Дорогою до школи

Умова задачі

Щоранку, ідучи до школи, Іванко радісно збігає по сходах. Щоб урізноманітнити свою дорогу та розім'яти не тільки ноги, а й мозок, хлопчик вирішив кожного дня ходити різними способами: піднімаючись сходами, він при кожному кроці може або ступати на чергову сходинку, або переступати через одну. Невдовзі перед школярем постало питання: за скільки днів він зможе перебрати всі способи піднятися сходами.

Допоможіть Іванкові визначити кількість способів k, якими він може піднятися сходами, якщо загальна кількість сходинок по дорозі до школи рівна n.

На вході задано число n, кількість сходинок $(2 \leq n \leq 40)$. На виході вивести одне ціле число k — кількість способів піднятися сходами.

n	k
2	2
5	8

Тести

Номер тесту	Вхідні дані, n	Вихідні дані, k
1	2	2
2	3	3
3	5	8
4	10	89
5	15	987
6	20	10946
7	25	121393
8	30	1346269
9	35	14930352
10	40	165580141

Пояснення розв'язку

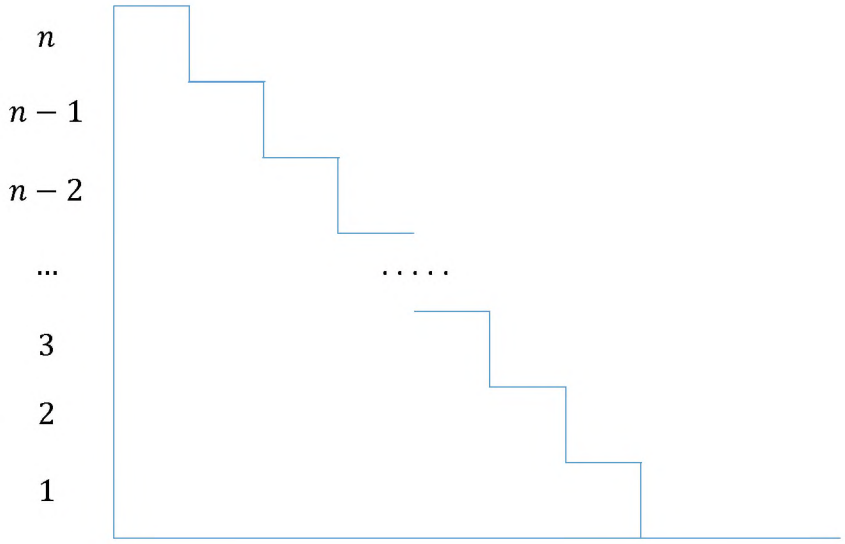
Нехай число k (кількість способів піднятися сходами) визначається функцією $F(n)$, де n — кількість сходинок:

$$k = F(n).$$

Розглянемо сходження на останню, n-ну сходинку. До неї можна дістатися, ступивши або з попередньої, $n-1$ сходинки, або переступивши через одну, тобто зі сходинки $n-2$.

Тоді, враховуючи, що кількість способів сходження для $(n-1)$-ї та $(n-2)$-ї сходинки складає, відповідно, $F(n-1)$ та $F(n-2)$, отримаємо залежність

$$F(n) = F(n-1) + F(n-2).$$



З курсу математики відомо, що дана залежність визначає рекурентне співвідношення чисел Фібоначчі.

За умовою задачі: $n \geq 2$, тобто, до уваги не беруть сходи з кількістю сходинок, рівною 0 та 1, зійти на які можна лише 1 способом, решта способів сходжень визначається таблицею.

n	0	1	2	3	4	5	6	...
$k = F(n)$	1	1	1 + 1 = 2	1 + 2 = 3	2 + 3 = 5	3 + 5 = 8	5 + 8 = 13	...

Програмна реалізація методу Фібоначчі проста, на кожному кроці додають два числа, отриманих на попередніх кроках.

Розв'язок задачі мовою Python

```
a=b=1
for i in range(int(input())): a,b=b,b+a
print(a)
```

Остання цифра

Умова задачі

Іванко ходить до школи разом з сестрою Марічкою, яка допомагає братові рахувати кількість способів сходження. Але дівчинка лише першокласниця і ще не знає великих чисел, навіть двоцифрових, тому може запам'ятати лише останню цифру числа.

Напишіть програму, яка визначає цю останню цифру загальної кількості способів підйому.

На вході задано число n, $(2 \leq n \leq 10^8)$. На виході треба вивести одне число k — останню цифру кількості способів піднятися n сходами.

n	k
7	1
2	2

Тести

Номер тесту	Вхідні дані, n	Вихідні дані, k
1	7	1
2	2	2
3	12	3
4	17	4
5	634	5
6	3047	6
7	36036	7
8	424892	8
9	2403697	9
10	9607244	0

Пояснення розв'язку

Дана задача відрізняється від попередньої тим, що треба ***виводити*** не все число способів сходження на n-ну сходинку, а лише його останню цифру. Іншими словами, ***залишок від ділення на 10***.

Крім того, спроба використати алгоритм прямого знаходження n-го члена послідовності Фібоначчі призведе до порушень часових обмежень задачі з огляду на велике значення числа n (за умовою $n \leq 10^8$).

Виходом з ситуації буде аналіз послідовності Фібоначчі з метою знаходження закономірностей для визначення останньої цифри. Проаналізувавши цю послідовність

самостійно, ви помітите, що останні цифри чисел Фібоначчі утворюють періодичну послідовність з періодом $\Pi = 60$.

Тобто для пошуку останньої цифри числа Фібоначчі ***кінцевим значенням циклу буде*** не вхідне значення змінної n, а ***залишок від ділення n на 60***.

Враховуючи все вищевказане, алгоритм знаходження останньої цифри n-го члену послідовності Фібоначчі матиме наступний вигляд.

Розв'язок задачі мовою Python

```
a=b=1
for i in range(int(input())%60): a,b=b,b+a
print(a%10)
```

Задача С. "Гра"

1. В цій задачі позиція однозначно визначається через кількість жовтих мішків.

2. Позиція може бути або виграшна для того хто робить хід, або програшна для того хто робить хід. Нічиїх не буває.

3. побудуємо табличку, вякій будемл позначати 0 - якщо позиція програшна, 1- якщо виграшна

якщо в наш хід є 0 жовтих мішків - то наша позиція програшна (бо суперник своїм ходом забрав останній мішок)

якщо в наш хід є 1 мішок - то наша позиція виграшна, бо ми заберем цей мішок і переведемо суперника в позицію 0 (яка програшна)

якщо в наш хід є 2 мішки - то наша позиція програшна, бо своїм ходом зможемо перевести суперника лише в позицію 1 (яка виграшна)

якщо в наш хід є 3,4,5 мішків - то наша позиція виграшна, бо своїм ходом зможемо перевести суперника в позицію 2 (яка програшна)

якщо в наш хід є 6 мішків - то наша позиція програшна, бо своїм ходом не зможемо перевести суперника в жодну з програшних позицій, тільки у виграшні

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

---

0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1

Знайдемо закономірність.

Очевидно, що якщо позиція X програшна, то наступною програшною позицією буде 2*X+2

Оскільки всі інші позиції ми зможемо звести до X

Наприклад якщо X=6, то в усіх позиціяї 7..13 ми зможемо зробити хід, щоб залишити супернику 6 мішків.

А в позиції 14 вже не зможемо.

Отже програшні позиції наступні: 0 2 6 14 30 62 126 254....

Легко помітити (і не важко довести), що всі ці числа виду 2^N - 2

Отже всі позиції в яких кількість жовтих мішків на 2 менше ніж степінь числа 2 - програшні.

Якщо врахувати ще 2 синіх мішка, які є в умові задачі, отримуємо - що в задачі програшні позиції ті в яких вхідне число дорівнює степені двійки.

Отже все що потрібно в цій задачі це:

1) зчитувати числа, поки не зустрінеться число 0

2) перевіряти чи число не є степінню двійки.

Останнє можна перевіряти послідовним діленням на 2 (і перевіркою скільки раз остача від ділення була рівна 1). якщо лише 1 раз - то число є степенем двійки.

Також, наприклад в C++, можна було використати готову функцію __builtin_popcountll() яка рахує кількість одиничних біт в заданому числі

Задача D. "Козаки та прибульці"

Хороша дослідницька задача для заочної олімпіади (хоча досвідчені олімпійці і так знають як її зробити, але для початківців дуже повчальна).

1. перше число - кількість чисел менших за N і взаємно простих з N носить назву функції Ейлера

https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F_%D0%95%D0%B9%D0%BB%D0%B5%D1%80%D0%B0

http://e-maxx.ru/algo/export_euler_function

і дуже просто обчислюється, якщо ми знаємо прості дільники числа N.

До прикладу візьмемо число 24, воно має прості дільники 2,3

Функція Ейлера для числа 24 обчислюється за формулою

24 * (1 - 1/2) * (1 - 1/3) = (24 - 24/2) * (1 - 1/3) = 12 * (1 - 1/3) = 12 - 12/3 = 8

для числа 420, яке має прості дільники 2,3,5,7:

420 * (1 - 1/2) * (1 - 1/3) * (1 - 1/5) * (1 - 1/7) = 96

2. Друге число в задачі можна було знайти знаючи математичний факт, або дослідивши закономірність самостійно.

Наприклад можна було зробити простий цикл від 1 до 100, рахувати кількість дільників в кожного числа і виводити лише ті числа, які мають непарну кількість дільників.

Такий цикл нам вивів би числа 1 4 9 16 25 36 49 64 81 100, і не важко здогадатись, що це числа які є повними квадратами.

Отже лише повні квадрати чисел мають непарну кількість дільників.

Скільки чисел від 1 до X є повними квадратами?

Очевидно, що корінь квадратний з X.

Отже скільки є чисел МЕНШИХ за N, які містять непарну кількісь дільників?

Ціла частина кореня квадратного з (N-1).

Задача E. "Козаки та прибульці-2"

Теж гарна повчальна дослідницька задача для заочної олімпіади.

В задачі нам потрібно знайти кількість перестановок з N чисел таких, в яких жодне число не стоїть на своєму місці.

В комбінаториці така перестановка називається БЕЗЛАД (анг. Derangement, рос. Беспорядок)

https://uk.wikipedia.org/wiki/%D0%91%D0%B5%D0%B7%D0%BB%D0%B0%D0%B4_(%D0%BF%D0%B5%D1%80%D0%B5%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0)

Кількістьтаких перестановок можна шукати за допомогою рекурентної формули

F(n) = (n-1) * (F(n-1) + F(n-2))

отже знаючи F(1) = 0, F(2) = 1, можемо порахувати всі наступні значення

F(3) = (3-1) * (1 + 0) = 2

F(4) = (4-1) * (2 + 1) = 9

F(5) = (5-1) * (9 + 2) = 44

Оскільки результат при n=800 може мати близько 2000 цифр, то для того аби пройти весь набір тестів необхідно було реалізувати методи "довгої арифметики" (додавання довгих чисел і множення довгого числа на коротке)

==========