«Кин-дза-дзá!» — полнометражный двухсерийный трагикомедийный художественный фильм в жанре антиутопии, снятый на киностудии «Мосфильм» в 1986 году режиссёром Георгием Данелией.
https://ru.wikipedia.org/wiki/Кин-дза-дза!

Задача 1. Плюканский словарь 100 баллов

Ограничение по времени	1 секунда
Ограничение по памяти	1 MiB
Входные данные	стандартный ввод или файл input.txt
Выходные данные	стандартный вывод или файл output.txt

Чатланин Уэф живёт и работает на планете Плюк. В письменной речи плюкан используются слова, состоящие ровно из 5 символов. Для формирования слова используются только заглавные латинские буквы в последовательности латинского алфавита. Словарь плюкан начинается с простого слова «AAAAA», после которого идёт слово «AAAAB» и заканчивается словом «ZZZZZ». Инопланетяне являются настолько развитыми, что в их словаре используются все возможные слова, которые можно составить из латинских букв. Естественно в словаре слова записаны в алфавитном порядке. Требуется определить, под каким порядковым номером значится в словаре указанное слово.

Требуется по указанному слову определить его порядковый номер в словаре.

Формат входных данных

Единственная текстовая строка, содержащая слово, порядковый номер которого в словаре требуется определить.

Формат выходных данных

Одно число – порядковый номер слова.

Пример входных и выходных данных

Стандартный ввод	Стандартный вывод
AAAAB	2
AAABZ	52

Решение. Удобно рассматривать слова Плюкан как 5-ти разрядные числа, записанные в 26-ричной системе счисления.

Пример решения на языке Pascal.

```
program solve_1;
 uses Math, SysUtils;
 var s: string; n: longint; i: byte;
begin
  n := 0; readln(s);
  for i := 5 downto 1 do
   n += trunc((ord(s[i])-ord('A')) * power(26,(5-i)));
  writeln(n+1);
end.
```

Пример решения на языке Python.

```
f = open('input.txt')
a = f.readline().strip()
f.close()
abc = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' #цифры плюканской системы
```

```
#счисления (они же буквы)
s = 0
digit = 26**4 #вес четвертого разряда (начиная с нулевого)
for c in a:   #перебираем цифры числа, переводим из 26-ричной
              #системы счисления в десятичную
    r=abc.find(c)
    s += r*digit
    digit //=26
f = open('output.txt','w')
f.write(str(s+1))
f.close()
```

Задача 2. Работа эцилоппа 100 баллов

Ограничение по времени	1 секунда
Ограничение по памяти	1 MiB
Входные данные	стандартный ввод или файл input.txt
Выходные данные	стандартный вывод или файл output.txt

Эцилопп – представитель власти на Плюке. По долгу службы ему приходится часто встречаться с длинными именами плюкан в документах. А это долгая и утомительная работа, так как имена у плюкан действительно бывают очень длинными (по крайней мере в письменной речи). Как-то командой эцилоппов было замечено, что некоторые буквы плюканского алфавита (который как мы узнали из предыдущей задачи состоит из заглавных символов латиницы) встречаются особенно часто. Такие буквы в обществе эцилоппов принято называть надоедливыми, а если в фамилии есть буква, которая встречается чаще других и она только одна, то такой символ принято называть особо надоедливым.

Например, фамилия «DABADABAD» — имеет максимально-надоедливый символ «A», но фамилия «MKAMK» — не содержит такового, потому что символы «K» и «M» встречаются в ней одинаковое максимальное количество раз.

Требуется по указанной фамилии определить максимально-надоедливый символ.

Формат входных данных

Единственная текстовая строка, содержащая фамилию плюканина, состоящая из N символов ($1 \le N \le 10^7$), для поиска в ней максимально-надоедливого символа.

Формат выходных данных

Максимально надоедливый символ или сообщение «NO» в случае его отсутствия.

Пример входных и выходных данных

Стандартный ввод	Стандартный вывод
DABADABAD	A
CABACBAC	NO

Решение. Для подсчёта встречаемости каждого символа удобно завести массив из 26 элементов. Затем в таком массиве можно найти максимальное значение и если оно только одно, то символ с таким номером и является максимально-надоедливым.

Пример решения на языке Pascal.

```
program solve_2;
  var s: string;
```

```
      let: array['A'..'Z'] of integer;
      n: integer; i,max: char;
      fin, fout: text;
begin
//Инициализация массива частоты встречаемости символов
  for i := 'A' to 'Z' do let[i] := 0;
  AssignFile(fin,'input.txt'); reSet(fin);
  AssignFile(fout,'output.txt'); reWrite(fout);
  readln(fin,s);
  CloseFile(Fin);
//Подсчёт частоты встречаемости символов
  for n := 1 to length(s) do
    let[s[n]] += 1;
  n := 1; max := 'A';
//Поиск символа с максимальной частотой (max)
//и количества таких (n)
  for i := 'B' to 'Z' do begin
    if let[i] = let[max] then n += 1;
    if let[i] > let[max] then begin
      max := i; n := 1;
    end;
  end;
//Если количество символов с максимально частотой >1,
//то максимально-надоедливых нет
  if n > 1 then writeln(fout,'NO') else writeln(fout,max);
  CloseFile(fout);
end.
```

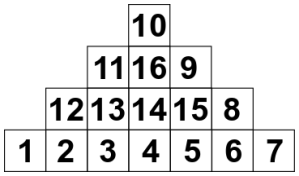
Пример решения на языке Python.

```
#a = input()
abc = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
f = open('input.txt')
a = f.readline().strip()
f.close()
words = {}
for c in abc:
     words.update({c:0})
max1,max2 = 0,0
smax1,smax2 = '',''
for c in a:
     words[c]+=1
for c in abc:
     if words[c]>max2:
          max2 = words[c]
          smax2 = c
     if max2>max1:
          max2,max1 = max1,max2
          smax2,smax1 = smax1,smax2
if max1 == max2: print('NO')
else: print (smax1)
```

Задача 3. Пирамида Господина ПЖ 100 баллов

Ограничение по времени	1 секунда
Ограничение по памяти	4 MiB
Входные данные	стандартный ввод или файл input.txt
Выходные данные	стандартный вывод или файл output.txt

Верховный правитель планеты Плюк – Господин ПЖ решил построить себе дворец в форме пирамиды. Её форма довольно простая, но Господин ПЖ решил внести системность и пронумеровал все блоки по своеобразной спирали: начиная с 1 в левом блоке основания пирамиды и далее.



Немного подумав, Господин ПЖ решил уменьшить количество этажей, затем увеличить, потом снова уменьшить. Процесс его размышлений был весьма долог и многоэтапен. Естественно при этом Господин ПЖ не утруждал себя повторной нумерацией блоков, а предоставлял решение этой задачи пацакам.

Пацаки для удобства, решили записывать нумерацию блоков построчно (каждый уровень в отдельную строку) перечисляя нумерацию блоков слева направо и разделяя номера блоков в строке одним пробелом (см. пример).

Помогите угнетённым пацакам автоматизировать процесс нумерации блоков пирамиды произвольной высоты.

Требуется по указанной высоте пирамиды представить схему нумерации блоков.

Формат входных данных

Одно число N ($1 \le N \le 10^3$) – высота пирамиды.

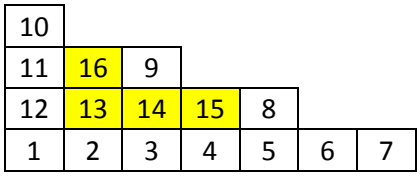
Формат выходных данных

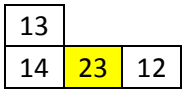
N строк, каждая из которых состоит из необходимого количества чисел, разделённых одним пробелом – номера блоков.

Пример входных и выходных данных

Стандартный ввод	Стандартный вывод
3	7 8 9 6 1 2 3 4 5
4	10 11 16 9 12 13 14 15 8 1 2 3 4 5 6 7

Решение. Один из путей решения задачи заключается в моделировании процесса. Для того, чтобы понять что происходит при нумерации блоков пирамиды описанным способом приведём несколько пирамид (высотой от 4 до 9) и отметим цветом отдельные, повторяющиеся элементы.





15	24	25	22	11				
16	17	18	19	20	21	10		
1	2	3	4	5	6	7	8	9

16										
17	30	15								
18	31	36	29	14						
19	32	33	34	35	28	13				
20	21	22	23	24	25	26	27	12		
1	2	3	4	5	6	7	8	9	10	11

19												
20	37	18										
21	38	47	36	17								
22	39	48	49	46	35	16						
23	40	41	42	43	44	45	34	15				
24	25	26	27	28	29	30	31	32	33	14		
1	2	3	4	5	6	7	8	9	10	11	12	13

22														
23	44	21												
24	45	58	43	20										
25	46	59	64	57	42	19								
26	47	60	61	62	63	56	41	18						
27	48	49	50	51	52	53	54	55	40	17				
28	29	30	31	32	33	34	35	36	37	38	39	16		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

25																
26	51	24														
27	52	69	50	23												
28	53	70	79	68	49	22										
29	54	71	80	81	78	67	48	21								
30	55	72	73	74	75	76	77	66	47	20						
31	56	57	58	59	60	61	62	63	64	65	46	19				
32	33	34	35	36	37	38	39	40	41	42	43	44	45	18		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Пример решения на языке Pascal.

```
program solve3;
  var n,i,j,b,k,x,y,d: integer;
      a: array of array of integer;
      fout: text;
begin
  readln(n);
  SetLength(a,n,2*n-1);
//Перый элемент
  k := 1;
```

```
//Нарисуем необходимое число спиралей спирали в массиве
  for b := 0 to (n+1) div 2 + 1 do begin //b - Номер спирали
    x := b; y := n-1-b; //Положение начала движения
    d := (2*n-1)-b*4; //Длина нижней части (основания) спирали
  //Заполняем нижнюю часть витка
    for i := 1 to d  do begin
      a[y,x] := k;
      k += 1; x += 1;
    end;
  //Заполняем правую часть витка
    x -= 1;
    for i := 1 to d div 2 do begin
      x -= 2; y -= 1;
      a[y,x] := k;
      k += 1;
    end;
  //Заполняем левую часть витка
    for i := 1 to d div 2 - 1 do begin
      y += 1;
      a[y,x] := k;
      k += 1;
    end;
  end;
//Вывод полученной пирамиды
assignfile(fout,'output.txt'); rewrite(fout);
 for i := 0 to n-2 do begin
   x := 0;
   while a[i,x] <> 0 do begin
     write(fout,a[i,x],' ');
     x += 1;
   end;
   writeln(fout);
 end;
 for i := 0 to 2*n-2 do write(fout,a[n-1,i],' ');
 closefile(fout);
end.
```

Задача 4. Подарки Деконт — 100 баллов

Ограничение по времени	1 секунда
Ограничение по памяти	1 MiB
Входные данные	стандартный ввод или файл input.txt
Выходные данные	стандартный вывод или файл output.txt

Деконт (помощница Абрадокса, высокопоставленного чиновника планеты Альфа) дарит цветы жителям планеты Узм (247 в тентуре, галактика Бета в спирали), которые путешествуют по Вселенной при помощи машинки перемещения. Для этого у неё есть полка, на которой стоят цветы в один ряд. Деконт дарит букеты из трёх одноцветных цветов, которые стоят рядом друг с другом. После того как букет подарен, оставшиеся цветы сдвигаются на освободившееся место. Для учёта, каждый из цветов кодируется заглавной буквой латинского алфавита, обозначающей конкретный цвет.

Требуется определить, сколько букетов сможет подарить Деконт и сколько цветов останется на полке?

Формат входных данных

Одна строка, состоящая из N ($1 \le N \le 10^4$) латинских заглавных букв – коды цвета каждого цветка в ряду.

Формат выходных данных

Два числа, разделённых одним пробелом – количество возможных букетов и количество оставшихся цветков.

Пример входных и выходных данных

Стандартный ввод	Стандартный вывод
RRKKKRRSSSSSSK	4 2

Пояснение к примеру – дарятся букеты KKK, SSS и SSS. Образуется букет RRR. Остаются цветки R и K.

Решение. Сводится к моделированию процесса, описанного в условии задачи.

Пример решения на языке Pascal.

```
program solve_4;
 var ch: char; flag: boolean;
     s, st: string;
     i,n: integer;
     fin, fout: text;
begin
  AssignFile(fin, 'input.txt'); AssignFile(fout, 'output.txt');
  reSet(fin); reWrite(fout);
  readln(fin,s); n := 0;
  CloseFile(fin);
  repeat
//Пусть удаления в строке тройки не было
    flag := true;
//Сформируем все возможные трйки одинаковых из букв латиницы
    for ch := 'A' to 'Z' do begin
      st := ch+ch+ch;
//Если такая тройка есть в строке, то удалим её
      i := pos(st, s);
      if i > 0  then begin
        delete(s, i, 3);
        n += 1;
        flag := false;
      end;
    end;
  until flag;
  writeln(fout,n,' ',length(s));
  CloseFile(fout);
end.
```

Пример решения на языке Python.

```
f = open('input.txt')
a = f.readline().strip()
```

```
f.close
s = a[:2]
a = a[2:]
k = 0
for c in a:
    s += c
    while len (s)>2 and c==s[-2]==s[-3]:
        s = s[:-3]
        k += 1
print (k,len(s))
```

Задача 5. Гала-концерт Цан — 100 баллов

Ограничение по времени	1 секунда
Ограничение по памяти	1 MiB
Входные данные	стандартный ввод или файл input.txt
Выходные данные	стандартный вывод или файл output.txt

Тачаночница Цан – бедная артистка, пацашка, путешествует по Плюку на самоходной тележке и даёт концерты. Совершая своё турне, она отмечала на карте каждый населённый пункт, в котором давала представление.

По завершению турне Цан повесила карту на стену и была поражена масштабностью своей гастрольной деятельности. Определить масштабность, с её точки зрения, можно площадью минимального многоугольника, в который входят все населённые пункты, в которых она давала концерты. Определить масштабность очень важно, ведь от этого зависит принятие решение о завершающем гала-концерте.

Требуется написать программу, определяющую минимальную площадь многоугольника, который содержит все остановки Цан.

Формат входных данных

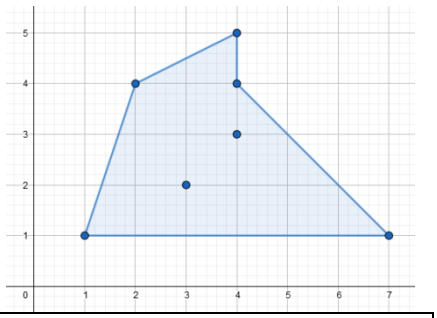
В первой строке входного файла содержится одно целое положительное число – количество остановок N ($3 \le N \le 10^3$).

Каждая из следующих N строк содержит по два разделенных одни пробелом целых числа – сначала координата X, затем координата Y очередного приёмника сигнала ($-10^5 \le X, Y \le 10^5$).

Формат выходных данных

Первая строка выходного файла должна содержать одно число с точностью до двух знаков в десятичной части – площадь минимального прямоугольника, включающего в себя все точки.

Пример входных и выходных данных



Стандартный ввод	Стандартный вывод

7 4 5 2 4 4 4 1 1 4 3 7 1 3 2	13.00

Решение. Предложенная задача состоит из двух частей. Первая, это построение минимальной выпуклой оболочки, включающей в себя все предложенные точки. Вторая, нахождение её площади. Первая задача может быть решена классическими алгоритмами построение выпуклой оболочки обходом Грэхэма или Джарвиса (алгоритм заворачивания подарков). Вторую задачу можно решить, разбив полученный многоугольник на треугольники и найдя их ориентированную площадь.

Пример решения на языке Pascal.

```
program solve_5;
  type tP = record x,y: longint; end;
//Вектор
function vct(a1,a2,b1,b2: tP): integer;
begin
  vct := (a2.x - a1.x)*(b2.y-b1.y)-(b2.x-b1.x)*(a2.y-a1.y)
end;
//Квадрат длины вектора
function dist2(a1,a2: tP): integer;
begin
  dist2 := sqr(a2.x-a1.x) + sqr(a2.y-a1.y);
end;
//Ориентированная площадь треугольника
function tria(x1,y1,x2,y2,x3,y3: integer): real;
begin
  tria := ((x2 - x1) * (y3 - y1) - (y2 - y1) * (x3 - x1)) / 2;
end;

 var fin,fout: text; s: real;
     a, b: array of tP;
     min,m,i,j,k,n: integer;
begin
  assignFile(fin,'input.txt');
  reset(fin);
//Чтение данных
  readln(fin,n);
  SetLength(a, n+1); SetLength(b, n+1);
  for i := 1 to n do
    read(fin, a[i].x, a[i].y);
  closeFile(fin);
//Построение выпуклой оболочки.
//1. Ищем правую нижнюю точку
    m := 1;
    for i := 2 to n do
      if a[i].y < a[m].y
        then m := i
```

```
          else if (a[i].y = a[m].y) and (a[i].x > a[m].x)
            then m := i;
//Запишем её в массив b и переставим на первое место
//в массиве a
    b[1] := a[m];   a[m] := a[1];   a[1] := b[1];
    k := 1;  min := 2;
    repeat
//2. Ищем очередную вершину оболочки
      for j := 2 to n do
        if (Vct(b[k],a[min],b[k],a[j]) < 0) or
           ((Vct(b[k],a[min],b[k],a[j]) = 0) and
           (dist2(b[k],a[min]) < dist2(b[k],a[j])))
          then min := j;
      k += 1;
      b[k] := a[min];
      min := 1;
    until (b[k].x = b[1].x)and (b[k].y = b[1].y);
//Поиск точек, пока ломанная не замкнётся
//Найдём площадь по известной ломаной многоугольника,
//включающего все точки
  s := 0;
  for i := 2 to k-2 do
    s += tria(b[1].x,b[1].y,b[i].x,b[i].y,b[i+1].x,b[i+1].y);
//Вывод
  assignFile(fout,'output.txt'); reWrite(fout);
  write(fout, s:0:2); closeFile(fout);
end.
```