Задача 1. Над златом чахнет

В честь нового года Кощей решил провести лотерею среди лесных жителей, одарив их своим златом. Всего было выпущено 200 000 билетов.

По правилам лотереи перед розыгрышем можно выбрать любой билет, но только один. Иван Царевич очень хочет выиграть и ухитрился узнать заранее выигрышный номер.

Номера билетов формируются следующим образом:

1. первый билет имеет номер 1

2. второй билет имеет номер 1

3. Остальные номера проставлены по следующему принципу:

билет, лежащий на месте 2n имеет такой же номер, как и билет, лежащий на месте n.

номер билета, лежащего на месте (2n+1) равен сумме номеров билетов, лежащих на местах n и n+1

Входные данные:

1<=n<=2023, выигрышный номер

Выходные данные:

Целое число, наименьший порядковый номер выигрышного билета или фразу "nobody gets my money", если билета с таким номером нет

Максимальное время работы программы:

1 сек

Решение, правильно работающее только для части случаев, получит оценку в соответствии с процентом верных ответов на тестовых данных

Для примера:

Входные данные	Результат
2023	145003
100	1179
200	4659
150	5619

Решение на Python

```
def main(k):
    ar = [None] * 200000
    ar[0] = 1
    ar[1] = 1
    ar[2] = 1

    if k==1:
        print(1)
        return 1
    if k==2:
        print(1)
        return 1
    for i in range(3,200000):
        ar[i]= ar[i//2] + ar[i//2+1] * (i%2)
        if k==ar[i]:
            print(i)
            return i
    print('nobody gets my money');

def dec_to_bin(x):
    return int(bin(x)[2:])

k = int(input())
main(k)
```

Задача 2. Забор для тридевятого царства

Царь постановил возвести ограду прочную, чтобы отделить земли своего царства великого, тридевятого от прочих.
Инженерам поставлена задача - начертить границу вокруг владений царя, используя символы +, -, | (вертикальная черта) и пробелы
Данные о владениях - координаты точек, принадлежащих царству.

Известно следующее:

- Царство может быть различной формы
- В центре полученной фигуры могут быть пропуски
- Внутри царства заборы не нужны
- Царство состоит из одной фигуры
-

Примеры:

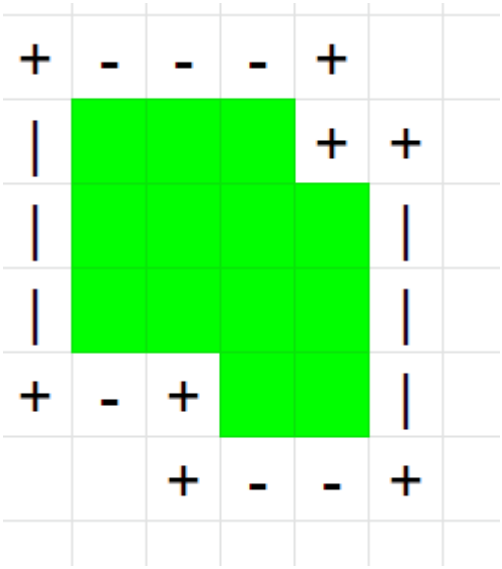


рис. 1 - тридевятое царство

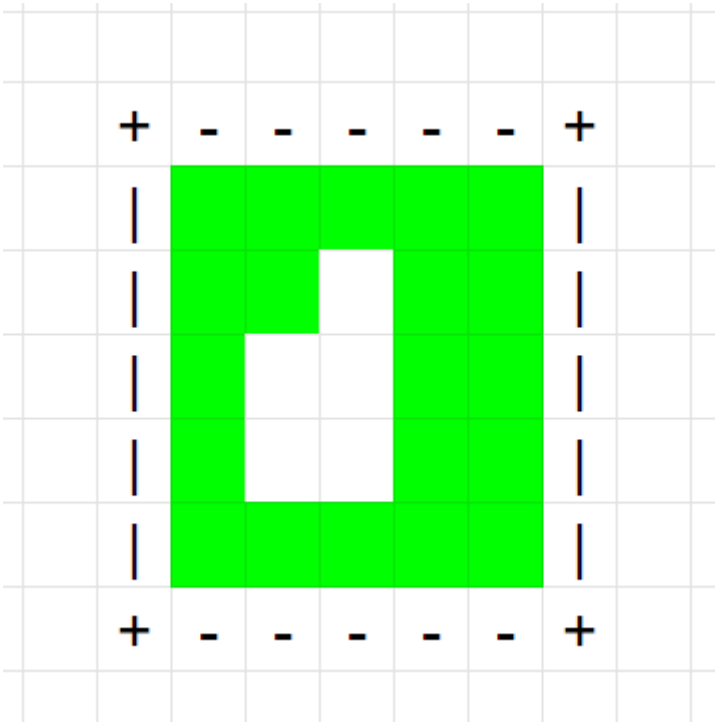


рис. 2 - тридевятое царство с внутренним пространством

Допустимые значения:
Размер области: 1-2000 единичных квадратов
Длина/ширина области: 1-90

Входные данные:
координаты точек x,y, разделенные пробелом, каждая пара с новой строки, -150<=x,y<=150
Выходные данные:
Чертеж.
Требования к выходным данным:
Минимальное количество начальных пробелов(хотя бы одна строка должна начинаться не с пробела)
Отсутствие пробелов в конце строки
Отсутствие лишних переносов строк в конце вывода
Максимальное время работы программы:
1 сек
Решение, правильно работающее только для части случаев, получит оценку в соответствии с процентом верных ответов на тестовых данных
Для примера:

Входные данные	Результат
1 1	+-+ | | +-+

Входные данные	Результат
1 1 2 1	+--+ | | +--+
1 1 2 1 3 1 1 2 2 2 3 2	 +---+ | | | | +---+

Решение на C++

С задачей полностью справились всего три участника. Приводим решение участника.

```
#include <bits/stdc++.h>
using namespace std;

const int N = 305;
const int INF = 1e+3;

char m[N][N];
bool used[N][N];

void dfs(int y, int x)
{
   if (0 <= y and y < N and 0 <= x and x < N and m[y][x] == 0)
   {
      m[y][x] = 1;

      dfs(y - 1, x);
      dfs(y + 1, x);
      dfs(y, x - 1);
      dfs(y, x + 1);
   }
}

void replace(int y, int x)
{
   if (m[y][x] == 1) m[y][x] = '+';
}

void draw(int y, int x)
{
   for (int i = y + 1; y < N and m[i][x] == '+'; i++)
   {
```

```
        m[i - 1][x] = '|';

        if (m[i][x + 1] == '+' and m[i][x + 2] == '+')
        {
            draw(i, x); break;
        }

        while (m[i][x - 1] == '+' and m[i + 1][x] == '+') i++;
    }

    for (int i = x + 1; i < N and m[y][i] == '+'; i++)
    {
        m[y][i - 1] = '-';

        if (m[y + 1][i] == '+' and m[y + 2][i] == '+')
        {
            draw(y, i); break;
        }

        while (m[y - 1][i] == '+' and m[y][i + 1] == '+') i++;
    }

    used[y][x] = true, m[y][x] = '+';
}

signed main()
{
    ios_base::sync_with_stdio(0); cin.tie(0);

    int y, x;

    while (cin >> y >> x)
        m[N - x - 152][y + 152] = ' ';

    dfs(1, 1);

    int sy = INF, sx = INF, ey = -INF, ex = -INF;

    for (y = 2; y < N; y++)
        for (x = 2; x < N; x++)
            if (m[y][x] == ' ')
            {
                sy = min(sy, y - 1);
                sx = min(sx, x - 1);
                ey = max(ey, y + 1);
                ex = max(ex, x + 1);
```

```
            replace(y - 1, x - 1);
            replace(y - 1, x + 1);
            replace(y + 1, x - 1);
            replace(y + 1, x + 1);
            replace(y - 1, x);
            replace(y + 1, x);
            replace(y, x - 1);
            replace(y, x + 1);
        }

    for (y = 2; y < N; y++)
        for (x = 2; x < N; x++)
            if (!used[y][x] and m[y][x] == '+')
                draw(y, x);

    for (y = sy; y <= ey; y++)
    {
        for (x = sx; x <= ex; x++)
            if (m[y][x] != ' ' and m[y][x] != '|' and m[y][x] != '-' and m[y][x] != '+')
                cout << ' ';
            else
                cout << m[y][x];

        cout << '\n';
    }
}
```

Задача 3. 2048

У детворы тридесятого царства новая забава: они узнали про популярную игру 2048. 2048 — браузерная игра, написанная 19-летним итальянским разработчиком Габриэле Чирулли (итал. Gabriele Cirulli) на языке программирования JavaScript. Игровое поле имеет форму квадрата 4x4. Целью игры является получение плитки номинала «2048» (при желании можно продолжить дальше). Код игры открыт и выложен на странице разработчика в GitHub
Только вот играют они пока, рисуя игровое поле мелом на доске, поэтому один тур длится несколько дней.
Чтобы сложности не мешали веселью, взялись они за разработку этой игры, пользуясь тем, чему научились в школе на уроках информатики. Первая задача, которую им предстоит решить - вычисление изменений одной строки игрового поля после сдвига влево
Одна строка игрового поля задается четырьмя числами. Каждое из этих чисел - степень числа 2. При сдвиге влево соседние элементы, если на них одинаковое число, объединяются в один, в два раза большего номинала. Следующие элементы сдвигаются

на освободившееся место. Элементы, разделенные свободным местом(нулем) также считаются соседними.
Операция выполняется слева направо, и если возможно несколько таких объединений, они выполняются за один ход.

Входные данные:

Одна строка, содержащая 4 целых неотрицательных числа n_k<=1024, разделенных пробелами

Выходные данные:

Вид строки после преобразования. Одна строка, содержащая 4 целых неотрицательных числа n_k<=2048, разделенных пробелами

Максимальное время работы программы:

1 сек

Решение, правильно работающее только для части случаев, получит оценку в соответствии с процентом верных ответов на тестовых данных

Для примера:

Входные данные	Результат
2 0 2 2	4 2 0 0
4 4 8 16	8 8 16 0
0 0 0 0	0 0 0 0

Решение на Python

```
def merge(line):
    lst = [x for x in line if x != 0]
    for i in range(len(lst) - 1):
        if lst[i] == lst[i + 1]:
            lst = lst[:i] + [lst[i] + lst[i + 1]] + lst[i + 2:] + [0]
    return ' '.join(list(map(str,(lst + [0] * (len(line) - len(lst))))))

line = list(map(int,input().split()))
print(merge(line))
```

Задача 4. Градиент

Елена Прекрасная взялась за изучение дизайна. Понравилось ей одно слово заморское "градиент". И стала она изображения свои в разные цвета раскрашивать, с использованием градиента.
Градие́нт (англ. Gradient) — вид заливки в компьютерной графике, которая по заданным параметрам цвета в ключевых точках рассчитывает промежуточные цвета остальных точек. При этом создаются плавные переходы из одного цвета в другой. Обычно в градиенте можно использовать более двух цветов и дополнительно указывать настройки прозрачности и смещения границы цветов.
Елена размещает на холсте три точки: одна содержит красный цвет максимальной яркости, другая - зеленый, третья - синий. Цвета расходятся от центров кругами и смешиваются. И хочется ей знать, какой цвет получается в случайно выбранной четвертой точке.

При вычислениях считать, что заливка распространяется на соседние координаты, теряя одну единицу цвета и три цвета распространяются независимо:

			FD			
		FD	FE	FD		
	FD	FE	FF	FE	FD	
		FD	FE	FD		
			FD			

рис. 1 - схема распространения одного цвета

3	FCFBFA	FBFCFB	FAFBFC	
2	FDFCFB	FCFDFC	FBFCFD	
1	FEFDFC	FDFEFD	FCFDFE	
0	FFFEFD	FEFFFE	FDFEFF	
	0	1	2	

рис. 2 - распространение трех цветов, координаты красного (0,0), зеленого (1,0), синего - (2,0)

Для вычисления значения цвета используется HEX-модель.

HEX происходит от слова Hexadecimal (в переводе "шестнадцать"). В веб-дизайне используется так называемый шестнадцатеричный код цвета #RRGGBB, RR – красный, GG – зеленый и BB – синий. Каждая доля цвета находится в диапазоне от 00 до FF.

Пример цвета пастельно-розовый: #FFD4DD

Входные данные:

1 строка - координаты x и y красной точки

2 строка - координаты x и y зеленой точки

3 строка - координаты x и y синей точки

4 строка - координаты x и y точки, цвет которой надо определить

Выходные данные:

HEX-код(HTML-код) цвета, который получается в четвертой точке.

Ограничения на значения:

0<=x,y<=1024

Максимальное время работы программы:

1 мин

Решение, правильно работающее только для части случаев, получит оценку в соответствии с процентом верных ответов на тестовых данных

Для примера:

Входные данные	Результат
0 0 0 256 0 512 0 768	#000000
0 0 0 256 0 512 128 0	#7F0000
10 10 20 20 30 30 40 40	#C3D7EB

Решение на языке Python

```
def componenta(d1,d2):
    return format(max(255 - abs(d1[0] - d2[0]) -  abs(d1[1] - d2[1]),0),'02X')


d1 = list(map(int,input().split()))
d2 = list(map(int,input().split()))
d3 = list(map(int,input().split()))
t = list(map(int,input().split()))

print ('#'+componenta(d1,t)+componenta(d2,t)+componenta(d3,t))
```

Задача 5. Книга рецептов

Баба-Яга получила в подарок книгу с заморскими рецептами зелий и снадобий. Да только непонятно там как-то написано, не по-нашему. Например:
Pepper(MouseWater2)4. Язык английский Баба-Яга знает в совершенстве, но это знание ей не помогло.
Загрустила Яга, такая книга хорошая, да ничего не понятно. И попросила Василису Премудрую написать программу для расшифровки принятой в книге нотации.

Входные данные:
1 строка - рецепт.
В рецепте допустимы круглые, квадратные и фигурные скобки, цифра за скобками обозначает, что все, находящееся в скобках, берется n раз.

Каждый ингридиент начинается с заглавной буквы.

Выходные данные:

Каждый ингридиент с новой строки.

Формат строки: ингридиент и количество, разделенные пробелом

Ограничения на значения:

Входные данные содержат буквы латинского алфавита, скобки и цифры

Максимальное время работы программы:

1 сек

Решение, правильно работающее только для части случаев, получит оценку в соответствии с процентом верных ответов на тестовых данных

Для примера:

Входные данные	Результат
`Pony2Rainbow5`	`Pony 2` `Rainbow 5`

`Pepper(MouseWater2)4`	`Pepper 1`
	`Mouse 4`
	`Water 8`

`Frog2Hair1Haredroppings(Breachbark)3Water10`	`Frog 2`
	`Hair 1`
	`Haredroppings 1`
	`Breachbark 3`
	`Water 10`

Решение на языке Python

Решение с использованием регулярных выражений

```
from collections import Counter
import re

COMPONENT_RE = (
 r'('
 r'[A-Z][a-z]*'
 r'|'
 r'\([^()]+\)'
 r'|'
 r'\[[^[]]+\]'
 r'|'
 r'\{[^}]+\}'
 r')'
 r'(\d*)'
)


def parse_recipe(formula):
 counts = Counter()
```

```
    for element, count in re.findall(COMPONENT_RE, formula):
        count = int(count) if count else 1
        if element[0] in '([{':
            for k, v in parse_recipe(element[1:-1]).items():
                counts[k] += count * v
        else:
            counts[element] += count
    return counts

counts = parse_recipe(input())
dict_val = dict(counts.items())
for key in dict_val:
    print(key, dict_val[key])
```

Решение призера отборочного тура

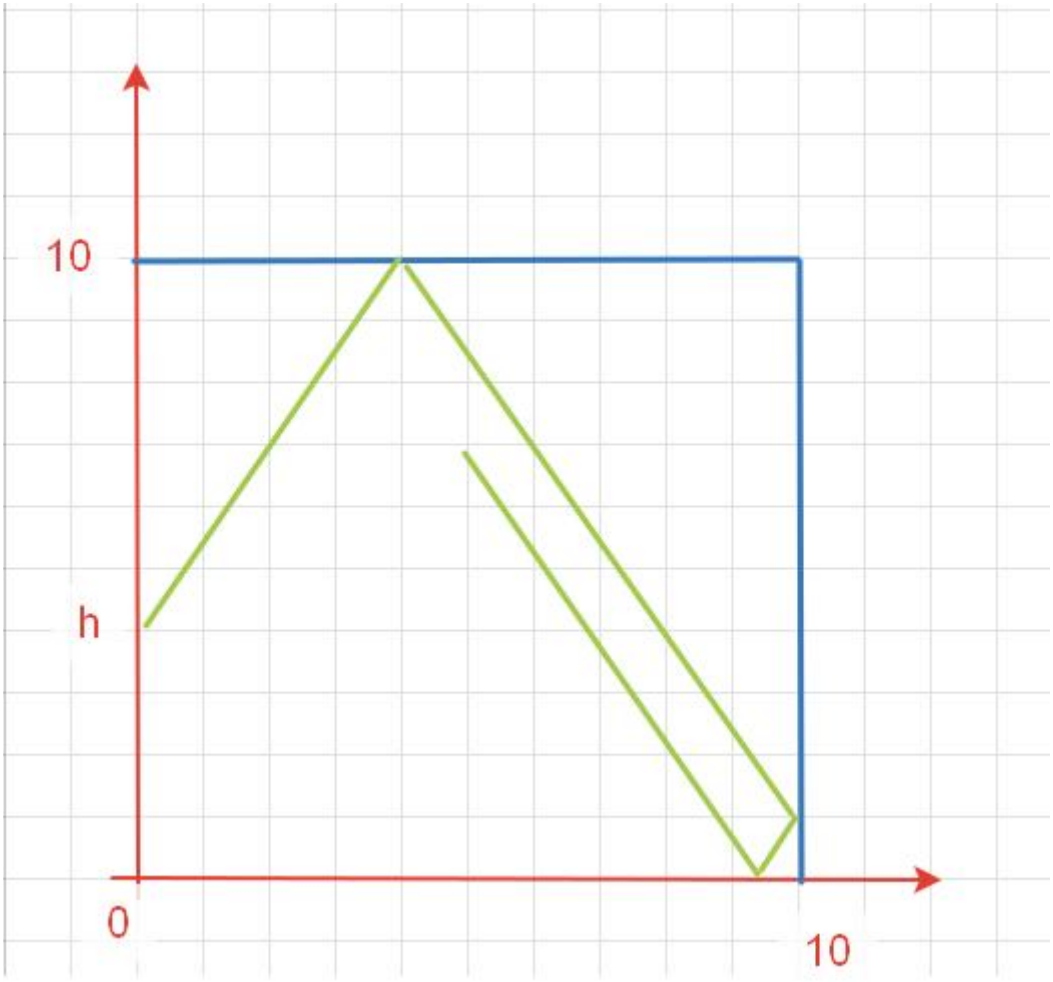
```
string = input()
lst = []
word = ''
nums = 1
sc = 0
dg = ''
for nnn, i in enumerate(string):
    if (nnn == 0 and i not in ["(", "[", "{", ")", "]", "}"])or (not i.isupper() and not i.isdigit() and i not
in ["(", "[", "{", ")", "]", "}"]):
        word += i
    elif i.isupper() and nnn != 0:
        if dg:
            nums = int(dg)
            dg = ''
        if word == '':
            nums = 1
            word = i
        else:
            lst.append([word, nums, sc])
            word = i
            nums = 1

    elif i in ["(", "[", "{", ")", "]", "}"]:
        if dg:
            nums = int(dg)
            dg = ''
        if word != '':
```

```
            lst.append([word, nums, sc])
        word = ''
        if i in ["(", "[", "{"]:
            sc += 1
        else:
            n = max(list(map(lambda x: x[2], lst)))
            if nnn < len(string) - 1:
                if string[nnn + 1].isdigit():
                    add = ''
                    c = 0
                    while True:
                        if nnn + 1 + c <= len(string) - 1 and string[nnn + 1 + c].isdigit():
                            add += string[nnn + 1 + c]
                            c += 1
                        else:
                            break
                    if len(add) >= 1:
                        add = int(add)
                    else:
                        add = 1

                    for a in lst:
                        if a[2] == n and a[2] != 0:
                            a[2] -= 1
                            a[1] *= add
            sc -= 1
    elif i.isdigit() and word != '':
        dg += i

if len(word) >= 1:
    if dg:
        nums = int(dg)
    else:
        nums = 1
    lst.append([word, nums, sc])
    word = ''

sv = {}
for i in lst:
    if i[0] not in sv.keys():
        sv[i[0]] = i[1]
```

```
    else:
        sv[i[0]] += i[1]
for i in sv.keys():
    print(f'{i} {sv[i]}')
```

Задача 6. Лазерная указка

Получил Емеля от щуки лазерную указку. Чтобы не ходить никуда, а просто показывать на нужную вещь и она сама в руки прилетала. И, разумеется, сразу приступил к тестам у себя в избе. То на одну стену посветит, то на другую, а свет еще и отражается, как будто все стены из зеркала.
Комната, в которой проходит тестирование, имеет размеры 10*10 метров. Емеля включает указку из произвольной точки, всегда у одной и той же стены комнаты, направляя ее параллельно полу, и смотрит, где луч отразился от стены.
Если свет луча попал в угол, считать, что он отражается обратно по той же траектории, по которой в угол попал.



Входные данные:

Одна строка, содержащая числа, разделенные пробелом:

Значение h координаты из которой луч указки начинает свое движение, 0<=h<=10

Угол между лучом и стеной 0<=a<90

Длина луча 0.5<=l<20000000000

Выходные данные:

Координата x, y, в которой луч затухнет, округленная до трех знаков после запятой

Максимальное время работы программы:

1 сек

Решение, правильно работающее только для части случаев, получит оценку в соответствии с процентом верных ответов на тестовых данных

Для примера:

Входные данные	Результат
5 60 18	9.0 0.588
5 60 25	7.5 6.651
5 60 18000000007.5	3.75 0.385

Решение задачи на языке C++

Решение одного из призеров отборочного тура

```
#include <iostream>
#include <cmath>

using namespace std;

double h;
double a;
double l;

double x, y;
#define PI 3.14159265358979323846264338327
inline double toRadians(double a) {return a * (PI / 180.0);}
double round3(double value) { return round(value * 1000) / 1000.0; }
inline void out(double a) {
	a = round3(a);
	if (a == double(int(a)))
		printf("%.1f", a);
	else std::cout << a;
}

void solve() {
	a = toRadians(a);
	//std::cout << sin(a) << "\n";
	x = l * cos(a);
	y = l * sin(a)+ h;
	long long xc = (long long)x / 10;
	long long yc = (long long)y / 10;
	x = x - xc * 10.0;
	y = y - yc * 10.0;
	if (xc % 2 == 1) x = 10.0 - x;
	if (yc % 2 == 1) y = 10.0 - y;
}
int main() {
	cin >> h >> a >> l;
	solve();
	out(x);
	std::cout << ' ';
	out(y);
	return 0;
}
```

Решение задачи на Python

```
from math import sin, cos, radians

def helper(n):
    q, r = divmod(n, 10)
    return 10-r if q%2 else r

def laser_coord(h, a, l):
    return helper(l * cos(radians(a))), helper(h + l * sin(radians(a)))

(h,a,l) = list(map(float,input().split()))

res = laser_coord(h,a,l)
print(round(res[0],3), round(res[1],3))
```