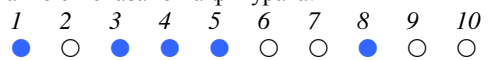
```
//Закръгляне
double round(double x)
{  return floor(x+0.5);  }

//Време в стандартен формат
char *format(double t,char *r)
{  double h,m,s;
   float p=modf(t,&h);
   p=round(p*3600);
   s=fmod(p,60);
   m=floor(p/60);
   sprintf(r,"%.0f:%.0f:%.0f",h,m,s);
   return r;
}
//Намиране на минимално време
double minTime(void)
{  double m,min,x,st;
   Point R;
   R.x=0;R.y=w/2;
   min=getTime(R);
   st=x=0.1; //"едра" стъпка
   do
   {  do
      {  R.x=x;
         m=getTime(R);
         if (m<=min) min=m;
         else break;
         x+=st;
      }while (x<=s/2);
      x-=2*st;
      st/=10;// намаляване на стъпката 10 пъти
      R.x=x;
      m=getTime(R);
      if(fabs(m-min)<0.00001) break;//край
      min=m;
   }while(1);
   return 2*min;
}
int main (void)
{  char buf[16];
   scanf("%lf %lf %lf %lf",&w,&s,&a,&b);
   A.x=0;A.y=w;
   P.x=s/2;P.y=w/2; //Координати на точките A и P.
   printf("%s\n",format(minTime(),buf));
   return 0;
}
```

Задача С3. ОГЪРЛИЦИ, автор Павлин Пеев

Фирма „Гери и Дани" има утвърден оригинален патент за мънистени огърлици в класическите два цвята – синьо и бяло. Освен интересната закопчалка на винт, към патента е включен и уникален метод за подреждане

на мънистата, който е и гаранция за автентичност. Нека при разкопчана огърлица с дължина n да номерираме мънистата, отляво надясно, с целите числа от 1 до n, както е показано на фигурата:



Патентованата подредба се подчинява на следните две правила:

1. Най-лявото мънисто (това с номер 1) е синьо.
2. Нека сините мъниста с нечетни номера са p на брой, а сините мъниста с четни номера – q на брой. Разглеждаме абсолютната стойност d на разликата на числата p и q, $d = |p - q|$. Тя е цяло неотрицателно число. Патентът гласи, че правилна е тази подредба, за която d се дели на 3 без остатък.

Показаната по-горе огърлица, например, не е правилно подредена: тя изпълнява първото условие (мънисто 1 е синьо), но при нея, както се вижда, $p = 3$, $q = 2$, т. е. $d = 1$ и d не се дели на 3. Напротив, огърлиците ●●, ●●○●● и ●○●○●●●○○, например, отговарят и на двете условия.

От фирмата искат да знаят колко са правилните огърлици с определена дължина (брой мъниста). Помогнете им, като напишете програма **NECKLACE**, която решава задачата.

Вход

Програмата трябва да въведе от единствения ред на стандартния вход дължината n ($1 \le n \le 60$) на огърлицата.

Изход

Програмата трябва да изведе броя на различните правилни огърлици с дължина n на стандартния изход.

ПРИМЕР

Вход	**Изход**
6	11

Обяснение: Правилните огърлици с дължина 6 са: ●○○○○●, ●○○○●○○, ●○○●●●, ●○●○●○, ●○●●●○●, ●●○○○○, ●●○○●●, ●●○●●●○, ●●●○○●, ●●●●○○ и ●●●●●●.

Решение

Очевидна е връзката с двоични числа. Ако кодираме синьото мънисто с 1, а бялото – с 0, то зададените правила описват n-цифрените двоични числа без водещи нули, които се делят на 3. Наистина, ако с a_i означим двоичните цифри на едно число A, то $A = \overline{a_1 a_2 a_3 \ldots a_n} = a_1.2^{n-1} + a_2.2^{n-2} + a_3.2^{n-3} + \ldots + a_n$. Като имаме предвид, че $2 \equiv -1 \pmod 3$, то четните степени в това представяне могат да се заместят с 1, а нечетните – с -1 по модул 3. Тогава е ясно, че един признак за делимост на 3 в двоична бройна система е делимостта на 3 на алтернативната сума от двоичните цифри $a_1 - a_2 + a_3 - a_4 + \ldots$. Това изказване всъщност е еквивалентно на формулировката в задачата. Ще припомним само, че, поради аналогични причини, подобно изглежда и известният признак за делимост на 11 в десетична бройна система. И естествено, защото $11_2 = 3$.

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 $C(n) = \begin{cases} \frac{2^{n-1}-1}{3}, \text{при нечетно } n \\ \frac{2^{n-1}+1}{3}, \text{при четно } n \end{cases}$.

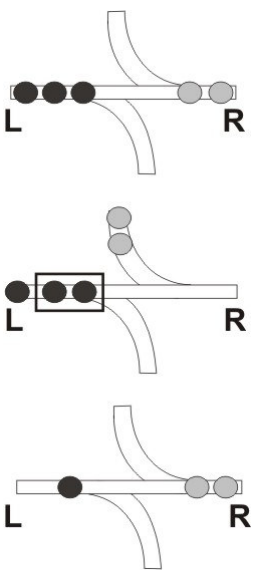
Тъй като ограничението за n е 60, то 64-битово цяло число ще е достатъчно да обхване този брой. По-долу е даден вариант на решението, написан на C/C++:

```
#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;
```