

## 1. feladat

### A feladat címe: Bit Strings

#### A feladat linkje:

[https://cses.fi/problemset/task/1617?fbclid=IwY2xjawOPyk5leHRuA2FlbQIxMABicmlkETBPaE1zMm9wS3ozRzg4aVMxc3J0YwZhcHBfaWQQMjIyMDM5MTc4ODIwMDg5MgABHucZ5\\_4HydIHENGDbzijY5yBKgPoPr10N98KmnlgggmNUL5TQDeTxOGONoq8\\_aem\\_ViT1p\\_2F3P\\_b3jnpAAwNGA](https://cses.fi/problemset/task/1617?fbclid=IwY2xjawOPyk5leHRuA2FlbQIxMABicmlkETBPaE1zMm9wS3ozRzg4aVMxc3J0YwZhcHBfaWQQMjIyMDM5MTc4ODIwMDg5MgABHucZ5_4HydIHENGDbzijY5yBKgPoPr10N98KmnlgggmNUL5TQDeTxOGONoq8_aem_ViT1p_2F3P_b3jnpAAwNGA)

#### A feladat szövege:

Your task is to calculate the number of bit strings of length n.

For example, if n=3, the correct answer is 8, because the possible bit strings are 000, 001, 010, 011, 100, 101, 110, and 111.

Input

The only input line has an integer n.

Output

Print the result modulo  $10^{9+7}$ .

A feladat az n hosszúságú bitsorozatok számának meghatározása.

Adott egy egész szám, n, és meg kell mondanunk, hogy hány különböző 0–1 sorozat létezik, amelyek pontosan n hosszúak.

Az eredményt a  $10^9 + 7$  modulus alatt kell kiírni.

Például ha n = 3, akkor a helyes válasz 8, mert a lehetséges bitsorozatok:

000, 001, 010, 011, 100, 101, 110 és 111.

Bemenet

Az egyetlen bemeneti sor egy egész számot tartalmaz: n.

Kimenet

Írd ki az eredményt  $10^9 + 7$  mod szerint.

#### A feladat megoldása:

```
MOD = 10**9 + 7
n = int(input())
print(pow(2, n, MOD))
```

#### A feladat magyarázata:

Egy bitsorozat minden pozíciója kétféle lehet: 0 vagy 1. Ez azt jelenti, hogy az első helyre 2 lehetőség van, a második helyre 2 lehetőség van, így tovább, összesen n helyre.

Tehát a kombinációk száma:  $2 \cdot 2 \cdot 2 \cdot \dots = 2^n$

Probléma: n akár milliós nagyságrendű is lehet, így  $2^n$ -en óriási szám lenne. Erre megoldás a modulus használata.

Az 1. lépésben létrehozzuk a MOD változót, a feladat kiírásának megfelelően.

A 2. lépésben beolvassunk az inputról egy n egész számot.

A harmadik lépésben kiíratjuk az eredményt és használjuk a pow függvényt. A pow függvény három paraméterrel pontosan azt csinálja, amire szükségünk van: kiszámolja a  $2^n$  értékét a MOD-osztás alatt, ráadásul nagyon gyorsan, logaritmikus időben /O(logn)/. Így akkor is hatékony marad a program, ha n akár egymillió.