x = n%6;
int cnt=0;
if(x==0) cnt=1; else cnt=x;
while(x<n)
{ x = x + 6;
cnt = cnt + x;
if(cnt >= M) cnt = cnt%M;
}
cout << cnt << endl;
```

Složenost O(n), ali bez množenja u odnosu na rešenje 2

Rešenje 4
U odnosu na rešenje 3, uočimo da rešenje za x=25 je $cnt(25) = 1 + 7 + 13 + 19 + 25$,
Tj. $cnt(6k + 1) = 1 + 7 + 13 + \ldots + (6k + 1)$.
Dakle, uvećanje svakog sabirka u sumi je 6, a broj sabiraka je $k + 1$.

Možemo primeniti Gausov trik da sračunamo svaku sumu. Na primer:
$S = 1 + 7 + 13 + 19 + 25$
$S = 25 + 19 + 13 + 7 + 1$
$2S = 26 + 26 + 26 + 26 + 26 = 26 * 5$
$S = 26 * 5 / 2 = 65.$
Za n=6k+1 imamo da S=(1+n)/2*(1+k)

Implementirajmo ovu ideju:
long long cnt = ((n%6 + n)/2%M * ((1 + n/6)%M));
if(n%6==0) cnt++;
cout << cnt%M << endl;

Uocite da smo napravili malu korekciju resenja za slucaj kad je n deljivo sa 6, tj.
if(n%6==0) cnt++;

Još test primera

ULAZ	IZLAZ
15	27
234	4681
123456789	244272
987654321	324748
111222333	441408
100500700	291184
88888888	267490

```
#include <iostream>
typedef long long ll;
const ll M = 1000000;
using namespace std;
int main() {
 int n;
 cin >> n;
 long long cnt = ((n%6 + n)/2%M * ((1 + n/6)%M));
  if(n%6==0) cnt++;
  cout << cnt%M << endl;
 return 0;
}
```