## SOLUTIE PIRAMIDA

Înălţimea maximă a piramidei este: $[\log_2 S]+1$.

Formula de calcul a valorii din vârful piramidei de înălţime **n** care are la bază numerele nenule $b_0, b_1, \ldots b_{n-1}$ este $C_n^0 \cdot b_0 + C_n^1 \cdot b_1 + \ldots + C_n^{n-1} \cdot b_{n-1} + C_n^n \cdot b_n$. Deoarece $C_n^k = C_n^{n-k}$ obţinem formula de calcul S= $C_n^0 \cdot (b_0 + b_n) + C_n^1 \cdot (b_1 + b_{n-1}) + \ldots$

Calculul posibilităţilor de a obţine S ca sumă de astfel de termeni are la bază un algoritm de programare dinamică tip ”rucsac” (modalităţi de obţinere a capacităţii S cu ajutorul capacităţilor $C_n^0$, $C_n^1, \ldots, C_n^{\left[\frac{n}{2}\right]}$ .

```
#include <stdio.h>

#define NMAX (512*1024)
#define LN 20
#define MOD 10000

int A[NMAX], C[LN][LN], V[LN];

int N;

int main()
{
 int i, j, db, cn;
 register v;

 for (i = 0; i < LN; i++)
     C[i][0] = 1;
 for (i = 1; i < LN; i++)
     for (j = 1; j <= i; j++)
         C[i][j] = (C[i-1][j] + C[i-1][j-1]) % MOD;

 freopen("piramida.in", "r", stdin);
 scanf("%d", &N);
 cn = N;
 db = 0;
 while (cn /= 2) db++;

 N -= (1 << db);

 A[0] = 1;
 for (i = 0; i <= db; i++)
 {
     v = C[db][i] % MOD;
     for (j = v; j <= N; j++)
         A[j] = (A[j] + A[j-v]) % MOD;
 }

 freopen("piramida.out", "w", stdout);
 printf("%d\n", A[N]);

 return 0;
}
```