И така, в задачата се търси броят на n-цифрените двоични числа без водещи нули, които се делят на 3. Като извършим елементарни изчисления, получаваме, че търсеният брой е $C(n) = \begin{cases} \frac{2^{n-1}-1}{3}, \text{при нечетно } n \\ \frac{2^{n-1}+1}{3}, \text{при четно } n \end{cases}$.

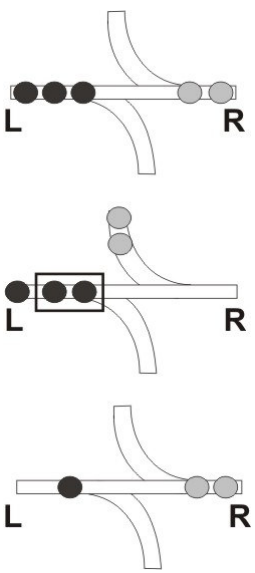
Тъй като ограничението за n е 60, то 64-битово цяло число ще е достатъчно да обхване този брой. По-долу е даден вариант на решението, написан на C/C++:

```
#include <iostream>
using namespace std;
int main(void)
{  int n;
   long long c;
   cin>>n;
   c=(long long)1<<(n-1);
   if (n&1) c--;
   else c++;
   cout<<c/3<<endl;
}
```

Група D

Задача D1. ЧАЙКА, Галина Момчева

На една от улиците в курортен комплекс „Чайка" е паднал пътният знак, който показвал, че улицата е еднопосочна. Автомобили, идващи от двете посоки трябва да се разминат. Но как?



Случайно минаващият от там пешеходец (и начинаещ програмист) Станчо измислил алгоритъм за разминаване. За щастие има достатъчно дълги отбивки в дясно за всяка от колоните. Правилата, които Станчо измислил (според него справедливи) са следните:

1. Дава знак на колоната с по-малко коли да влезе в отбивката си. Ако колите от двете страни са по равен брой, тогава в отбивката влизат колите отляво.
2. Пуска да минат през кръстовището толкова коли от по-дългата колона, колкото е имало в по-късата.
3. Връща колите от отбивката обратно на улицата.

След изпълнението на стъпки 1, 2 и 3 следва нова оценка на ситуацията, която се регулира по същите правила. Не е изключено, обаче, по време на изпълнението на стъпки 1, 2 и 3 да се появят нови коли и от двете страни. Напишете програма **CHAI**, която да помогне на Станчо в регулирането на движението.

Вход

На първия ред на стандартния вход са зададени две цели числа: общият брой N на пристигналите отляво и отдясно коли и броят C на повторенията на стъпките 1, 2 и 3, $0 < N \leq 100$, $0 < C \leq 20$. Следва последователност от $N + C$ реда, на всеки от които има по една от латинските букви `L`, `R` и `P`, като броят на редовете с букви `P` е точно C. Буквата `L` означава, че е пристигнала кола отляво, а буквата `R` – отдясно. Буквата `P` означава, че е извършено регулиране. Входът започва с буква, различна от `P` и завършва с буква `P`. При всяко регулиране, във всяка от двете посоки ще има поне по една кола.

Изход

На първите C реда на стандартния изход, за всяко изпълняване на стъпките 1, 2 и 3, се извежда колко коли са минали през кръстовището и от коя страна на кръстовището са дошли. На последните два реда се извеждат броя на колите останали след последното регулиране в кръстовището, съответно от лявата и дясната част на пътя. Ако няма такива се извежда 0.

ПРИМЕР

Вход	**Изход**
7 2 L R L R L P R R P	2 L 1 R 1 3

Решение

Броят на колите, пристигнали в кръстовището отляво, ще съраняваме в променливата `l`, а тези пристигнали отдясно – в променливата `r`, които в началото имат стойности нула. След това `n+c` пъти се извършва четене на поредният ред и увеличаване на брояча за левите (ако буквата е `L`), увеличаване на брояча за десните (ако буквата е `R`) или извършване на регулиране (ако буквата е `P`). Регулирането е определяне на по-малкото от `l` и `r`. Ако `r` е по-малко от `l`, то колите от дясно влизат в отбивката и същият брой коли от ляво преминават кръстовището, т.е. трябва да се изведе броят на колите отясно и буквата `L` и да се отчете, че колите в ляво са намалели с `r`. Аналогично, ако `l` е по-малко или равно на `r`.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   int n, c;
   cin >> n >> c;
   n += c;
```

```
   int l = 0, r = 0;
   string inp;
   for (int i = 0; i < n; ++i)
   {
      cin >> inp;
      if (inp == "L") ++l;
      if (inp == "R") ++r;
      if (inp == "P")
      {
         cout << min(l, r) << " ";
         if (l > r)
         {
            cout << "L" << endl;
            l -= r;
         }
         else
         {
            cout << "R" << endl;
            r -= l;
         }
      }
  }
  cout << l << endl << r << endl;
  return 0;
}
```

Задача D2. ЧИСЛО НА ЕДИНГТЪН, автор Красимир Манев

Всеки уважаващ себе си велосипедист трябва да знае собственото си *число на Единтън*, а то е най-голямото цяло неотрицателно E такова, че поне E пъти в живота си е изминавал с велосипеда си поне E метра без почивка. Естествено, ако един човек никога не е карал велосипед, числото му на Единтън е нула. Но младият програмист Станчо е и запален велосипедист. Научавайки за числото на Единтън той решил да пресметне, колко е то в неговия случай. Записал на един лист броя N на случаите, в които е бил на седлото на любимия си „байк" без почивка и по колко метра е изминал във всеки от тях. Започнал да пресмята, но числата били толкова много, че скоро се забъркал в сметките. Знанията му по програмиране, пък, не били достатъчни и затова оставя на Вас да се справите с проблема. Напишете програма **EDI**, която да пресмята числото на Единтън.

Вход

На първия ред на стандартния вход ще бъде зададено числото N, $1 \le N \le 100000$. Всеки от следващите N реда ще съдържа по едно цяло положително число L, дължината в метри измината при поредното каране $1 \le L \le 10000$.

Изход

На стандартния изход програмата трябва да изведе намереното число на Единтън.

ПРИМЕР

Вход	Изход
10 3 4 12 5 2 6 2 8 1 7	5

Решение

Това, че числото N е голямо, е без значение. Решението на задачата, което предлагаме, ще може да работи и с много по-големи стойности на N – милиони и даже милиарди. Важното е, че при всяко каране Станчо е изминавал не повече от 10000 метра. Затова дефинираме масив `int a[10001]` и даваме стойност нула на елементите му (първият цикъл `for`). В него ще натрупаме статистиката за каранията, които е осъществил Станчо – в `a[i]` ще намерим колко пъти е изминавал по `i` метра (вторият цикъл `for`). Числото на Единтън намираме в променливата `E` с цикъла `while`. Първата стойност, която проверяваме е 10000. За да има Станчо число на Единтън 10000 е необходимо `a[10000]`≥10000. Ако това е така, изпълнението на цикъла ще се прекрати и програмата ще изведе 10000. Ако това не е така, за да има Станчо число на Единтън 9999 е необходимо `a[9999]+a[10000]`≥9999. Затова, в тялото на цикъла, към съдържанието на `a[E-1]` (в случая `a[9999]`) добавяме съдържанието на `a[E]` (в случая `a[10000]`) и намаляваме стойността на `E` (след `E--` в променливата `E` ще имаме 9999). Така ще накараме оператора `while` да сравни броя на каранията от поне 9999 метра с 9999 и т.н. На всяка стъпка на цикъла, при която `E` още не е търсеното число на Единтън, ще добавяме общия брой на каранията от поне `E` метра към броя на тези от `E-1` метра, за да намерим броя на каранията от поне `E-1` метра и да го сравним с `E-1`. Тъй като дължините са положителни числа, натрупваната сума ще расте, а числото в `E` ще намалява и неравенството `a[E]<E`, рано или късно, ще бъде нарушено. Изпълнението на цикъла ще се прекрати и съдържанието на `E` ще бъде търсеното число на Единтън.

```
#include <iostream>
using namespace std;
int main()
{
   int a[10001],N,i,L,E;
   cin>>N;
   for(i=0;i<=10000;i++) a[i]=0;
```