Problem A. Странный прибор

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 секунда
Memory limit:	512 мебибайт

В случае использования данных в шестнадцатеричном виде часто бывает полезно перевести число в двоичную систему, чтобы более наглядно увидеть распределение битов.

Переведя результаты в двоичную систему счисления, получим:

Строка	**Код**
олово	10101
тол	10
остаток	1001010
информатика	10010010101
ромб	100
круг	10

По словам `олово`, `остаток` и `информатика` видно, что единица соответствует гласной букве, а ноль — согласной. Соответственно, если дописать к остальным числам нули спереди так, чтобы количество двоичных цифр совпадало с длиной слова, это же правило верно и для остальных слов. Поэтому осталось закодировать слово `программирование` в соответствии с обнаруженной закономерностью.

Получаем число 0010 0100 1010 1011, или же 24AB в шестнадцатеричной системе.

Problem B. Урок

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

После действий Алисы число стало равно N^2. после действий Боба — $N^2 - 1$, что раскладывается как $(N + 1) \cdot (N - 1)$, после действий Клары — $N \cdot (N - 1) \cdot (N + 1)$. Это произведение трёх подряд идущих целых чисел. Из них хотя бы одно чётно и хотя бы одно делится на три (если ни одно не делится на три, то, так как ненулевых остатков два, а чисел три, то какие-то два остатка совпадают, и разность соответствующих чисел делится на 3, но она по построению меньше 3 — противоречие; чётность доказывается аналогичным образом). Соответственно, их произведение делится и на 2, и на 3, а значит, делится на 6, и остаток всегда равен нулю.

То есть ваша программа должна просто вывести 0.

Problem C. Сортировка дат

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 секунда
Memory limit:	64 мебибайта

Задача про даты — чисто техническая. На языке C/C++ лучше всего использовать функцию форматного ввода/вывода (`printf`/`scanf`) из библиотеки `<stdio.h>`. По сути, нужно прочитать даты в формате "`%d/%d/%d`" и отсортировать их.

Самый удобный вариант — воспользоваться встроенной сортировкой (например, библиотечной функцией `qsort()`) и написать для неё компаратор, который сравнивает сначала год, затем месяц, затем день.

Впрочем, ограничения задачи позволяли отсортировать данные любым известным алгоритмом, например, "пузырьком".

После сортировки даты выводятся в формате "`%04d/%02d/%02d`" (то есть вывод целых чисел с заполнением нулями до 2 или 4 цифр соответственно). Также при прочтении задачи стоит обратить внимание на то, что порядок частей даты во вводе и выводе разный!

Ниже приведён пример решения на языке C (под C++ также работает).

```
#include <stdio.h>
#include <stdlib.h>

#define N 1000

struct Date
{
  int d; // День
  int m; // Месяц
  int y; // Год
};
//
// Чтобы не сравнивать даты <<по частям>>, будем считать величину <<день+100*месяц+10000*год>>.
// Легко заметить, что соответствующая величина больше тогда и только тогда, когда дата позднее.
//
long int
metric (struct Date d)
{
  return d.d + d.m * 100 + d.y * 10000;
}

int
readDate (struct Date *d)
{
  return (3 == scanf ("%d/%d/%d", &(d->d), &(d->m), &(d->y)));
}

void
printDate (struct Date *d)
{
  printf ("%04d/%02d/%02d\n", (d->y), (d->m), (d->d));
}

int
```

```
cmp (const void *p1, const void *p2)
{
  struct Date d1 = *(struct Date *) p1;
  struct Date d2 = *(struct Date *) p2;
  return metric (d1) - metric (d2);
}

int
main ()
{
  int n, i;
  struct Date a[N + 1];

  scanf ("%d", &n);
  for (i = 0; readDate (&a[i]) == 1; i++);
  if (n != i || n > N)
    {
      fprintf (stderr,
       "ERROR: too many data, only %d numbers are expected\n", n);
      return 7;
    }
  qsort (a, n, sizeof (struct Date), cmp);
  for (i = 0; i < n; i++)
    {
      printDate (&a[i]);
    }
  return 0;
}
```

Problem D. Последовательности

Input file:	*none*
Output file:	*текстовый ввод*
Time limit:	1 секунда
Memory limit:	512 мебибайт

Всего строк длины 64 из символов ‘A’ и ‘B’ — 2^{64} (всего 64 позиции, в каждой позиции независимо возможны два варианта).

Подсчитаем количество строк длины 64, в которых никакие два рядом стоящих символа не являются одинаковыми. Для этого заметим, что выбором первого символа строка определяется однозначно. Действительно, пусть первый символ ‘A’, тогда второй обязан быть ‘B’, третий ‘A’ и так далее. Аналогично строка строится и для первого символа ‘B’. Получаем две строки. А во всех остальных строках, соответственно, хотя бы два одинаковых символа находятся рядом, и количество таких строк равно $2^{64} - 2$. Вычислив ответ любым удобным способом, получаем 18 446 744 073 709 551 614.

Problem E. Оптимизация

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Нам нужно перевести целое число из системы счисления с основанием N в систему счисления с основанием N^k. Пусть $a_m a_n m-1 \dots a_0$ — запись числа X в системе счисления с основанием N^k. Тогда $X = a_0 + N^k \cdot a_1 + \ldots + (N^k)^m \cdot a_n$, где a_i — целые неотрицательные числа, меньшие N^k. Таким образом, в системе счисления с основанием N каждое такое число записывается (с ведущими нулями) ровно k цифрами.

Далее рассмотрим представление X в системе счисления с основанием N: $X = b_0 + b_l \cdot N + \ldots + b_l \cdot N^l$; тем самым $a_0 + N^k \cdot a_1 + \ldots + (N^k)^m \cdot a_n = b_0 + b \cdot N + \ldots + N^l \cdot b_l$. Но на первые k цифр представления $b_0 \dots b_{k-1}$ влияет только a_0 (так как остальные a_i домножены на какую-то степень N^k и тем самым на первые k цифр влиять не могут). То есть младшие k цифр представления b есть просто представление a_0 в N-ичной системе счисления. Вычитаем a_0 из каждой части, делим оба представления на N^k и приходим к предыдущей ситуации с заменой a_0 на a_1. Соответственно, следующие k цифр — это представление a_1 в системе счисления с основанием N и так далее.

То есть для перевода достаточно посчитать разложение каждой N^k-ичной цифры в k N-ичных, после чего, начиная с последней цифры, заменять блок из k цифр на соответствующую N^k-ичную цифру (если последний обрабатываемый блок короче k, он дозаполняется ведущими нулями). Никаких вычислений с самим числом (которое может не влезть ни в один стандартный тип данных) делать не требуется.

```
#include <stdio.h>
#include <string.h>

int dlog (int M, int N)
{
int res=1,j=0;
while (res<M) {res=res*N;j++;}
if (res!=M) return -1;
else return j;
}

char getdigit (int i)
{
if (i<10) return ('0'+i);
else return ('a'+i-10);
}

main()
{
char number[10000],number1[10000];
int i,j,M,N;
int len;
scanf ("%d %d",&N,&M);
scanf ("%s",number);
if (N==M) printf ("%s\n",number);
else
 {
// add some zeroes at the beginning
  len=dlog(M,N);
  j=0;
```

```
  if (strlen(number)%len!=0)
   {
    int d=len-strlen(number)%len;
    for (;j<d;j++)
      number1[j]='0';
   }
   for (i=0;i<strlen(number);i++)
    {
     number1[j++]=number[i];
    }
   number1[j++]=0;
// parsing the block
   j=0;

   while (number1[j])
   {
    int dig=0;
    for (i=0;i<len;i++)
      dig=dig*N+number1[j++]-'0';
   printf ("%c",getdigit(dig));
   }
  printf ("\n");
 }
return 0;
```