§2 Второй отборочный этап

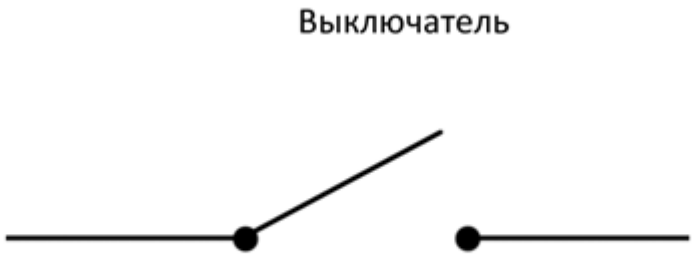
Второй отборочный этап проводится в командном формате в сети интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. Продолжительность второго отборочного этапа — 3 месяца с учетом того, что задачи открываются участникам в онлайн-системе последовательно, на каждую задачу отводится 3 недели. Задачи для 9-11 классов носят междисциплинарный характер и в более простой форме воссоздают инженерную задачу заключительного этапа. Решение каждой задачи дает определенное количество баллов. Баллы зачисляются в полном объеме за правильное решение задачи в зависимости от значимости задачи для финального этапа. В данном этапе можно получить суммарно от 0 до 32 баллов.

Задача 2.1 (3 балла)

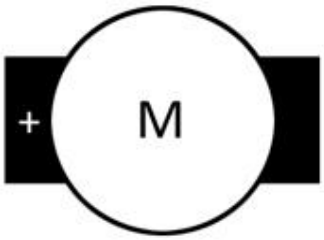
Условие:

Даны следующие компоненты:

- Элемент питания 1 (Б1)
- Элемент питания 2 (Б2)
- Коллекторный электромотор 1 (ЭМ1)
- Коллекторный электромотор 2(ЭМ2)
- Концевой переключатель 1 (КП1)
- Концевой переключатель 2 (КП2)
- Выключатель (В)

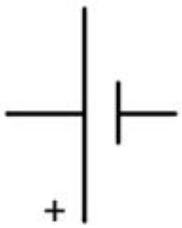


Электромотор



При подключени положительного потенциала к “+” мотор вращается по часовой стрелке

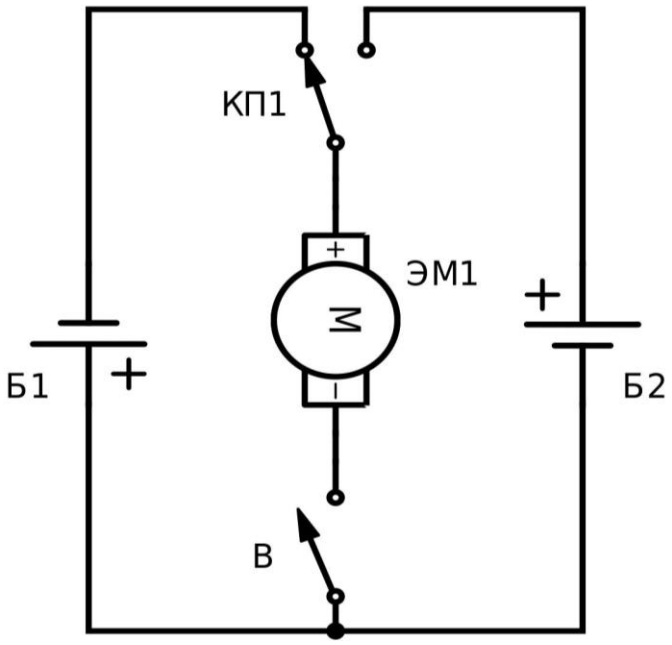
Элемент питания



Разработать электрическую цепь, которая при включении выключателя вращает мотор в одну сторону. При срабатывании концевого переключателя двигатель начинает вращаться в обратную сторону.

Формат ответа: нарисуйте изображение электрической схемы и пояснение в свободном формате (если оно необходимо для понимания схемы) - в одном файле. Файл должен быть назван по названию вашей команды, английскими буквами. От команды должен быть только один файл, независимо от того, сколько участников в команде. Если вы выступаете индивидуально, то вы тоже считаетесь командой из 1 человека.

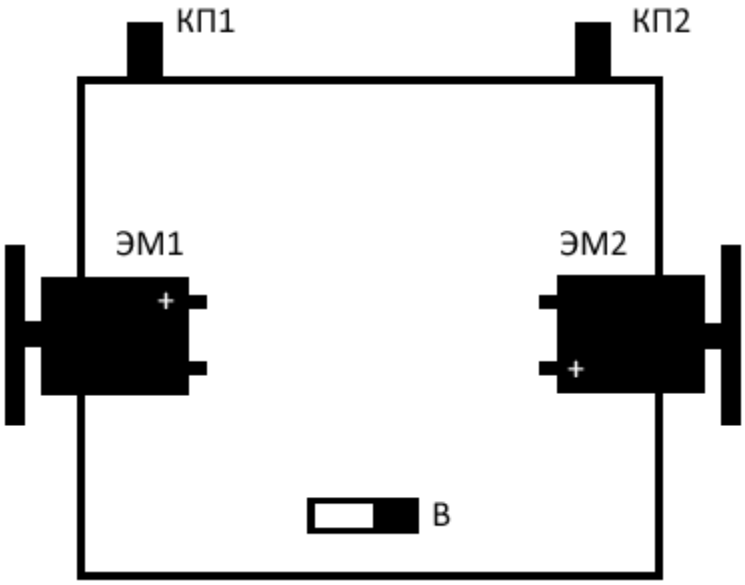
Решение:



Задача 2.2 (5 баллов)

Условие:

Представьте, что вы сделали тележку, изображенную на схеме:

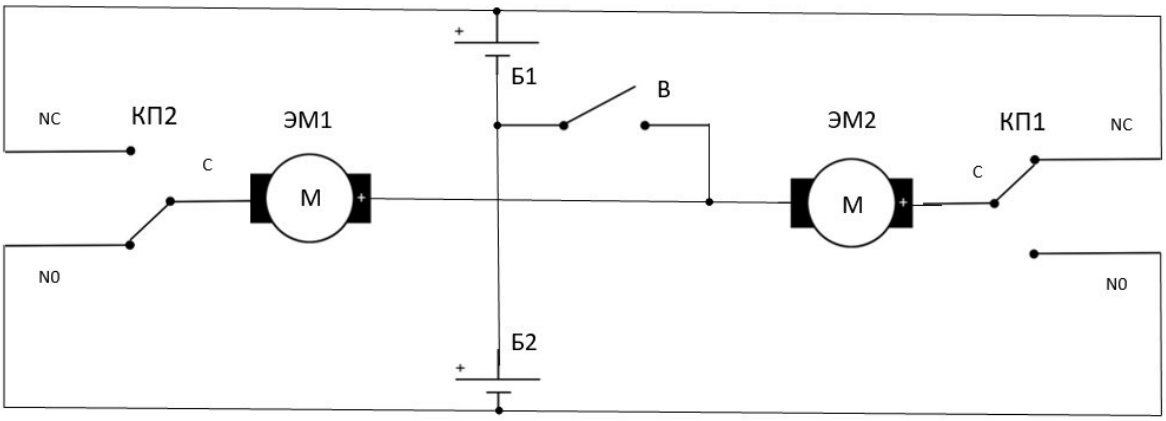


Условная передняя часть тележки – со стороны кнопок-переключателей (КП1 и КП2).

Используя доступные компоненты, разработайте электрическую схему, удовлетворяющую следующим условиям:

- При выключенном выключателе (В) и не нажатых КП1 и КП2 тележка стоит на месте;
- При включении выключателя оба двигателя начинают вращаться таким образом, что тележка едет вперед;
- При нажатии КП1 электродвигатель ЭМ2 начинает вращаться в обратную сторону, ЭМ1 продолжает вращаться вперед;
- При нажатии КП2 электродвигатель ЭМ1 начинает вращаться в обратную сторону, ЭМ2 вращается вперед;
- При нажатии обоих концевых переключателей, оба электродвигателя начинают вращаться назад.

Решение:



Анализ условий задачи:

- Для того, чтобы тележка ехала вперед (при условии, что на рисунке вид сверху, а передняя часть – это то, что КП1 и КП2, то ЭМ1 должен вращаться против часовой стрелки, а ЭМ2 по часовой
- КП1 должен быть в цепи ЭМ2, а КП2 в цепи ЭМ1
- В (выключатель) должен обесточивать оба двигателя.

Задача 2.3 (8 баллов)

Условие:

Точка посадки беспилотного летательного аппарата отмечена с помощью нескольких (не менее 4) светодиодов, расположенных определенным образом (образующих на площадке определенный паттерн). Это, например, могут быть 4 светодиода по вершинам квадрата, 7 светодиодов уголком или любой другой вариант. Бортовой компьютер, используя подключенную к нему камеру, определяет координаты всех небольших ярких объектов в кадре. Камера откалибрована и стабилизирована по двум осям (всегда направлена точно вниз), а диоды расположены на горизонтальной площадке. Однако в кадре, помимо светодиодов посадочной площадки, могут присутствовать другие яркие объекты. Также координаты могут незначительно меняться из-за несовершенства систем коррекции оптических искажений и стабилизации камеры беспилотника.

Требуется написать программу (функцию), принимающую на вход два массива: координаты точек эталонного паттерна и координаты распознанных на кадре ярких точек. Программа должна возвращать координаты распознанных точек, предположительно являющихся светодиодами посадочной площадки.

Поскольку камера откалибрована и стабилизирована, перспективные искажения исправлять не придётся. То есть преобразования, которые необходимо произвести, чтобы совместить эталонный паттерн и распознанные точки такие:

- масштабирование — то есть размер паттерна (в пикселях) может отличаться от размера реальной фигуры из светящихся точек (масштаб зависит от высоты полета).
- поворот — из-за того, что камера не стабилизирована относительно вертикальной оси (по так называемому углу рыскания).
- параллельный сдвиг/перенос — посадочная площадка может находиться в разных частях кадра.

Из-за описанных выше искажений, как правило, паттерн не будет абсолютно точно совпадать с каким-либо поднабором светящихся точек, особенно в случае небольшого числа точек в паттерне. Программа должна возвращать такое подмножество светящихся точек, которое наиболее приближено к паттерну (при этом формализацию понятие “наиболее приближено” мы предлагаем командам определять исходя из здравого смысла самостоятельно, считая это частью задачи).

Требования к решению:

Решения принимаются на любом языке программирования, доступном на платформе stepic.org. Написание красивого и понятного кода строго обязательным требованием не является, поскольку проверка задачи осуществляется при помощи набора тестов. Однако мы рекомендуем писать код какой вы бы написали в рамках тестового задания, проходя собеседование на должность разработчика. Дополнительно рекомендуется (но строго не требуется) кратко описать подход к выбору алгоритма или алгоритмов, использующихся для решения задачи.

Следует понимать, что вычислительные ресурсы на борту ограничены, что накладывает ограничения на вычислительную сложность используемых алгоритмов.

Ограничения на входные данные

Ограничения на количество светящихся точек на площадке соответствует экспериментальным данным, полученным при решении реальных задач.

Ограничения на количество точек в паттерне тоже получены из практики, так как при слишком малом их числе надежно распознать паттерн в зашумленном окружении будет малореально.

Количество точек в паттерне от 5 до 20.

Для количества светящихся точек на площадке могут быть два случая:

Базовый: от 5 до 25 точек (но не менее, чем точек в паттерне) — первая половина проверочного набора данных.

Продвинутый: от 25 до 200 точек — вторая половина проверочного набора.

Следует отметить, что алгоритмы, которые находят точное решение в первом случае, могут не справляться во втором из-за недостатка ресурсов.

Формат входных данных:

Первая строка: число n точек в эталонном паттерне.

Последующие n строк: координаты этих точек (абсцисса и ордината разделяются пробелом).

Затем в (n+2)-ой строке: число m распознанных точек.

В следующих за ней m строках: координаты этих точек, также разделенные пробелом.

Абсциссы распознанных точек принимают значения от 0 до 1920, ординаты — от 0 до 1080.

Формат выходных данных:

n строк с координатами соответствующих паттерну распознанных точек (абсцисса и ордината разделяются пробелом). Число строк должно соответствовать числу точек в эталонном паттерне.

```
Sample Input 1:
  5
  0 0
  10 0
  20 0
  30 0
  40 0
  10
  1220 583
  533 230
  1485 327
  852 849
  1359 453
  1734 923
  1283 521
  1422 394
  1246 1027
  625 544

Sample Output 1:
  1485 327
  1422 394
  1359 453
  1283 521
  1220 583

Sample Input 2:
  7
  30 0
  20 0
  10 0
  0 0
  0 10
```

```
  0 20
  0 30
  15
  1113 908
  159 934
  1135 873
  777 346
  1267 506
  1031 942
  1664 144
  454 1006
  967 896
  1738 439
  999 921
  1917 818
  496 923
  1086 934
  1063 969

Sample Output 2:
  967 896
  999 921
  1031 942
  1063 969
  1086 934
  1113 908
  1135 873

Sample Input 3:
  9
  40 20
  20 40
  0 20
  20 0
  20 20
  30 30
  10 30
  30 10
  10 10
  20
  1037 831
  1793 696
  1242 519
  1655 675
  1299 870
  815 649
  1747 915
  1034 404
  846 839
  907 530
  1337 291
  189 618
  1018 629
  1005 978
  947 941
  1605 379
  727 769
  931 739
```

```
  1679 51
  1118 701

Sample Output 3:
  1118 701
  947 941
  727 769
  907 530
  931 739
  1037 831
  846 839
  1018 629
  815 649

Sample Input 4:
  5
  32 43
  22 35
  13 29
  27 1
  15 16
  25
  514 142
  1666 571
  261 182
  1036 164
  29 161
  1327 1068
  1181 417
  668 729
  531 395
  1420 671
  486 729
  380 478
  1016 753
  1020 924
  1808 520
  440 178
  900 642
  1125 74
  1653 730
  548 286
  868 1056
  846 913
  366 830
  1523 55
  916 720

Sample Output 4:
  29 161
  261 182
  440 178
  486 729
  531 395

Sample Input 5:
  7
  13 49
```

```
  16 13
  48 47
  41 41
  14 46
  22 24
  23 25
  23
  966 144
  229 613
  1139 834
  243 1037
  1106 172
  706 973
  407 710
  1078 39
  267 28
  146 71
  79 509
  283 636
  622 122
  468 768
  856 699
  200 827
  425 571
  439 549
  1097 386
  1189 978
  239 772
  402 708
  291 430

Sample Output 5:
  439 549
  468 768
  229 613
  283 636
  425 571
  407 710
  402 708
```

Решение:

```
import itertools
import math
import time
import random

def read_coord():
   """ Считывает число, а затем координаты """
   newlist = []
   n = int(input())
   for _ in range(n):
       x, y = map(float, input().split())
       newlist.append((x, y))
   return newlist

def rot(x, y, alpha):
```

```
    """ поворот точки относительно начала координата угол alpha """
    _x = x * math.cos(alpha) - y * math.sin(alpha)
    _y = x * math.sin(alpha) + y * math.cos(alpha)
    return (_x, _y)

# Чтение паттерна
pattern = read_coord()

# Чтение распознанных точек
detect = read_coord()

# задаем отклонение(степень похожести) для начала бесконечно большое
resotkl = 99999999
# результат
resdot = []
# Выбор опорных точек
# op1 = 0
# op2 = len(pattern)-1
t = time.clock()
lisdot = list(range(len(pattern)))
random.shuffle(lisdot)
for op1, op2 in itertools.combinations(lisdot, 2):
    if time.clock() > 9:
        break
    for coord1, coord2 in itertools.permutations(detect, 2):
        if time.clock() > 9:
            break
        # coord1 и coord2 две разные точки
        # совмещаем паттерн с этими точками, а также вычисляем угол
поворота(относительно первой точки) и маштаб

        # смещение
        # dx, dy = coord1[0]-pattern[0][0], coord1[1]-pattern[0][1]
        # a, b это вектор направления от точки coord1 к coord2
        a, b = coord2[1] - coord1[1], coord2[0] - coord1[0]

        # длины отрезков между выбранными точками
        l1 = math.sqrt(a ** 2 + b ** 2)
        l2 = math.sqrt((pattern[op1][0] - pattern[op2][0]) ** 2 +
(pattern[op1][1] - pattern[op2][1]) ** 2)
        k = l1 / l2

        # cos угла поворота
        cos_alpha = (b * (pattern[op2][0] - pattern[op1][0]) + a *
(pattern[op2][1] - pattern[op1][1])) / (l1 * l2)
        if cos_alpha>1:
            cos_alpha = 1
        elif cos_alpha < -1:
            cos_alpha = -1

        alpha = math.acos(cos_alpha)
             xx1, yy1 = rot(pattern[op2][0] - pattern[op1][0],
pattern[op2][1] - pattern[op1][1], -alpha)
        xx1 = xx1 * k + coord1[0]
        yy1 = yy1 * k + coord1[1]
        xx2, yy2 = rot(pattern[op2][0] - pattern[op1][0],
pattern[op2][1] - pattern[op1][1], alpha)
        xx2 = xx2 * k + coord1[0]
```

```
        yy2 = yy2 * k + coord1[1]
        r1 = (xx1 - coord2[0]) ** 2 + (yy1 - coord2[1]) ** 2
        r2 = (xx2 - coord2[0]) ** 2 + (yy2 - coord2[1]) ** 2
        if r1 > r2:
            alpha = -alpha

        # развернем и отмасштабируем паттерн в rotpat
        rotpat = []
        for i in range(len(pattern)):
            newx, newy = rot(pattern[i][0] - pattern[op1][0],
pattern[i][1] - pattern[op1][1], -alpha)
            newx = newx * k + coord1[0]
            newy = newy * k + coord1[1]
            rotpat.append((newx, newy))

        # ищем для каждой точки ближайшую из набора и считаем сумму
отклонений
        otkl = 0
        ind1, ind2 = detect.index(coord1), detect.index(coord2)
        tmpdot = [ind1, ind2] # первые две точки всегда совпадают
        tmpdetect = detect[:] # временная копия массива detect
        # исключаем первые две точки из списка ближайших
        tmpdetect[ind1] = [-10000, -10000]
        tmpdetect[ind2] = [-10000, -10000]
        for i, coordr in enumerate(rotpat):
            if i!=op1 and i!=op2:
                # в lenotkl будут квадраты отклонений от каждой точки
                lenotkl = [(coord_[0] - coordr[0]) ** 2 + (coord_[1] -
coordr[1]) ** 2 for coord_ in tmpdetect]
                # добавим в отклонение расстояние до самой ближайшей
точки
                otkl += min(lenotkl)
                if otkl > resotkl:
                    # Отклонение уже больше чем лучший результат
                    break
                # индекс этой точки в массиве
                ind = lenotkl.index(min(lenotkl))
                # запомним индекс этой точки
                tmpdot.append(ind)
                # уберем эту точку из дальнейших вычислений (отодвинем
её далеко!)
                tmpdetect[ind] = (-10000, -10000)

        if (resotkl > otkl):
            # если мы нашли лучшее приближение - запомним его
            resotkl = otkl
            resdot = tmpdot

for i in resdot:
   # печать результата
   print('{:.0f} {:.0f}'.format(*detect[i]))
```

Критерии оценки:

```
import math
import random
import time

def generate():
   random.seed()
   input_lists = []
   distorted_patterns_lists = []
   generated_lists = []

      #generated_lists.append(generate_one(5, 10, ((0, 0), (10, 0),
(20, 0), (30, 0), (40, 0))))
      #generated_lists.append(generate_one(7, 15, ((30, 0), (20, 0),
(10, 0), (0, 0), (0, 10), (0, 20), (0, 30))))
      #generated_lists.append(generate_one(9, 20, ((40, 20), (20, 40),
(0, 20), (20, 0), (20, 20), (30, 30), (10, 30), (30, 10), (10, 10))))

   for _ in range(15):
       pattern_pts_count = math.floor(random.triangular(5, 21, 5))
       sum_pts_count = random.randint(pattern_pts_count, 25)
       generated_lists.append(generate_one(pattern_pts_count,
sum_pts_count))

   for i in range(20):
       pattern_pts_count = random.randint(5, 20)
       sum_pts_count = 25 + (i ** 2) // 3
       generated_lists.append(generate_one(pattern_pts_count,
sum_pts_count))

   output = [tuple([str(len(test_pattern)) + "\n" + "\n".join((str(x)
+ " " + str(y) for x, y in test_pattern)) + "\n" +
                str(len(test_points)) + "\n" + "\n".join((str(x) + "
" + str(y) for x, y in test_points)) + "\n",
                distorted_pattern_points]) for test_pattern,
test_points, distorted_pattern_points in generated_lists]

   output.insert(0, ('7\n13 49\n16 13\n48 47\n41 41\n14 46\n22 24\n23
25\n23\n966 144\n229 613\n1139 834\n243 1037\n1106 172\n706 973\n407
710\n1078 39\n267 28\n146 71\n79 509\n283 636\n622 122\n468 768\n856
699\n200 827\n425 571\n439 549\n1097 386\n1189 978\n239 772\n402
708\n291 430\n', [(439, 549), (468, 768), (229, 613), (283, 636),
(425, 571), (407, 710), (402, 708)]))
   output.insert(0, ('5\n32 43\n22 35\n13 29\n27 1\n15 16\n25\n514
142\n1666 571\n261 182\n1036 164\n29 161\n1327 1068\n1181 417\n668
729\n531 395\n1420 671\n486 729\n380 478\n1016 753\n1020 924\n1808
520\n440 178\n900 642\n1125 74\n1653 730\n548 286\n868 1056\n846
913\n366 830\n1523 55\n916 720\n', [(29, 161), (261, 182), (440, 178),
(486, 729), (531, 395)]))
   output.insert(0, ('9\n40 20\n20 40\n0 20\n20 0\n20 20\n30 30\n10
30\n30 10\n10 10\n20\n1037 831\n1793 696\n1242 519\n1655 675\n1299
870\n815 649\n1747 915\n1034 404\n846 839\n907 530\n1337 291\n189
618\n1018 629\n1005 978\n947 941\n1605 379\n727 769\n931 739\n1679
51\n1118 701\n', [(1118, 701), (947, 941), (727, 769), (907, 530),
(931, 739), (1037, 831), (846, 839), (1018, 629), (815, 649)]))
   output.insert(0, ('7\n30 0\n20 0\n10 0\n0 0\n0 10\n0 20\n0
30\n15\n1113 908\n159 934\n1135 873\n777 346\n1267 506\n1031 942\n1664
144\n454 1006\n967 896\n1738 439\n999 921\n1917 818\n496 923\n1086
```

```
934\n1063 969\n', [(967, 896), (999, 921), (1031, 942), (1063, 969),
(1086, 934), (1113, 908), (1135, 873)]))
   output.insert(0, ('5\n0 0\n10 0\n20 0\n30 0\n40 0\n10\n1220
583\n533 230\n1485 327\n852 849\n1359 453\n1734 923\n1283 521\n1422
394\n1246 1027\n625 544\n', [(1485, 327), (1422, 394), (1359, 453),
(1283, 521), (1220, 583)]))

   return output

def generate_one(pattern_pts_num, sum_pts_num, test_pattern=None):
     global input_pts, distorted_pattern_pts
     random.seed()
     if test_pattern is None:
          test pattern =
generate_test_pattern(pattern_pts_num=pattern_pts_num)

     while True:
          scaling = random.randint(1, 20)
          variance = random.randint(0, scaling)
          random_pts_count = sum_pts_num - pattern_pts_num
          test_points, distorted_pattern_points =
generate_test_input(test_pattern, scaling,
random_points_count=random_pts_count, variance=variance)
          for (x, y) in test_points:
               if x < 0 or x > 1920 or y < 0 or y > 1080:
                    break
               for (x1, y1) in test_points:
                    if (x - x1) ** 2 + (y - y1) ** 2 < 4:
                         break
          else:
               break

     input_pts = test_points
     distorted_pattern_pts = distorted_pattern_points
     random.shuffle(test_points)

     return test_pattern, test_points, distorted_pattern_points

def generate_test_pattern(pattern_pts_num, rect=((0, 0), (50, 50))):
     random.seed()
     test_pattern = []
     for _ in range(pattern_pts_num):
          test_pattern.append((random.randint(rect[0][0],
rect[1][0]), random.randint(rect[0][1], rect[1][1])))
     if len(set(test_pattern)) != pattern_pts_num:
          test_pattern = generate_test_pattern(pattern_pts_num, rect)
     return test_pattern

def generate_test_input(pattern, scaling=1, random_points_count=0,
variance=0, rect=((0, 0), (1920, 1080))):
     global distorted_pattern_pts
     random.seed()
     random_shift_x = random.randint(0, rect[1][0] - 50 - variance)
     random_shift_y = random.randint(0, rect[1][1] - 50 - variance)
     # rot_matrix = get_rot_matrix(random.uniform(0, 2*math.pi))
     # pattern_rotated = [rot_matrix.dot(point) for point in pattern]
     rot_angle = random.uniform(0, 2*math.pi)
```

```
    pattern_rotated = [(round(x*math.cos(rot_angle)-
y*math.sin(rot_angle)),
round(x*math.sin(rot_angle)+y*math.cos(rot_angle))) for (x, y) in
pattern]

    # while True:
    test_pts = [(x * scaling + variance + random_shift_x +
random.randint(-variance, variance),
                 y * scaling + variance + random_shift_y +
random.randint(-variance, variance))
                 for (x, y) in pattern_rotated]
        # min_dist = 100
        # for a in test_pts:
        #     for b in test pts:
        #         min_dist = min((a[0]-b[0]) ** 2 + (a[1]-b[1]) **
2, min_dist)
        # if min_dist > 5:
        #     break

    distorted_pattern_pts = test_pts[:]

    for _ in range(random_points_count):
        test_pts.append((random.randint(rect[0][0], rect[1][0]),
random.randint(rect[0][1], rect[1][1])))

    return test_pts, distorted_pattern_pts


def solve(dataset):
   reply = "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]"
   pts = None
   if dataset == '5\n0 0\n10 0\n20 0\n30 0\n40 0\n10\n1220 583\n533
230\n1485 327\n852 849\n1359 453\n1734 923\n1283 521\n1422 394\n1246
1027\n625 544\n':
      pts = [(1485, 327), (1422, 394), (1359, 453), (1283, 521),
(1220, 583)]
   if dataset == '7\n30 0\n20 0\n10 0\n0 0\n0 10\n0 20\n0 30\n15\n1113
908\n159 934\n1135 873\n777 346\n1267 506\n1031 942\n1664 144\n454
1006\n967 896\n1738 439\n999 921\n1917 818\n496 923\n1086 934\n1063
969\n':
      pts = [(967, 896), (999, 921), (1031, 942), (1063, 969), (1086,
934), (1113, 908), (1135, 873)]
   if dataset == '9\n40 20\n20 40\n0 20\n20 0\n20 20\n30 30\n10 30\n30
10\n10 10\n20\n1037 831\n1793 696\n1242 519\n1655 675\n1299 870\n815
649\n1747 915\n1034 404\n846 839\n907 530\n1337 291\n189 618\n1018
629\n1005 978\n947 941\n1605 379\n727 769\n931 739\n1679 51\n1118
701\n':
      pts = [(1118, 701), (947, 941), (727, 769), (907, 530), (931,
739), (1037, 831), (846, 839), (1018, 629), (815, 649)]
   if dataset == '5\n32 43\n22 35\n13 29\n27 1\n15 16\n25\n514
142\n1666 571\n261 182\n1036 164\n29 161\n1327 1068\n1181 417\n668
729\n531 395\n1420 671\n486 729\n380 478\n1016 753\n1020 924\n1808
520\n440 178\n900 642\n1125 74\n1653 730\n548 286\n868 1056\n846
913\n366 830\n1523 55\n916 720\n':
      pts = [(29, 161), (261, 182), (440, 178), (486, 729), (531,
395)]
```

```
   if dataset == '7\n13 49\n16 13\n48 47\n41 41\n14 46\n22 24\n23
25\n23\n966 144\n229 613\n1139 834\n243 1037\n1106 172\n706 973\n407
710\n1078 39\n267 28\n146 71\n79 509\n283 636\n622 122\n468 768\n856
699\n200 827\n425 571\n439 549\n1097 386\n1189 978\n239 772\n402
708\n291 430\n':
       pts = [(439, 549), (468, 768), (229, 613), (283, 636), (425,
571), (407, 710), (402, 708)]
   if pts is not None:
       reply = "\n".join((str(x) + " " + str(y) for x, y in pts)) +
"\n"

   return reply

def check(reply, clue):
   stat = False
   if reply == "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]":
       stat = True
   else:
       split_reply = reply.splitlines()
       parsed_reply = []
       for string in split_reply:
           num = string.split()
           parsed_reply.append([int(num[0]), int(num[1])])
       if sorted(parsed_reply) == sorted(clue):
           stat = True
   return stat
```

Задача 2.4 (1 балл)

Условие:

Этот шаг содержит тесты с 6 по 10 (базовый вариант). Используйте его, если ваша программа не проходит полный тест (шаг 1).

Решение:

```
#include <functional>
#include <iostream>
#include <vector>
#include <unordered_map>
#include <unordered_set>
#include <set>
#include <stdexcept>
#include <limits>
#include <math.h>
#include <stdint.h>


using Int   =  int32_t;
using Index = uint32_t;
using Cell  = uint64_t;
using Model = uint64_t;

template <class A, class B> using    Pair = std::pair<A,B>;
template <class K, class V> using     Map = std::unordered_map<K,V>;
template <class T>          using GridMap = std::unordered_map<Cell,
T>;
template <class T>          using     Set = std::unordered_set<T>;
```

```
template <class T>          using     Vec = std::vector<T>;


void log(std::string s) {
  std::cout << s << std::endl;
}



Model to_model(Index from, Index to, Index point) {
  Model mf = from  & 0xfffff;
  Model mt = to    & 0xfffff;
  Model mp = point & 0xfffff;

  Model result = (mf << 40) | (mt << 20) | mp;
  return result;
}

Index model_from (Model m) {return  m >> 40;}
Index model_to   (Model m) {return (m >> 20) & 0xfffff;}
Index model_point(Model m) {return  m        & 0xfffff;}


Cell components_to_cell(Int cx, Int cy) {
  Cell result;

  Index ux = cx;
  Index uy = cy;

  result = uy;
  result <<= 32;
  result &= ux;

  return result;
}


Cell point_to_cell(double x, double y, double eps) {
  Int cx = floor(x/eps);
  Int cy = floor(y/eps);

  return components_to_cell(cx, cy);
}

template <class T>
struct Grid {
  double          eps;
  GridMap<Vec<T>> map;

  Grid(double eps) : eps(eps) {}

  void add_point(double x, double y, T index) {
    map[point_to_cell(x,y,eps)].insert(index);
  }

  void add_ball(double x, double y, double r, T index) {
    Int cx_min = floor((x-r)/eps);
    Int cy_min = floor((y-r)/eps);
```

```
    Int cx_max = floor((x+r)/eps);
    Int cy_max = floor((y+r)/eps);

    double d_bound  = r + eps * 0.87;
    double dd_bound = d_bound * d_bound;
    for (Int cx = cx_min; cx <= cx_max; cx++) {
      for (Int cy = cy_min; cy <= cy_max; cy++) {
        double x_min = cx*eps;
        double y_min = cy*eps;

        double x_center = x_min + eps/2;
        double y_center = y_min + eps/2;

        double dx = x_center - x;
        double dy = y_center - y;


        if (dx*dx + dy*dy > dd_bound) {break;}

        map[components_to_cell(cx,cy)].push_back(index);
      }
    }
  }
};



struct Point {
  double x;
  double y;
};


using Indices  = std::vector<Index>;
using Points   = std::vector<Point>;
using Distance = double;
using ItoIDMap = std::unordered_map<Index,std::pair<Index,Distance>>;
using DDSet    = std::set<std::pair<double,double>>;


void foreach_transformed_point(Point &start, Point &end, Points
&points, std::function<void(double,double,Index)> f) {
  double dx = end.x - start.x;
  double dy = end.y - start.y;

  double dd = dx*dx + dy*dy;

  for (Index i = 0; i != points.size(); i++) {
    auto &p = points[i];

    double x = p.x - start.x;
    double y = p.y - start.y;

    double tx = (dx*x + dy*y)/dd;
    double ty = (dx*y - dy*x)/dd;

    f(tx,ty,i);
  }
```

```
}



double squared_distance(Point &a, Point &b) {
  double dx = b.x - a.x;
  double dy = b.y - a.y;

  return dx*dx + dy*dy;
}




Grid<Pair<Model,Point>> pattern_to_grid(Points &points, double
transformed radius, double ratio) {
  Grid<Pair<Model,Point>> result(transformed_radius*ratio);

  for (Index i = 0; i != points.size(); i++) {
    for (Index j = 0; j != points.size(); j++) {
      if (j == i) {break;}

      Point start = points[i];
      Point end   = points[j];

      foreach_transformed_point(start, end, points, [&](double bx,
double by, Index k){
        Point p;
        p.x = bx;
        p.y = by;
        result.add_ball(bx, by, transformed_radius,
{to model(i,j,k),p});
      });
    }
  }

  return result;
}



Indices try_image(Point &start, Point &end, Points &image, size_t
pattern_size, Grid<Pair<Model,Point>> &grid, double tolerance, double
&norm) {
  double tt = tolerance*tolerance;

  Map<Model,ItoIDMap>   closest_points;
  Map<Model,Set<Index>> used_points;
  Map<Model,Distance>   max_distance;

  foreach_transformed_point(start, end, image, [&](double tx, double
ty, Index i){
    Cell cell = point_to_cell(tx, ty, grid.eps);
    Distance mult = squared_distance(start, end);
```

```
    if (grid.map.count(cell)) {
      for (auto mp : grid.map[cell]) {
        Model m = mp.first;
        Index j = model_point(m);

        Model truncated_m = to_model(model_from(m), model_to(m), 0);
        if (used_points[truncated_m].count(i)) {continue;}

        double dx = tx - mp.second.x;
        double dy = ty - mp.second.y;
        Distance new_distance = dx*dx + dy*dy;

        if (new_distance > tt) {continue;}
        new_distance *= mult;

        if (closest_points[truncated_m].count(j)) {
          Index    old_index    =
closest_points[truncated_m][j].first;
          Distance old distance =
closest_points[truncated_m][j].second;

          if (new distance < old distance) {
            used_points[truncated_m].erase(old_index);
            closest_points[truncated_m][j] = {i, new_distance};
            used_points[truncated_m].insert(i);
            continue;
          }
        }
        else {
          closest points[truncated m][j] = {i, new_distance};
          used_points[truncated_m].insert(i);
          continue;
        }
      }
    }
  });


  for (auto mu : used_points) {

    if (mu.second.size() == pattern_size) {
      Distance max_d = 0;

      for (auto i_id : closest_points[mu.first]) {
        if (i_id.second.second > max_d) {max_d = i_id.second.second;}
      }

      max_distance[mu.first] = max_d;
    }
  }


  Model    minmax_m = 0;
  Distance minmax_d = std::numeric_limits<Distance>::infinity();


  Indices result;
```

```
  if (max_distance.size() == 0) {return result;}


  for (auto m_md : max_distance) {

    if (minmax_d > m_md.second) {
      minmax_m = m_md.first;
      minmax_d = m_md.second;
    }
  }

  norm = minmax_d;

  for (auto i_id : closest_points[minmax_m]) {
    result.push_back(i_id.second.first);
  }

  return result;
}




void print_match(Points &pattern, Points &image) {
  double t_radius  = 0.1; // РЎ Р-РўР□Рњ РњРћР-РќРћ РџРћР□Р"Р
РђРўР¬РЎРЇ!
  double kappa = 1.145;
  double norm = 0;

  int iterations=0;
  while (true) {
    auto grid = pattern_to_grid(pattern, t_radius, kappa);
    bool found = false;

    for (Index i = 0; i != image.size(); i++) {
      for (Index j = 0; j != image.size(); j++) {
        if (i == j) {continue;}

        Indices a_try = try_image(image[i], image[j], image,
pattern.size(), grid, t_radius, norm);

        if (a_try.size() != 0) {

          for (auto index : a_try) {
            Point p = image[index];

            std::cout << p.x << ' ' << p.y << std::endl;
          }
          return;
        }
      }
    }
    t_radius *= 2;
  }
}
```

```
Points read_array() {
  Index N;
  std::cin >> N;


  DDSet set;

  for (Index i = 0; i != N; i++) {
    double x;
    double y;

    std::cin >> x >> y;
    set.insert({x,y});
  }

  Points result;

  for (auto &p : set) {result.push_back({p.first, p.second});}

  return result;
}


int main() {
  Points pattern = read_array();
  Points image   = read_array();

  print_match(pattern, image);
}
```

Критерии оценки:

```
import math
import random
import time
def generate():
   return []

def solve(dataset):
   reply = "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]"
   pts = None
   if dataset == '5\n0 0\n10 0\n20 0\n30 0\n40 0\n10\n1220 583\n533
230\n1485 327\n852 849\n1359 453\n1734 923\n1283 521\n1422 394\n1246
1027\n625 544\n':
       pts = [(1485, 327), (1422, 394), (1359, 453), (1283, 521),
(1220, 583)]
   if dataset == '7\n30 0\n20 0\n10 0\n0 0\n0 10\n0 20\n0 30\n15\n1113
908\n159 934\n1135 873\n777 346\n1267 506\n1031 942\n1664 144\n454
1006\n967 896\n1738 439\n999 921\n1917 818\n496 923\n1086 934\n1063
969\n':
       pts = [(967, 896), (999, 921), (1031, 942), (1063, 969), (1086,
934), (1113, 908), (1135, 873)]
```

```
   if dataset == '9\n40 20\n20 40\n0 20\n20 0\n20 20\n30 30\n10 30\n30
10\n10 10\n20\n1037 831\n1793 696\n1242 519\n1655 675\n1299 870\n815
649\n1747 915\n1034 404\n846 839\n907 530\n1337 291\n189 618\n1018
629\n1005 978\n947 941\n1605 379\n727 769\n931 739\n1679 51\n1118
701\n':
       pts = [(1118, 701), (947, 941), (727, 769), (907, 530), (931,
739), (1037, 831), (846, 839), (1018, 629), (815, 649)]
   if dataset == '5\n32 43\n22 35\n13 29\n27 1\n15 16\n25\n514
142\n1666 571\n261 182\n1036 164\n29 161\n1327 1068\n1181 417\n668
729\n531 395\n1420 671\n486 729\n380 478\n1016 753\n1020 924\n1808
520\n440 178\n900 642\n1125 74\n1653 730\n548 286\n868 1056\n846
913\n366 830\n1523 55\n916 720\n':
       pts = [(29, 161), (261, 182), (440, 178), (486, 729), (531,
395)]
   if dataset == '7\n13 49\n16 13\n48 47\n41 41\n14 46\n22 24\n23
25\n23\n966 144\n229 613\n1139 834\n243 1037\n1106 172\n706 973\n407
710\n1078 39\n267 28\n146 71\n79 509\n283 636\n622 122\n468 768\n856
699\n200 827\n425 571\n439 549\n1097 386\n1189 978\n239 772\n402
708\n291 430\n':
       pts = [(439, 549), (468, 768), (229, 613), (283, 636), (425,
571), (407, 710), (402, 708)]
   if pts is not None:
       reply = "\n".join((str(x) + " " + str(y) for x, y in pts)) +
"\n"

   return reply

def check(reply, clue):
   clue_arr = eval(clue)
   stat = False
   if reply == "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]":
       stat = True
   else:
       split_reply = reply.splitlines()
       parsed_reply = []
       for string in split_reply:
           num = string.split()
           parsed_reply.append([int(num[0]), int(num[1])])
       if sorted(parsed_reply) == sorted(clue_arr):
           stat = True
   return stat
```

Задача 2.5 (1 балл)

Условие:

Этот шаг содержит тесты с 11 по 15 (базовый вариант). Используйте его, если ваша программа не проходит полный тест (шаг 1).

Решение:

```
#include <ctime>
#include <cmath>
#include <cstdlib>
#include <vector>
#include <iostream>
#include <algorithm>
```

```
using namespace std;

template <class _T>
struct point
{
     _T x, y;
    point(){};
    template<class _T1, class _T2>
    point(_T1 _x, _T2 _y) : x(_x), y(_y) {};
    template <class _T1>
    bool operator==(const point<_T1>& a) const
    {
        return a.x == x && a.y == y;
    }
};
template <class _T1, class _T2>
inline bool cmp_x(const point<_T1>& a, const point<_T2>& b) { return
a.x != b.x ? a.x < b.x : a.y < b.y; }
template <class _T1, class _T2>
inline bool cmp_y(const point<_T1>& a, const point<_T2>& b) { return
a.y != b.y ? a.y < b.y : a.x < b.x; }

vector<point<int> > get_points()
{
    int n; cin >> n;
    vector<point<int> > res(n);
    for (auto& p : res)
        cin >> p.x >> p.y;
    return res;
}

template <class _T1>
long double abs(const point<_T1>& a) { return sqrtl(a.x*a.x +
a.y*a.y); }
template <class _T1, class _T2>
inline point<long double> add(const point<_T1>& a, const point<_T2>&
b) { return point<long double>(a.x+b.x, a.y+b.y); }
template <class _T1, class _T2>
inline point<long double> sub(const point<_T1>& a, const point<_T2>&
b) { return point<long double>(a.x-b.x, a.y-b.y); }
template <class _T1, class _T2>
inline point<long double> mul(const point<_T1>& a, const point<_T2>&
b) { return point<long double>(a.x*b.x - a.y*b.y, a.x*b.y + a.y*b.x);
}
template <class _T1, class _T2>
inline point<long double> divide(const point<_T1>& a, const
point<_T2>& b)
{
    long double d = b.x*b.x + b.y*b.y;
    return point<long double>((a.x*b.x + a.y*b.y)/d, (a.y*b.x -
a.x*b.y)/d);
}

struct pattern
{
    const vector<point<int> > pattern_base;
    vector<point<long double> > pattern_points;
```

```
    pattern(const vector<point<int> >& base) : pattern_base(base) {};

    void move_to(point<int> a, point<int> b)
    {
        point<int> a0 = pattern_base[0];
        point<int> b0 = pattern_base[1];

        /// in complex (b0 - a0)*k = (b - a)
        ///            k = (b - a)/(b0 - a0)
        point<long double> k = divide(sub(b, a), sub(b0, a0));
        pattern_points.clear();
        pattern_points.reserve(pattern_base.size());
        for (auto p : pattern_base)
            pattern_points.push_back(point<long double>(p.x, p.y));

        /// f(z) = k*z + z0
        /// p = (p - a0)*k + a
        /// p = p*k - a0*k + a
        /// z0 = a - a0*k
        point<long double> z0 = sub(a, mul(a0, k));
        for (auto& p : pattern_points)
            p = add(mul(p, k), z0);
    }
};

template <class _T>
using iter = typename vector<point<_T> >::iterator;
template <class _T>
void build_2d_tree(iter<_T> begin, iter<_T> end, bool y_split = false)
{
    if (end - begin < 2) return;
    iter<_T> middle = begin + (end - begin)/2;
    nth_element(begin, middle, end, y_split ? cmp_y<int, int> :
cmp_x<int, int>);
    build_2d_tree<int>(begin, middle, !y_split);
    build_2d_tree<int>(middle+1, end, !y_split);
}

template <class _T1, class _T2>
long double get_dist(iter<_T1> begin, iter<_T1> end, bool y_split,
                     const point<_T2>& t, long double& dist,
point<_T1>& ans)
{
    if (end - begin < 1) return HUGE_VALL;
    iter<_T1> middle = begin + (end - begin)/2;
    long double dx = (t.x - middle->x)*(t.x - middle->x),
                dy = (t.y - middle->y)*(t.y - middle->y),
                d3 = dx+dy;

    if (d3 < dist) dist = d3, ans = *middle;

    if (end - begin == 1) return d3;

    if ((y_split ? cmp_y<long double, int> : cmp_x<long double,
int>)(t, *middle))
    {
```

```
        if ((y_split ? dy : dx) < get_dist(begin, middle, !y_split, t,
dist, ans))
                                    get_dist(middle+1, end, !y_split, t,
dist, ans);
    }
    else
    {
        if ((y_split ? dy : dx) < get_dist(middle+1, end, !y_split, t,
dist, ans))
                                    get_dist(begin, middle, !y_split, t,
dist, ans);
    }

    return dist;
}

template <class _T1, class _T2>
inline
point<_T1> get_nearest_point(vector<point<_T1> >& base, point<_T2> p)
{
    long double dist = HUGE_VALL;
    point<_T1> ans;
    get_dist(base.begin(), base.end(), false,
                                p, dist, ans);
    return ans;
}

template <class _T1, class _T2>
inline
long double get_dist(vector<point<_T1> >& base, point<_T2> p)
{
    long double dist = HUGE_VALL;
    point<_T1> ans;
    return get_dist(base.begin(), base.end(), false,
                    p, dist, ans);
}

struct screen
{
    const vector<point<int> > screen_base;
    vector<point<int> > tree;

    inline
    point<int> get_nearest(point<long double> p)
    {
        return get_nearest_point(tree, p);
    }

    inline
    long double get_dist_to_nearest(point<long double> p)
    {
        return get_dist(tree, p);
    }

    screen(const vector<point<int> >& base) : screen_base(base)
    {
        tree = screen_base;
        build_2d_tree<int>(tree.begin(), tree.end());
```

```
    };

    vector<point<int> > get_score(const pattern& ptrn, long double&
sum)
    {
        sum = 0;
        vector<point<int> > res;
        res.reserve(ptrn.pattern_points.size());
        for (auto p : ptrn.pattern_points)
            res.push_back(this->get_nearest(p)),
            sum += abs(sub(res.back(), p));
        sort(res.begin(), res.end(), [](const point<int>& a, const
point<int>& b){return a.x != b.x ? a.x < b.x : a.y < b.y; });
        if (unique(res.begin(), res.end()) != res.end())
            sum = HUGE_VALL,
            res.clear();
        return res;
    }
};

pair<long double, vector<point<int> > > get_solve(pattern ptrn,
screen& scr)
{
    int n = scr.screen_base.size();

    long double best_score = HUGE_VALL;
    int besti = 0;
    int bestj = 1;

    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            if (i != j)
            {
                ptrn.move_to(scr.screen_base[i], scr.screen_base[j]);
                long double score;
                scr.get_score(ptrn, score);
                if (score < best_score)
                    best_score = score,
                    besti = i,
                    bestj = j;
            }

    vector<point<int> > ans;
    ans.reserve(ptrn.pattern_base.size());

    ptrn.move_to(scr.screen_base[besti], scr.screen_base[bestj]);
    for (auto p : ptrn.pattern_points)
        ans.push_back(scr.get_nearest(p));

    return {best_score, ans};
}

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    srand(time(NULL));
```

```
    vector<point<int> > pattern_points = get_points();
    screen scr(get_points());

    long double best_score = HUGE_VALL;
    vector<point<int> > best_ans;

    int n = pattern_points.size();
    int m = scr.screen_base.size();
    int T = min(150, (int)floorl(25e7/(m*m*m*n)));

    for (int t = 0; t < T; ++t)
    {
        pair<long double, vector<point<int> > > it =
get_solve(pattern(pattern_points), scr);
        if (it.first < best_score)
            best_score = it.first,
            best_ans = it.second;
        random_shuffle(pattern_points.begin(), pattern_points.end());
    }

    //cout << T << " " << best_score << '\n';
    for (auto p : best_ans)
        cout << p.x << ' ' << p.y << '\n';

    return 0;
}
```

Критерии оценки:

```
#Python3
import math
import random
import time


def generate():
   return []

def solve(dataset):
   reply = "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]"
   pts = None
   if dataset == '5\n0 0\n10 0\n20 0\n30 0\n40 0\n10\n1220 583\n533
230\n1485 327\n852 849\n1359 453\n1734 923\n1283 521\n1422 394\n1246
1027\n625 544\n':
       pts = [(1485, 327), (1422, 394), (1359, 453), (1283, 521),
(1220, 583)]
   if dataset == '7\n30 0\n20 0\n10 0\n0 0\n0 10\n0 20\n0 30\n15\n1113
908\n159 934\n1135 873\n777 346\n1267 506\n1031 942\n1664 144\n454
1006\n967 896\n1738 439\n999 921\n1917 818\n496 923\n1086 934\n1063
969\n':
       pts = [(967, 896), (999, 921), (1031, 942), (1063, 969), (1086,
934), (1113, 908), (1135, 873)]
   if dataset == '9\n40 20\n20 40\n0 20\n20 0\n20 20\n30 30\n10 30\n30
10\n10 10\n20\n1037 831\n1793 696\n1242 519\n1655 675\n1299 870\n815
649\n1747 915\n1034 404\n846 839\n907 530\n1337 291\n189 618\n1018
```

```
629\n1005 978\n947 941\n1605 379\n727 769\n931 739\n1679 51\n1118
701\n':
       pts = [(1118, 701), (947, 941), (727, 769), (907, 530), (931,
739), (1037, 831), (846, 839), (1018, 629), (815, 649)]
   if dataset == '5\n32 43\n22 35\n13 29\n27 1\n15 16\n25\n514
142\n1666 571\n261 182\n1036 164\n29 161\n1327 1068\n1181 417\n668
729\n531 395\n1420 671\n486 729\n380 478\n1016 753\n1020 924\n1808
520\n440 178\n900 642\n1125 74\n1653 730\n548 286\n868 1056\n846
913\n366 830\n1523 55\n916 720\n':
       pts = [(29, 161), (261, 182), (440, 178), (486, 729), (531,
395)]
   if dataset == '7\n13 49\n16 13\n48 47\n41 41\n14 46\n22 24\n23
25\n23\n966 144\n229 613\n1139 834\n243 1037\n1106 172\n706 973\n407
710\n1078 39\n267 28\n146 71\n79 509\n283 636\n622 122\n468 768\n856
699\n200 827\n425 571\n439 549\n1097 386\n1189 978\n239 772\n402
708\n291 430\n':
       pts = [(439, 549), (468, 768), (229, 613), (283, 636), (425,
571), (407, 710), (402, 708)]
   if pts is not None:
       reply = "\n".join((str(x) + " " + str(y) for x, y in pts)) +
"\n"

   return reply

def check(reply, clue):
   clue_arr = eval(clue)
   stat = False
   if reply == "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]":
       stat = True
   else:
       split_reply = reply.splitlines()
       parsed_reply = []
       for string in split_reply:
           num = string.split()
           parsed_reply.append([int(num[0]), int(num[1])])
       if sorted(parsed_reply) == sorted(clue_arr):
           stat = True
   return stat
```

Задача 2.6 (1 балл)

Условие:

Этот шаг содержит тесты с 16 по 20 (базовый вариант). Используйте его, если ваша программа не проходит полный тест (шаг 1).

Решение:

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

using namespace std;

double findangle(int x1, int x2, int y1, int y2){ //Поиск угла
    double angle=atan2(y2-y1, x2-x1);
    return angle;
}
```

```
void rotate(int x1, int y1, double &x2, double &y2, double angle){
//Поворот отрезка вокруг начала
    double dx=x2-x1;
    double dy=y2-y1;
    x2=x1+dx*cos(angle)-dy*sin(angle);
    y2=y1+dx*sin(angle)+dy*cos(angle);
}

int main()
{
    //Reading == ready
    unsigned short n, m;
    cin >> n;

    int p[2][n], output[2][n], curr[2][n];
    for (int i=0; i<n; i++)
        cin >> p[0][i] >> p[1][i]; //точки паттерна
    cin >> m;

    int c[2][m];

    for (int i=0; i<m; i++)
        cin >> c[0][i] >> c[1][i];//распознанные точки

    double sbest=10000000;
    const double alpha0=findangle(p[0][0], p[0][1], p[1][0], p[1][1]);
//угол между первым отрезком паттерна и OX
    const double length0=sqrt( (p[0][0]-p[0][1])*(p[0][0]-
p[0][1])+(p[1][0]-p[1][1])*(p[1][0]-p[1][1]) );//Длина первого отрезка
паттерна

    for (int i=0; i<m; i++)
    {
        int counter=1; //Искомая точка паттерна
        curr[0][0]=c[0][i];
        curr[1][0]=c[1][i];
        for (int j=0; j<m; j++)
        {
            if (j==i) continue;//Костыль
            set <int> used;//Множество распознанных точек
            used.insert(i);
            used.insert(j);
            curr[0][1]=c[0][j];
            curr[1][1]=c[1][j];
            double length=sqrt( (c[0][i]-c[0][j])*(c[0][i]-
c[0][j])+(c[1][i]-c[1][j])*(c[1][i]-c[1][j]) );
            double alpha=findangle(c[0][i], c[0][j], c[1][i],
c[1][j])-alpha0;// Угол поворота
            double k=length/length0;//Коэффициент масштабирования
            counter=2;
            int jcurr=j;//Проверяемая точка
            double sdelta=0;//Сумма отклонений от предсказания
            while (counter<n){
                //(Предсказание следующей точки
                int dx=p[0][counter]-p[0][counter-1];
                int dy=p[1][counter]-p[1][counter-1];
```

```
                double xnew= (double)
c[0][jcurr]+k*sqrt(dx*dx+dy*dy)*cos(atan2(dy, dx));
                double ynew= (double)
c[1][jcurr]+k*sqrt(dx*dx+dy*dy)*sin(atan2(dy, dx));

                rotate(c[0][jcurr], c[1][jcurr], xnew, ynew, alpha);

                //Конец предсказания)

                int coord=0; //Текущая точка
                double dbest=10000000; // Минимальное отклонение
                int prev=-1; //Предыдущая подходящая точка

                while (coord<m){
                    double delta=sqrt((c[0][coord]-xnew)*(c[0][coord]-
xnew)+(c[1][coord]-ynew)*(c[1][coord]-ynew)); // Расчёт отклонения
                    if (delta<dbest && !used.count(coord)){
//Нахождение минимального отклонения с проверкой, не занята ли уже
точка.
                        jcurr=coord;
                        dbest=delta;
                        used.insert(jcurr);
                        used.erase(prev); // Удаление предыдущей точки
при нахождении более подходящей
                        prev=jcurr;
                    }
                    coord++;
                }
                sdelta+=dbest;
                curr[0][counter]=c[0][jcurr];
                curr[1][counter]=c[1][jcurr];
                counter++;
            }
            if (sdelta<sbest){
                sbest=sdelta;
                for (int o=0; o<n; o++){
                    output[0][o]=curr[0][o];
                    output[1][o]=curr[1][o];
                }
            }
        }
    }

    for (int i=0; i<n; i++)
        cout << output[0][i] << ' ' << output[1][i] << endl;
    return 0;
}
```

Критерии оценки:

```
import math
import random
import time


def generate():
   return []
```

```
def solve(dataset):
   reply = "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]"
   pts = None
   if dataset == '5\n0 0\n10 0\n20 0\n30 0\n40 0\n10\n1220 583\n533
230\n1485 327\n852 849\n1359 453\n1734 923\n1283 521\n1422 394\n1246
1027\n625 544\n':
       pts = [(1485, 327), (1422, 394), (1359, 453), (1283, 521),
(1220, 583)]
   if dataset == '7\n30 0\n20 0\n10 0\n0 0\n0 10\n0 20\n0 30\n15\n1113
908\n159 934\n1135 873\n777 346\n1267 506\n1031 942\n1664 144\n454
1006\n967 896\n1738 439\n999 921\n1917 818\n496 923\n1086 934\n1063
969\n':
       pts = [(967, 896), (999, 921), (1031, 942), (1063, 969), (1086,
934), (1113, 908), (1135, 873)]
   if dataset == '9\n40 20\n20 40\n0 20\n20 0\n20 20\n30 30\n10 30\n30
10\n10 10\n20\n1037 831\n1793 696\n1242 519\n1655 675\n1299 870\n815
649\n1747 915\n1034 404\n846 839\n907 530\n1337 291\n189 618\n1018
629\n1005 978\n947 941\n1605 379\n727 769\n931 739\n1679 51\n1118
701\n':
       pts = [(1118, 701), (947, 941), (727, 769), (907, 530), (931,
739), (1037, 831), (846, 839), (1018, 629), (815, 649)]
   if dataset == '5\n32 43\n22 35\n13 29\n27 1\n15 16\n25\n514
142\n1666 571\n261 182\n1036 164\n29 161\n1327 1068\n1181 417\n668
729\n531 395\n1420 671\n486 729\n380 478\n1016 753\n1020 924\n1808
520\n440 178\n900 642\n1125 74\n1653 730\n548 286\n868 1056\n846
913\n366 830\n1523 55\n916 720\n':
       pts = [(29, 161), (261, 182), (440, 178), (486, 729), (531,
395)]
   if dataset == '7\n13 49\n16 13\n48 47\n41 41\n14 46\n22 24\n23
25\n23\n966 144\n229 613\n1139 834\n243 1037\n1106 172\n706 973\n407
710\n1078 39\n267 28\n146 71\n79 509\n283 636\n622 122\n468 768\n856
699\n200 827\n425 571\n439 549\n1097 386\n1189 978\n239 772\n402
708\n291 430\n':
       pts = [(439, 549), (468, 768), (229, 613), (283, 636), (425,
571), (407, 710), (402, 708)]
   if pts is not None:
       reply = "\n".join((str(x) + " " + str(y) for x, y in pts)) +
"\n"

   return reply

def check(reply, clue):
   clue_arr = eval(clue)
   stat = False
   if reply == "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]":
       stat = True
   else:
       split_reply = reply.splitlines()
       parsed_reply = []
       for string in split_reply:
           num = string.split()
           parsed_reply.append([int(num[0]), int(num[1])])
       if sorted(parsed_reply) == sorted(clue_arr):
           stat = True
   return stat
```

Задача 2.7 (1 балл)

Условие:

Этот шаг содержит тесты с 21 по 25 (продвинутый вариант). Используйте его, если ваша программа не проходит полный тест (шаг 1).

Решение:

```
#include <functional>
#include <iostream>
#include <vector>
#include <unordered_map>
#include <unordered_set>
#include <set>
#include <stdexcept>
#include <limits>
#include <math.h>
#include <stdint.h>

using Int   =  int32_t;
using Index = uint32_t;
using Cell  = uint64_t;
using Model = uint64_t;

template <class A, class B> using    Pair = std::pair<A,B>;
template <class K, class V> using     Map = std::unordered_map<K,V>;
template <class T>          using GridMap = std::unordered_map<Cell,
T>;
template <class T>          using     Set = std::unordered_set<T>;
template <class T>          using     Vec = std::vector<T>;

void log(std::string s) {
 std::cout << s << std::endl;
}


Model to_model(Index from, Index to, Index point) {
 Model mf = from  & 0xfffff;
 Model mt = to    & 0xfffff;
 Model mp = point & 0xfffff;

 Model result = (mf << 40) | (mt << 20) | mp;
 return result;
}

Index model_from (Model m) {return  m >> 40;}
Index model_to   (Model m) {return (m >> 20) & 0xfffff;}
Index model_point(Model m) {return  m        & 0xfffff;}

Cell components_to_cell(Int cx, Int cy) {
 Cell result;

 Index ux = cx;
 Index uy = cy;

 result = uy;
 result <<= 32;
 result &= ux;
```

```
 return result;
}

Cell point_to_cell(double x, double y, double eps) {
 Int cx = floor(x/eps);
 Int cy = floor(y/eps);

 return components_to_cell(cx, cy);
}

template <class T>
struct Grid {
 double          eps;
 GridMap<Vec<T>> map;

 Grid(double eps) : eps(eps) {}

 void add_point(double x, double y, T index) {
   map[point_to_cell(x,y,eps)].insert(index);
 }

 void add_ball(double x, double y, double r, T index) {
   Int cx_min = floor((x-r)/eps);
   Int cy_min = floor((y-r)/eps);
   Int cx_max = floor((x+r)/eps);
   Int cy_max = floor((y+r)/eps);

   double d_bound  = r + eps * 0.87;
   double dd_bound = d_bound * d_bound;
   for (Int cx = cx_min; cx <= cx_max; cx++) {
     for (Int cy = cy_min; cy <= cy_max; cy++) {
       double x_min = cx*eps;
       double y_min = cy*eps;

       double x_center = x_min + eps/2;
       double y_center = y_min + eps/2;

       double dx = x_center - x;
       double dy = y_center - y;


       if (dx*dx + dy*dy > dd_bound) {break;}

       map[components_to_cell(cx,cy)].push_back(index);
     }
   }
 }
};


struct Point {
 double x;
 double y;
};

using Indices  = std::vector<Index>;
using Points   = std::vector<Point>;
```

```
using Distance = double;
using ItoIDMap = std::unordered_map<Index,std::pair<Index,Distance>>;
using DDSet    = std::set<std::pair<double,double>>;

void foreach_transformed_point(Point &start, Point &end, Points
&points, std::function<void(double,double,Index)> f) {
 double dx = end.x - start.x;
 double dy = end.y - start.y;

 double dd = dx*dx + dy*dy;

 for (Index i = 0; i != points.size(); i++) {
   auto &p = points[i];

   double x = p.x - start.x;
   double y = p.y - start.y;

   double tx = (dx*x + dy*y)/dd;
   double ty = (dx*y - dy*x)/dd;

   f(tx,ty,i);
 }
}



double squared_distance(Point &a, Point &b) {
 double dx = b.x - a.x;
 double dy = b.y - a.y;

 return dx*dx + dy*dy;
}




Grid<Pair<Model,Point>> pattern_to_grid(Points &points, double
transformed_radius, double ratio) {
 Grid<Pair<Model,Point>> result(transformed_radius*ratio);

 for (Index i = 0; i != points.size(); i++) {
   for (Index j = 0; j != points.size(); j++) {
     if (j == i) {break;}

     Point start = points[i];
     Point end   = points[j];

     foreach_transformed_point(start, end, points, [&](double bx,
double by, Index k){
       Point p;
       p.x = bx;
       p.y = by;
       result.add_ball(bx, by, transformed_radius,
{to_model(i,j,k),p});
     });
   }
 }
```

```
 return result;
}


Indices try_image(Point &start, Point &end, Points &image, size_t
pattern_size, Grid<Pair<Model,Point>> &grid, double tolerance, double
&norm) {
 double tt = tolerance*tolerance;

 Map<Model,ItoIDMap>   closest_points;
 Map<Model,Set<Index>> used_points;
 Map<Model,Distance>   max_distance;

 foreach_transformed_point(start, end, image, [&](double tx, double
ty, Index i){
   Cell cell = point_to_cell(tx, ty, grid.eps);
   Distance mult = squared_distance(start, end);

   if (grid.map.count(cell)) {
     for (auto mp : grid.map[cell]) {
       Model m = mp.first;
       Index j = model_point(m);

       Model truncated_m = to_model(model_from(m), model_to(m), 0);
       if (used_points[truncated_m].count(i)) {continue;}

       double dx = tx - mp.second.x;
       double dy = ty - mp.second.y;
       Distance new_distance = dx*dx + dy*dy;

       if (new_distance > tt) {continue;}
       new_distance *= mult;

       if (closest_points[truncated_m].count(j)) {
         Index    old_index    = closest_points[truncated_m][j].first;
         Distance old_distance =
closest_points[truncated_m][j].second;

         if (new_distance < old_distance) {
           used_points[truncated_m].erase(old_index);
           closest_points[truncated_m][j] = {i, new_distance};
           used_points[truncated_m].insert(i);
           continue;
         }
       }
       else {
         closest_points[truncated_m][j] = {i, new_distance};
         used_points[truncated_m].insert(i);
         continue;
       }
     }
   }
 });


 for (auto mu : used_points) {
```

```
    if (mu.second.size() == pattern_size) {
      Distance max_d = 0;

      for (auto i_id : closest_points[mu.first]) {
        if (i_id.second.second > max_d) {max_d = i_id.second.second;}
      }

      max_distance[mu.first] = max_d;
    }
  }


  Model    minmax_m = 0;
  Distance minmax_d = std::numeric_limits<Distance>::infinity();


  Indices result;

  if (max_distance.size() == 0) {return result;}


  for (auto m_md : max_distance) {

    if (minmax_d > m_md.second) {
      minmax_m = m_md.first;
      minmax_d = m_md.second;
    }
  }

  norm = minmax_d;

  for (auto i_id : closest_points[minmax_m]) {
    result.push_back(i_id.second.first);
  }

  return result;
}




void print_match(Points &pattern, Points &image) {
 double t_radius  = 0.05; // РЎ РРўР□Рњ РњРћР–РќРћ РџРћР□Р“Р
РђРўР¬РЎРЇ!
 double kappa = 1.145;
 double norm = 0;

 int iterations=0;
 while (true) {
   auto grid = pattern_to_grid(pattern, t_radius, kappa);
   bool found = false;

   for (Index i = 0; i != image.size(); i++) {
     for (Index j = 0; j != image.size(); j++) {
       if (i == j) {continue;}
```

```
       Indices a_try = try_image(image[i], image[j], image,
pattern.size(), grid, t_radius, norm);

       if (a_try.size() != 0) {

         for (auto index : a_try) {
           Point p = image[index];

           std::cout << p.x << ' ' << p.y << std::endl;
         }
         return;
       }
     }
   }
   t_radius *= 2;
 }
}



Points read_array() {
 Index N;
 std::cin >> N;


 DDSet set;

 for (Index i = 0; i != N; i++) {
   double x;
   double y;

   std::cin >> x >> y;
   set.insert({x,y});
 }

 Points result;

 for (auto &p : set) {result.push_back({p.first, p.second});}

 return result;
}

int main() {
 Points pattern = read_array();
 Points image   = read_array();

 print_match(pattern, image);
}
```

Критерии оценки:

```
import math
import random
import time


def generate():
   return []
```

```
def solve(dataset):
   reply = "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]"
   pts = None
   if dataset == '5\n0 0\n10 0\n20 0\n30 0\n40 0\n10\n1220 583\n533
230\n1485 327\n852 849\n1359 453\n1734 923\n1283 521\n1422 394\n1246
1027\n625 544\n':
       pts = [(1485, 327), (1422, 394), (1359, 453), (1283, 521),
(1220, 583)]
   if dataset == '7\n30 0\n20 0\n10 0\n0 0\n0 10\n0 20\n0 30\n15\n1113
908\n159 934\n1135 873\n777 346\n1267 506\n1031 942\n1664 144\n454
1006\n967 896\n1738 439\n999 921\n1917 818\n496 923\n1086 934\n1063
969\n':
       pts = [(967, 896), (999, 921), (1031, 942), (1063, 969), (1086,
934), (1113, 908), (1135, 873)]
   if dataset == '9\n40 20\n20 40\n0 20\n20 0\n20 20\n30 30\n10 30\n30
10\n10 10\n20\n1037 831\n1793 696\n1242 519\n1655 675\n1299 870\n815
649\n1747 915\n1034 404\n846 839\n907 530\n1337 291\n189 618\n1018
629\n1005 978\n947 941\n1605 379\n727 769\n931 739\n1679 51\n1118
701\n':
       pts = [(1118, 701), (947, 941), (727, 769), (907, 530), (931,
739), (1037, 831), (846, 839), (1018, 629), (815, 649)]
   if dataset == '5\n32 43\n22 35\n13 29\n27 1\n15 16\n25\n514
142\n1666 571\n261 182\n1036 164\n29 161\n1327 1068\n1181 417\n668
729\n531 395\n1420 671\n486 729\n380 478\n1016 753\n1020 924\n1808
520\n440 178\n900 642\n1125 74\n1653 730\n548 286\n868 1056\n846
913\n366 830\n1523 55\n916 720\n':
       pts = [(29, 161), (261, 182), (440, 178), (486, 729), (531,
395)]
   if dataset == '7\n13 49\n16 13\n48 47\n41 41\n14 46\n22 24\n23
25\n23\n966 144\n229 613\n1139 834\n243 1037\n1106 172\n706 973\n407
710\n1078 39\n267 28\n146 71\n79 509\n283 636\n622 122\n468 768\n856
699\n200 827\n425 571\n439 549\n1097 386\n1189 978\n239 772\n402
708\n291 430\n':
       pts = [(439, 549), (468, 768), (229, 613), (283, 636), (425,
571), (407, 710), (402, 708)]
   if pts is not None:
       reply = "\n".join((str(x) + " " + str(y) for x, y in pts)) +
"\n"

   return reply

def check(reply, clue):
   clue_arr = eval(clue)
   stat = False
   if reply == "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]":
       stat = True
   else:
       split_reply = reply.splitlines()
       parsed_reply = []
       for string in split_reply:
           num = string.split()
           parsed_reply.append([int(num[0]), int(num[1])])
       if sorted(parsed_reply) == sorted(clue_arr):
           stat = True
   return stat
```

Задача 2.8 (1 балл)

Условие:

Этот шаг содержит тесты с 26 по 30 (продвинутый вариант). Используйте его, если ваша программа не проходит полный тест (шаг 1).

Решение:

```
#include <ctime>
#include <cmath>
#include <cstdlib>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
template <class _T>
struct point
{
    _T x, y;
   point(){};
   template<class _T1, class _T2>
   point(_T1 _x, _T2 _y) : x(_x), y(_y) {};
   template <class _T1>
   bool operator==(const point<_T1>& a) const
   {
       return a.x == x && a.y == y;
   }
};
template <class _T1, class _T2>
inline bool cmp_x(const point<_T1>& a, const point<_T2>& b) { return
a.x != b.x ? a.x < b.x : a.y < b.y; }
template <class _T1, class _T2>
inline bool cmp_y(const point<_T1>& a, const point<_T2>& b) { return
a.y != b.y ? a.y < b.y : a.x < b.x; }

vector<point<int> > get_points()
{
   int n; cin >> n;
   vector<point<int> > res(n);
   for (auto& p : res)
       cin >> p.x >> p.y;
   return res;
}
template <class _T1>
long double abs(const point<_T1>& a) { return sqrtl(a.x*a.x +
a.y*a.y); }
template <class _T1, class _T2>
inline point<long double> add(const point<_T1>& a, const point<_T2>&
b) { return point<long double>(a.x+b.x, a.y+b.y); }
template <class _T1, class _T2>
inline point<long double> sub(const point<_T1>& a, const point<_T2>&
b) { return point<long double>(a.x-b.x, a.y-b.y); }
template <class _T1, class _T2>
inline point<long double> mul(const point<_T1>& a, const point<_T2>&
b) { return point<long double>(a.x*b.x - a.y*b.y, a.x*b.y + a.y*b.x);
}
template <class _T1, class _T2>
```

```
inline point<long double> divide(const point<_T1>& a, const
point<_T2>& b)
{
   long double d = b.x*b.x + b.y*b.y;
   return point<long double>((a.x*b.x + a.y*b.y)/d, (a.y*b.x -
a.x*b.y)/d);
}

struct pattern
{
   const vector<point<int> > pattern_base;
   vector<point<long double> > pattern_points;

   pattern(const vector<point<int> >& base) : pattern_base(base) {};

   void move_to(point<int> a, point<int> b)
   {
       point<int> a0 = pattern_base[0];
       point<int> b0 = pattern_base[1];

       /// in complex (b0 - a0)*k = (b - a)
       ///            k = (b - a)/(b0 - a0)
       point<long double> k = divide(sub(b, a), sub(b0, a0));
       pattern_points.clear();
       pattern_points.reserve(pattern_base.size());
       for (auto p : pattern_base)
           pattern_points.push_back(point<long double>(p.x, p.y));
       point<long double> z0 = sub(a, mul(a0, k));
       for (auto& p : pattern_points)
           p = add(mul(p, k), z0);
   }
};

template <class _T>
using iter = typename vector<point<_T> >::iterator;
template <class _T>
void build_2d_tree(iter<_T> begin, iter<_T> end, bool y_split = false)
{
   if (end - begin < 2) return;
   iter<_T> middle = begin + (end - begin)/2;
   nth_element(begin, middle, end, y_split ? cmp_y<int, int> :
cmp_x<int, int>);
   build_2d_tree<int>(begin, middle, !y_split);
   build_2d_tree<int>(middle+1, end, !y_split);
}

template <class _T1, class _T2>
long double get_dist(iter<_T1> begin, iter<_T1> end, bool y_split,
                     const point<_T2>& t, long double& dist,
point<_T1>& ans)
{
   if (end - begin < 1) return HUGE_VALL;
   iter<_T1> middle = begin + (end - begin)/2;
   long double dx = (t.x - middle->x)*(t.x - middle->x),
               dy = (t.y - middle->y)*(t.y - middle->y),
               d3 = dx+dy;

   if (d3 < dist) dist = d3, ans = *middle;
```

```
   if (end - begin == 1) return d3;

   if ((y_split ? cmp_y<long double, int> : cmp_x<long double,
int>)(t, *middle))
   {
       if ((y_split ? dy : dx) < get_dist(begin, middle, !y_split, t,
dist, ans))
                                 get_dist(middle+1, end, !y_split, t,
dist, ans);
   }
   else
   {
       if ((y_split ? dy : dx) < get_dist(middle+1, end, !y_split, t,
dist, ans))
                                 get_dist(begin, middle, !y_split, t,
dist, ans);
   }

   return dist;
}

template <class _T1, class _T2>
inline
point<_T1> get_nearest_point(vector<point<_T1> >& base, point<_T2> p)
{
   long double dist = HUGE_VALL;
   point<_T1> ans;
   get_dist(base.begin(), base.end(), false,
                              p, dist, ans);
   return ans;
}

template <class _T1, class _T2>
inline
long double get_dist(vector<point<_T1> >& base, point<_T2> p)
{
   long double dist = HUGE_VALL;
   point<_T1> ans;
   return get_dist(base.begin(), base.end(), false,
                   p, dist, ans);
}

struct screen
{
   const vector<point<int> > screen_base;
   vector<point<int> > tree;

   inline
   point<int> get_nearest(point<long double> p)
   {
       return get_nearest_point(tree, p);
   }

   inline
   long double get_dist_to_nearest(point<long double> p)
   {
       return get_dist(tree, p);
```

```
    }

    screen(const vector<point<int> >& base) : screen_base(base)
    {
        tree = screen_base;
        build_2d_tree<int>(tree.begin(), tree.end());
    };

    vector<point<int> > get_score(const pattern& ptrn, long double&
sum)
    {
        sum = 0;
        vector<point<int> > res;
        res.reserve(ptrn.pattern_points.size());
        for (auto p : ptrn.pattern_points)
            res.push_back(this->get_nearest(p)),
            sum += abs(sub(res.back(), p));
        sort(res.begin(), res.end(), [](const point<int>& a, const
point<int>& b){return a.x != b.x ? a.x < b.x : a.y < b.y; });
        if (unique(res.begin(), res.end()) != res.end())
            sum = HUGE_VALL,
            res.clear();
        return res;
    }
};

pair<long double, vector<point<int> > > get_solve(pattern ptrn,
screen& scr)
{
    int n = scr.screen_base.size();

    long double best_score = HUGE_VALL;
    int besti = 0;
    int bestj = 1;

    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            if (i != j)
            {
                ptrn.move_to(scr.screen_base[i], scr.screen_base[j]);
                long double score;
                scr.get_score(ptrn, score);
                if (score < best_score)
                    best_score = score,
                    besti = i,
                    bestj = j;
            }

    vector<point<int> > ans;
    ans.reserve(ptrn.pattern_base.size());

    ptrn.move_to(scr.screen_base[besti], scr.screen_base[bestj]);
    for (auto p : ptrn.pattern_points)
        ans.push_back(scr.get_nearest(p));

    return {best_score, ans};
}
```

```
int main()
{
   ios_base::sync_with_stdio(false);
   cin.tie(NULL);
   srand(time(NULL));

   vector<point<int> > pattern_points = get_points();
   screen scr(get_points());

   long double best_score = HUGE_VALL;
   vector<point<int> > best_ans;

   int n = pattern_points.size();
   int m = scr.screen_base.size();
   int T = min(150, (int)floorl(25e7/(m*m*m*n)));

   for (int t = 0; t < T; ++t)
   {
       pair<long double, vector<point<int> > > it =
get_solve(pattern(pattern_points), scr);
       if (it.first < best_score)
           best_score = it.first,
           best_ans = it.second;
       random_shuffle(pattern_points.begin(), pattern_points.end());
   } for (auto p : best_ans)
       cout << p.x << ' ' << p.y << '\n';

   return 0;
}
```

Критерии:

```
import math
import random
import time


def generate():
   return []

def solve(dataset):
   reply = "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]"
   pts = None
   if dataset == '5\n0 0\n10 0\n20 0\n30 0\n40 0\n10\n1220 583\n533
230\n1485 327\n852 849\n1359 453\n1734 923\n1283 521\n1422 394\n1246
1027\n625 544\n':
       pts = [(1485, 327), (1422, 394), (1359, 453), (1283, 521),
(1220, 583)]
   if dataset == '7\n30 0\n20 0\n10 0\n0 0\n0 10\n0 20\n0 30\n15\n1113
908\n159 934\n1135 873\n777 346\n1267 506\n1031 942\n1664 144\n454
1006\n967 896\n1738 439\n999 921\n1917 818\n496 923\n1086 934\n1063
969\n':
       pts = [(967, 896), (999, 921), (1031, 942), (1063, 969), (1086,
934), (1113, 908), (1135, 873)]
   if dataset == '9\n40 20\n20 40\n0 20\n20 0\n20 20\n30 30\n10 30\n30
10\n10 10\n20\n1037 831\n1793 696\n1242 519\n1655 675\n1299 870\n815
```

```
649\n1747 915\n1034 404\n846 839\n907 530\n1337 291\n189 618\n1018
629\n1005 978\n947 941\n1605 379\n727 769\n931 739\n1679 51\n1118
701\n':
       pts = [(1118, 701), (947, 941), (727, 769), (907, 530), (931,
739), (1037, 831), (846, 839), (1018, 629), (815, 649)]
   if dataset == '5\n32 43\n22 35\n13 29\n27 1\n15 16\n25\n514
142\n1666 571\n261 182\n1036 164\n29 161\n1327 1068\n1181 417\n668
729\n531 395\n1420 671\n486 729\n380 478\n1016 753\n1020 924\n1808
520\n440 178\n900 642\n1125 74\n1653 730\n548 286\n868 1056\n846
913\n366 830\n1523 55\n916 720\n':
       pts = [(29, 161), (261, 182), (440, 178), (486, 729), (531,
395)]
   if dataset == '7\n13 49\n16 13\n48 47\n41 41\n14 46\n22 24\n23
25\n23\n966 144\n229 613\n1139 834\n243 1037\n1106 172\n706 973\n407
710\n1078 39\n267 28\n146 71\n79 509\n283 636\n622 122\n468 768\n856
699\n200 827\n425 571\n439 549\n1097 386\n1189 978\n239 772\n402
708\n291 430\n':
       pts = [(439, 549), (468, 768), (229, 613), (283, 636), (425,
571), (407, 710), (402, 708)]
   if pts is not None:
       reply = "\n".join((str(x) + " " + str(y) for x, y in pts)) +
"\n"

   return reply

def check(reply, clue):
   clue_arr = eval(clue)
   stat = False
   if reply == "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]":
       stat = True
   else:
       split_reply = reply.splitlines()
       parsed_reply = []
       for string in split_reply:
           num = string.split()
           parsed_reply.append([int(num[0]), int(num[1])])
       if sorted(parsed_reply) == sorted(clue_arr):
           stat = True
   return stat
```

Задача 2.9 (1 балл)

Условие: Этот шаг содержит тесты с 31 по 35 (продвинутый вариант). Используйте его, если ваша программа не проходит полный тест (шаг 1).

Решение:

```
import math
detectedPoints = [] #лист с найденными точками
n_pattern=int(input()) #ввод количества точек в паттерне

listPattern=[] #лист с координатами всех точек паттерна
for i in range(n_pattern): #ввод и заполнение точек паттерна
 a_t_1=input().split(' ',maxsplit = 1) #ввод координат одной точки,
конверт строки в временный лист
```

```
 listPattern.append([int(a_t_1[0]),int(a_t_1[1])]) #добавление точки в
лист

n_img=int(input()) #количество распознанных точек в фото
listImg=[] #лист с координатами всех точек на изображении
for i in range (n_img):
 a_t_1=input().split(' ',maxsplit = 1) #ввод координат одной точки,
конверт строки в временный лист
 listImg.append([int(a_t_1[0]),int(a_t_1[1])]) #добавление точки в
лист
if len(listImg) == 200:
   print(listImg-1)
dmax=0 # Насстояние между двумя самыми отдаленными точками в паттерне
(для уменьшения погрешности)
dm=0 # Среднее расстояние между точками в паттерне
dmin=2000 # Минимальное расстояние между точками в паттерне
pm=[0,0] #лист с индексами максимально удаленных точек (из листа
listPatten)
for i in range(n_pattern): # Вычисляем максимальное минимальное и
среднее растояния
       for j in range(i+1,n_pattern):
               d0=math.sqrt((listPattern[i][0]-
listPattern[j][0])**2+(listPattern[i][1]-listPattern[j][1])**2)
#находим промежуточное расстояние между точками listPattern[i] и
<..>[j]
               dm+=d0
               if d0<dmin:
                       dmin=d0
               if d0>dmax:
                       dmax=d0
                       pm[0]=i
                       pm[1]=j

dm=dm*2/(n_pattern*(n_pattern-1))

#для удобства в паттерне сдвигается начало координат в нулевую точку
(listPattern[0])
XshiftPattern = listPattern[pm[0]][0]
YshiftPattern = listPattern[pm[0]][1]
#координаты 0-й точки
# Перемещение всех координат в listPattern на XshiftPattern и
YshiftPattern влево и вниз
for i in range (len(listPattern)):
 listPattern[i][0]=listPattern[i][0]-XshiftPattern
 listPattern[i][1]=listPattern[i][1]-YshiftPattern
anglePattern = math.atan2(listPattern[pm[1]][1],listPattern[pm[1]][0])
#угол между горизонталью и прямой, образованной 0-й и 1-й точками в
паттерне
dPattern = dmax #расстояние между теми же точками
det=False
isEnd = False #две переменные нужные для остановки цикла после
нахождения всех точек
nFound=[] #лист с найденными точками в один "оборот" цикла, т. е.
каждый новый цикл он чистится, чтобы исключить возможные повторы среди
найденных точек
sol=[]
d_sol=[]
n_sol=0
```

```
all_sol=[]
all_d=[]
if_war=[]
p_war=[]
win_war=[]
Kc=1
trig=False
dK=10
while isEnd==False:
       if trig==True:
              isEnd=True
       for i in range (n_img):
              p0=listImg[i] #считаем, что некая i-я точка из
множества найденных точек соответствует 0-й точке из паттерна
              for j in range (n_img):
                     war=False
                     if i!=j: #and: isEnd==False:
                            p1=listImg[j] #считаем, что некая j-я
точка из множества найденных точек соответствует 1-й точке из паттерна
                            angle_b=math.atan2((p1[1]-
p0[1]),(p1[0]-p0[0])) #угол между прямой p0p1 и горизонталью
                            delta_angle = angle_b-anglePattern
#угол поворота
                            dImg = math.sqrt((p0[0]-
p1[0])**2+(p0[1]-p1[1])**2) #расстояние между p0 и p1
                            R = dImg/dPattern #масштаб
                            dK=Kc*4*R
                            det = True
                            delta=0
                            detectedPoints.clear()
                            nFound.clear() #чистка листа найденных
точек за один цикл
                            d_sol.clear()
                            if_war.clear()
                            for k in range(n_pattern):
                                   if det==True:
                                          xP = round(p0[0] +
R*(listPattern[k][0]*math.cos(delta_angle)-
listPattern[k][1]*math.sin(delta_angle)))
                                          yP = round(p0[1] +
R*(listPattern[k][1]*math.cos(delta_angle)+listPattern[k][0]*math.sin(
delta_angle))) #преобразование координат, нахождение  примерного
расположения следующей точки
                                          ddd=dK
                                          dd=dK #маскимальная
погрешность, константа, определенная из здравого смысла
                                          for kk in
range(0,n_img):
                                                 if 1==1: #kk!=i
and kk!=j:
                                                        dd0=mat
h.sqrt((xP-listImg[kk][0])**2+(yP-listImg[kk][1])**2) #расстояние
между (xP; yP) и listImg[kk]
                                                        if
dd0<ddd:

 ddd=dd0
```

```
 mmP=kk
                                                                    if
dd0<dd and kk not in nFound:
 dd=dd0
 mP=kk
                                                  if dd<dK:
                                                          detectedPoints.
append(listImg[mP])
                                                          d_sol.append(dd
)
                                                          nFound.append(m
P)
                                                          delta+=dd
                                                          if_war.append(F
alse)
                                                          if mP!=mmP:
                                                                  war=Tru
e
                                                                  if_war[
k]=True
                                                                  for l
in range(k):
 if nFound[l]==mmP:
         if_war[l]=True
                                                  else:
                                                          detectedPoints.
clear()
                                                          nFound.clear()
                                                          d_sol.clear()
                                                          det = False



                                    if len(detectedPoints)==n_pattern and
len(d_sol)==n_pattern:
                                                  n_sol+=1
                                                  d2=0
                                                  for ii in range(n_pattern):
                                                          all_sol.append(detected
Points[ii])
                                                          d2+=d_sol[ii]**2
                                                  delta=math.sqrt(d2/(n_pattern+1
))/R
                                                  all_d.append(delta)
                                                  detectedPoints.clear()
                                                  isEnd=True
                                                  trig=True
       Kc*=3
min_d=dK
min_sol=0
for i in range(n_sol):
       if all_d[i]<min_d:
```

```
                min_d=all_d[i]
                min_sol=i

for ii in range(n_pattern):
       print(all_sol[min_sol*n_pattern+ii][0],all_sol[min_sol*n_patter
n+ii][1])
```

Критерии:

```
import math
import random
import time


def generate():
   return []

def solve(dataset):
   reply = "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]"
   pts = None
   if dataset == '5\n0 0\n10 0\n20 0\n30 0\n40 0\n10\n1220 583\n533
230\n1485 327\n852 849\n1359 453\n1734 923\n1283 521\n1422 394\n1246
1027\n625 544\n':
       pts = [(1485, 327), (1422, 394), (1359, 453), (1283, 521),
(1220, 583)]
   if dataset == '7\n30 0\n20 0\n10 0\n0 0\n0 10\n0 20\n0 30\n15\n1113
908\n159 934\n1135 873\n777 346\n1267 506\n1031 942\n1664 144\n454
1006\n967 896\n1738 439\n999 921\n1917 818\n496 923\n1086 934\n1063
969\n':
       pts = [(967, 896), (999, 921), (1031, 942), (1063, 969), (1086,
934), (1113, 908), (1135, 873)]
   if dataset == '9\n40 20\n20 40\n0 20\n20 0\n20 20\n30 30\n10 30\n30
10\n10 10\n20\n1037 831\n1793 696\n1242 519\n1655 675\n1299 870\n815
649\n1747 915\n1034 404\n846 839\n907 530\n1337 291\n189 618\n1018
629\n1005 978\n947 941\n1605 379\n727 769\n931 739\n1679 51\n1118
701\n':
       pts = [(1118, 701), (947, 941), (727, 769), (907, 530), (931,
739), (1037, 831), (846, 839), (1018, 629), (815, 649)]
   if dataset == '5\n32 43\n22 35\n13 29\n27 1\n15 16\n25\n514
142\n1666 571\n261 182\n1036 164\n29 161\n1327 1068\n1181 417\n668
729\n531 395\n1420 671\n486 729\n380 478\n1016 753\n1020 924\n1808
520\n440 178\n900 642\n1125 74\n1653 730\n548 286\n868 1056\n846
913\n366 830\n1523 55\n916 720\n':
       pts = [(29, 161), (261, 182), (440, 178), (486, 729), (531,
395)]
   if dataset == '7\n13 49\n16 13\n48 47\n41 41\n14 46\n22 24\n23
25\n23\n966 144\n229 613\n1139 834\n243 1037\n1106 172\n706 973\n407
710\n1078 39\n267 28\n146 71\n79 509\n283 636\n622 122\n468 768\n856
699\n200 827\n425 571\n439 549\n1097 386\n1189 978\n239 772\n402
708\n291 430\n':
       pts = [(439, 549), (468, 768), (229, 613), (283, 636), (425,
571), (407, 710), (402, 708)]
   if pts is not None:
       reply = "\n".join((str(x) + " " + str(y) for x, y in pts)) +
"\n"
```

```
   return reply

def check(reply, clue):
   clue_arr = eval(clue)
   stat = False
   if reply == "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]":
       stat = True
   else:
       split_reply = reply.splitlines()
       parsed_reply = []
       for string in split_reply:
           num = string.split()
           parsed_reply.append([int(num[0]), int(num[1])])
       if sorted(parsed_reply) == sorted(clue_arr):
           stat = True
   return stat
```

Задача 2.10 (1 балл)

Условие:

Этот шаг содержит тесты с 26 по 35 (продвинутый вариант).
Время, отведенное вашей программе на выполнение, уменьшено в 6 раз по сравнению с шагом 1 (с 30 до 5 секунд на тест).
Коэффициенты, увеличивающие отводимое на исполнение время для "медленных" языков (таких, как python), не используются. Используйте такие языки, как C++ и эвристические алгоритмы для увеличения производительности, если программа не успевает вычислить решение.
Шаг оценивается независимо от шага 1.

Решение:

```
#include <ctime>
#include <cmath>
#include <cstdlib>
#include <vector>
#include <iostream>
#include <algorithm>

using namespace std;

template <class _T>
struct point
{
    _T x, y;
    point(){};
    template<class _T1, class _T2>
    point(_T1 _x, _T2 _y) : x(_x), y(_y) {};
    template <class _T1>
    bool operator==(const point<_T1>& a) const
    {
        return a.x == x && a.y == y;
    }
};
template <class _T1, class _T2>
```

```
inline bool cmp_x(const point<_T1>& a, const point<_T2>& b) { return
a.x != b.x ? a.x < b.x : a.y < b.y; }
template <class _T1, class _T2>
inline bool cmp_y(const point<_T1>& a, const point<_T2>& b) { return
a.y != b.y ? a.y < b.y : a.x < b.x; }

vector<point<int> > get_points()
{
    int n; cin >> n;
    vector<point<int> > res(n);
    for (auto& p : res)
        cin >> p.x >> p.y;
    return res;
}

template <class _T1>
long double abs(const point<_T1>& a) { return sqrtl(a.x*a.x +
a.y*a.y); }
template <class _T1, class _T2>
inline point<long double> add(const point<_T1>& a, const point<_T2>&
b) { return point<long double>(a.x+b.x, a.y+b.y); }
template <class _T1, class _T2>
inline point<long double> sub(const point<_T1>& a, const point<_T2>&
b) { return point<long double>(a.x-b.x, a.y-b.y); }
template <class _T1, class _T2>
inline point<long double> mul(const point<_T1>& a, const point<_T2>&
b) { return point<long double>(a.x*b.x - a.y*b.y, a.x*b.y + a.y*b.x);
}
template <class _T1, class _T2>
inline point<long double> divide(const point<_T1>& a, const
point<_T2>& b)
{
    long double d = b.x*b.x + b.y*b.y;
    return point<long double>((a.x*b.x + a.y*b.y)/d, (a.y*b.x -
a.x*b.y)/d);
}

struct pattern
{
    const vector<point<int> > pattern_base;
    vector<point<long double> > pattern_points;

    pattern(const vector<point<int> >& base) : pattern_base(base) {};

    void move_to(point<int> a, point<int> b)
    {
        point<int> a0 = pattern_base[0];
        point<int> b0 = pattern_base[1];

        /// in complex (b0 - a0)*k = (b - a)
        ///            k = (b - a)/(b0 - a0)
        point<long double> k = divide(sub(b, a), sub(b0, a0));
        pattern_points.clear();
        pattern_points.reserve(pattern_base.size());
        for (auto p : pattern_base)
            pattern_points.push_back(point<long double>(p.x, p.y));

        /// f(z) = k*z + z0
```

```
        /// p = (p - a0)*k + a
        /// p = p*k - a0*k + a
        /// z0 = a - a0*k
        point<long double> z0 = sub(a, mul(a0, k));
        for (auto& p : pattern_points)
            p = add(mul(p, k), z0);
    }
};

template <class _T>
using iter = typename vector<point<_T> >::iterator;
template <class _T>
void build_2d_tree(iter<_T> begin, iter<_T> end, bool y_split = false)
{
    if (end - begin < 2) return;
    iter<_T> middle = begin + (end - begin)/2;
    nth_element(begin, middle, end, y_split ? cmp_y<int, int> :
cmp_x<int, int>);
    build_2d_tree<int>(begin, middle, !y_split);
    build_2d_tree<int>(middle+1, end, !y_split);
}

template <class _T1, class _T2>
long double get_dist(iter<_T1> begin, iter<_T1> end, bool y_split,
                     const point<_T2>& t, long double& dist,
point<_T1>& ans)
{
    if (end - begin < 1) return HUGE_VALL;
    iter<_T1> middle = begin + (end - begin)/2;
    long double dx = (t.x - middle->x)*(t.x - middle->x),
                dy = (t.y - middle->y)*(t.y - middle->y),
                d3 = dx+dy;

    if (d3 < dist) dist = d3, ans = *middle;

    if (end - begin == 1) return d3;

    if ((y_split ? cmp_y<long double, int> : cmp_x<long double,
int>)(t, *middle))
    {
        if ((y_split ? dy : dx) < get_dist(begin, middle, !y_split, t,
dist, ans))
                                  get_dist(middle+1, end, !y_split, t,
dist, ans);
    }
    else
    {
        if ((y_split ? dy : dx) < get_dist(middle+1, end, !y_split, t,
dist, ans))
                                  get_dist(begin, middle, !y_split, t,
dist, ans);
    }

    return dist;
}

template <class _T1, class _T2>
inline
```

```
point<_T1> get_nearest_point(vector<point<_T1> >& base, point<_T2> p)
{
    long double dist = HUGE_VALL;
    point<_T1> ans;
    get_dist(base.begin(), base.end(), false,
                                p, dist, ans);
    return ans;
}

template <class _T1, class _T2>
inline
long double get_dist(vector<point<_T1> >& base, point<_T2> p)
{
    long double dist = HUGE_VALL;
    point<_T1> ans;
    return get_dist(base.begin(), base.end(), false,
                    p, dist, ans);
}

struct screen
{
    const vector<point<int> > screen_base;
    vector<point<int> > tree;

    inline
    point<int> get_nearest(point<long double> p)
    {
        return get_nearest_point(tree, p);
    }

    inline
    long double get_dist_to_nearest(point<long double> p)
    {
        return get_dist(tree, p);
    }

    screen(const vector<point<int> >& base) : screen_base(base)
    {
        tree = screen_base;
        build_2d_tree<int>(tree.begin(), tree.end());
    };

    vector<point<int> > get_score(const pattern& ptrn, long double&
sum)
    {
        sum = 0;
        vector<point<int> > res;
        res.reserve(ptrn.pattern_points.size());
        for (auto p : ptrn.pattern_points)
            res.push_back(this->get_nearest(p)),
            sum += abs(sub(res.back(), p));
        sort(res.begin(), res.end(), [](const point<int>& a, const
point<int>& b){return a.x != b.x ? a.x < b.x : a.y < b.y; });
        if (unique(res.begin(), res.end()) != res.end())
            sum = HUGE_VALL,
            res.clear();
        return res;
    }
```

```
};

pair<long double, vector<point<int> > > get_solve(pattern ptrn,
screen& scr)
{
    int n = scr.screen_base.size();

    long double best_score = HUGE_VALL;
    int besti = 0;
    int bestj = 1;

    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            if (i != j)
            {
                ptrn.move_to(scr.screen_base[i], scr.screen_base[j]);
                long double score;
                scr.get_score(ptrn, score);
                if (score < best_score)
                    best_score = score,
                    besti = i,
                    bestj = j;
            }

    vector<point<int> > ans;
    ans.reserve(ptrn.pattern_base.size());

    ptrn.move_to(scr.screen_base[besti], scr.screen_base[bestj]);
    for (auto p : ptrn.pattern_points)
        ans.push_back(scr.get_nearest(p));

    return {best_score, ans};
}

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    srand(time(NULL));

    vector<point<int> > pattern_points = get_points();
    screen scr(get_points());

    long double best_score = HUGE_VALL;
    vector<point<int> > best_ans;

    int n = pattern_points.size();
    int m = scr.screen_base.size();
    int T = min(150, (int)floorl(25e7/(m*m*m*n)));

    for (int t = 0; t < T; ++t)
    {
        pair<long double, vector<point<int> > > it =
get_solve(pattern(pattern_points), scr);
        if (it.first < best_score)
            best_score = it.first,
            best_ans = it.second;
        random_shuffle(pattern_points.begin(), pattern_points.end());
```

```
    }

    //cout << T << " " << best_score << '\n';
    for (auto p : best_ans)
        cout << p.x << ' ' << p.y << '\n';

    return 0;
}
```

Критерии оценки:

```
import math
import random
import time


def generate():
   return []

def solve(dataset):
   reply = "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]"
   pts = None
   if dataset == '5\n0 0\n10 0\n20 0\n30 0\n40 0\n10\n1220 583\n533
230\n1485 327\n852 849\n1359 453\n1734 923\n1283 521\n1422 394\n1246
1027\n625 544\n':
       pts = [(1485, 327), (1422, 394), (1359, 453), (1283, 521),
(1220, 583)]
   if dataset == '7\n30 0\n20 0\n10 0\n0 0\n0 0\n10\n0 20\n0 30\n15\n1113
908\n159 934\n1135 873\n777 346\n1267 506\n1031 942\n1664 144\n454
1006\n967 896\n1738 439\n999 921\n1917 818\n496 923\n1086 934\n1063
969\n':
       pts = [(967, 896), (999, 921), (1031, 942), (1063, 969), (1086,
934), (1113, 908), (1135, 873)]
   if dataset == '9\n40 20\n20 40\n0 20\n20 0\n20 20\n30 30\n10 30\n30
10\n10 10\n20\n1037 831\n1793 696\n1242 519\n1655 675\n1299 870\n815
649\n1747 915\n1034 404\n846 839\n907 530\n1337 291\n189 618\n1018
629\n1005 978\n947 941\n1605 379\n727 769\n931 739\n1679 51\n1118
701\n':
       pts = [(1118, 701), (947, 941), (727, 769), (907, 530), (931,
739), (1037, 831), (846, 839), (1018, 629), (815, 649)]
   if dataset == '5\n32 43\n22 35\n13 29\n27 1\n15 16\n25\n514
142\n1666 571\n261 182\n1036 164\n29 161\n1327 1068\n1181 417\n668
729\n531 395\n1420 671\n486 729\n380 478\n1016 753\n1020 924\n1808
520\n440 178\n900 642\n1125 74\n1653 730\n548 286\n868 1056\n846
913\n366 830\n1523 55\n916 720\n':
       pts = [(29, 161), (261, 182), (440, 178), (486, 729), (531,
395)]
   if dataset == '7\n13 49\n16 13\n48 47\n41 41\n14 46\n22 24\n23
25\n23\n966 144\n229 613\n1139 834\n243 1037\n1106 172\n706 973\n407
710\n1078 39\n267 28\n146 71\n79 509\n283 636\n622 122\n468 768\n856
699\n200 827\n425 571\n439 549\n1097 386\n1189 978\n239 772\n402
708\n291 430\n':
       pts = [(439, 549), (468, 768), (229, 613), (283, 636), (425,
571), (407, 710), (402, 708)]
   if pts is not None:
```

```
        reply = "\n".join((str(x) + " " + str(y) for x, y in pts)) +
"\n"

    return reply

def check(reply, clue):
    clue_arr = eval(clue)
    stat = False
    if reply == "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]":
        stat = True
    else:
        split_reply = reply.splitlines()
        parsed_reply = []
        for string in split_reply:
            num = string.split()
            parsed_reply.append([int(num[0]), int(num[1])])
        if sorted(parsed_reply) == sorted(clue_arr):
            stat = True
    return stat
```

Задача 2.11 (2 балла)

Условие:

Этот шаг содержит тесты с 26 по 35 (продвинутый вариант).

Время, отведенное вашей программе на выполнение, уменьшено в 30 раз по сравнению с шагом 1 (с 30 до 1 секунды на тест).

Коэффициенты, увеличивающие отводимое на исполнение время для "медленных" языков (таких, как python), не используются. Используйте такие языки, как C++ и эвристические алгоритмы для увеличения производительности, если программа не успевает вычислить решение.

Шаг оценивается независимо от шага 1.

Решение:

```
#include <iostream>
#include <math.h>
#include <set>

using namespace std;

float findangle(int x1, int x2, int y1, int y2){ //Поиск угла
    float angle=atan2(y2-y1, x2-x1);
    return angle;
}

void rotate(float x1, float y1, float &x2, float &y2, float angle){
//Поворот отрезка вокруг начала
    float dx=x2-x1;
    float dy=y2-y1;
    x2=x1+dx*cos(angle)-dy*sin(angle);
    y2=y1+dx*sin(angle)+dy*cos(angle);
}

int main()
```

```
{
    //Reading == ready
    unsigned short n, m;
    cin >> n;

    int p[2][n], output[2][n], curr[2][n];
    for (int i=0; i<n; i++)
        cin >> p[0][i] >> p[1][i]; //точки паттерна
    cin >> m;

    int c[2][m];

    for (int i=0; i<m; i++)
        cin >> c[0][i] >> c[1][i];//распознанные точки

    int sbest=10000000;
    const float alpha0=findangle(p[0][0], p[0][1], p[1][0], p[1][1]);
//угол между первым отрезком паттерна и OX
    const float length0=sqrt( (p[0][0]-p[0][1])*(p[0][0]-
p[0][1])+(p[1][0]-p[1][1])*(p[1][0]-p[1][1]) );//Длина первого отрезка
паттерна

    for (int i=0; i<m; i++)
    {
        int counter=1; //Искомая точка паттерна
        curr[0][0]=c[0][i];
        curr[1][0]=c[1][i];
        for (int j=0; j<m; j++)
        {
            if (j==i) j++;
            if (j==m) break; //Костыль
            set <int> used;//Множество распознанных точек
            used.insert(i);
            used.insert(j);
            curr[0][1]=c[0][j];
            curr[1][1]=c[1][j];
            float length=sqrt( (c[0][i]-c[0][j])*(c[0][i]-
c[0][j])+(c[1][i]-c[1][j])*(c[1][i]-c[1][j]) );
            float alpha=findangle(c[0][i], c[0][j], c[1][i], c[1][j])-
alpha0;// Угол поворота
            float k=length/length0;//Коэффициент масштабирования
            counter=2;
            int jcurr=j;//Проверяемая точка
            int sdelta=0;//Сумма отклонений от предсказания
            while (counter<n){
                //   (Предсказание следующей точки
                int dx=p[0][counter]-p[0][counter-1];
                int dy=p[1][counter]-p[1][counter-1];
                float currlength=k*sqrt(dx*dx+dy*dy);
                float xnew=(float)
c[0][jcurr]+currlength*cos(atan2(dy, dx));
                float ynew=(float)
c[1][jcurr]+currlength*sin(atan2(dy, dx));

                rotate(c[0][jcurr], c[1][jcurr], xnew, ynew, alpha);

                //Конец предсказания)
```

```
                int coord=0; //Текущая точка
                float dbest=1000000; // Минимальное отклонение
                int prev=0; //Предыдущая подходящая точка
                bool fl=0;

                while (coord<m){
                    float delta= fabs(c[0][coord]-xnew)+
fabs(c[1][coord]-ynew); // Расчёт отклонения
                    if (delta<dbest && !used.count(coord)){
//Нахождение минимального отклонения с проверкой, не занята ли уже
точка.
                        jcurr=coord;
                        dbest=delta;
                        used.insert(jcurr);
                        if (fl) used.erase(prev); // Удаление
предыдущей точки при нахождении более подходящей
                        prev=jcurr;
                        fl=1;
                    }
                    coord++;
                }
                sdelta+=dbest;
                curr[0][counter]=c[0][jcurr];
                curr[1][counter]=c[1][jcurr];
                counter++;
            }
            if (sdelta<sbest){
                sbest=sdelta;
                for (int o=0; o<n; o++){
                    output[0][o]=curr[0][o];
                    output[1][o]=curr[1][o];
                }
            }
        }
    }

    for (int i=0; i<n; i++)
        cout << output[0][i] << ' ' << output[1][i] << endl;
    return 0;
}
```

Критерии оценки:

```
import math
import random
import time


def generate():
   return []

def solve(dataset):
   reply = "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]"
   pts = None
```

```
   if dataset == '5\n0 0\n10 0\n20 0\n30 0\n40 0\n10\n1220 583\n533
230\n1485 327\n852 849\n1359 453\n1734 923\n1283 521\n1422 394\n1246
1027\n625 544\n':
       pts = [(1485, 327), (1422, 394), (1359, 453), (1283, 521),
(1220, 583)]
   if dataset == '7\n30 0\n20 0\n10 0\n0 0\n0 10\n0 20\n0 30\n15\n1113
908\n159 934\n1135 873\n777 346\n1267 506\n1031 942\n1664 144\n454
1006\n967 896\n1738 439\n999 921\n1917 818\n496 923\n1086 934\n1063
969\n':
       pts = [(967, 896), (999, 921), (1031, 942), (1063, 969), (1086,
934), (1113, 908), (1135, 873)]
   if dataset == '9\n40 20\n20 40\n0 20\n20 0\n20 20\n30 30\n10 30\n30
10\n10 10\n20\n1037 831\n1793 696\n1242 519\n1655 675\n1299 870\n815
649\n1747 915\n1034 404\n846 839\n907 530\n1337 291\n189 618\n1018
629\n1005 978\n947 941\n1605 379\n727 769\n931 739\n1679 51\n1118
701\n':
       pts = [(1118, 701), (947, 941), (727, 769), (907, 530), (931,
739), (1037, 831), (846, 839), (1018, 629), (815, 649)]
   if dataset == '5\n32 43\n22 35\n13 29\n27 1\n15 16\n25\n514
142\n1666 571\n261 182\n1036 164\n29 161\n1327 1068\n1181 417\n668
729\n531 395\n1420 671\n486 729\n380 478\n1016 753\n1020 924\n1808
520\n440 178\n900 642\n1125 74\n1653 730\n548 286\n868 1056\n846
913\n366 830\n1523 55\n916 720\n':
       pts = [(29, 161), (261, 182), (440, 178), (486, 729), (531,
395)]
   if dataset == '7\n13 49\n16 13\n48 47\n41 41\n14 46\n22 24\n23
25\n23\n966 144\n229 613\n1139 834\n243 1037\n1106 172\n706 973\n407
710\n1078 39\n267 28\n146 71\n79 509\n283 636\n622 122\n468 768\n856
699\n200 827\n425 571\n439 549\n1097 386\n1189 978\n239 772\n402
708\n291 430\n':
       pts = [(439, 549), (468, 768), (229, 613), (283, 636), (425,
571), (407, 710), (402, 708)]
   if pts is not None:
       reply = "\n".join((str(x) + " " + str(y) for x, y in pts)) +
"\n"

   return reply

def check(reply, clue):
   clue_arr = eval(clue)
   stat = False
   if reply == "[(1485, 327), (1422, 384), (1359, 453), (1283, 521),
(1220, 583)]":
       stat = True
   else:
       split_reply = reply.splitlines()
       parsed_reply = []
       for string in split_reply:
           num = string.split()
           parsed_reply.append([int(num[0]), int(num[1])])
       if sorted(parsed_reply) == sorted(clue_arr):
           stat = True
   return stat
```

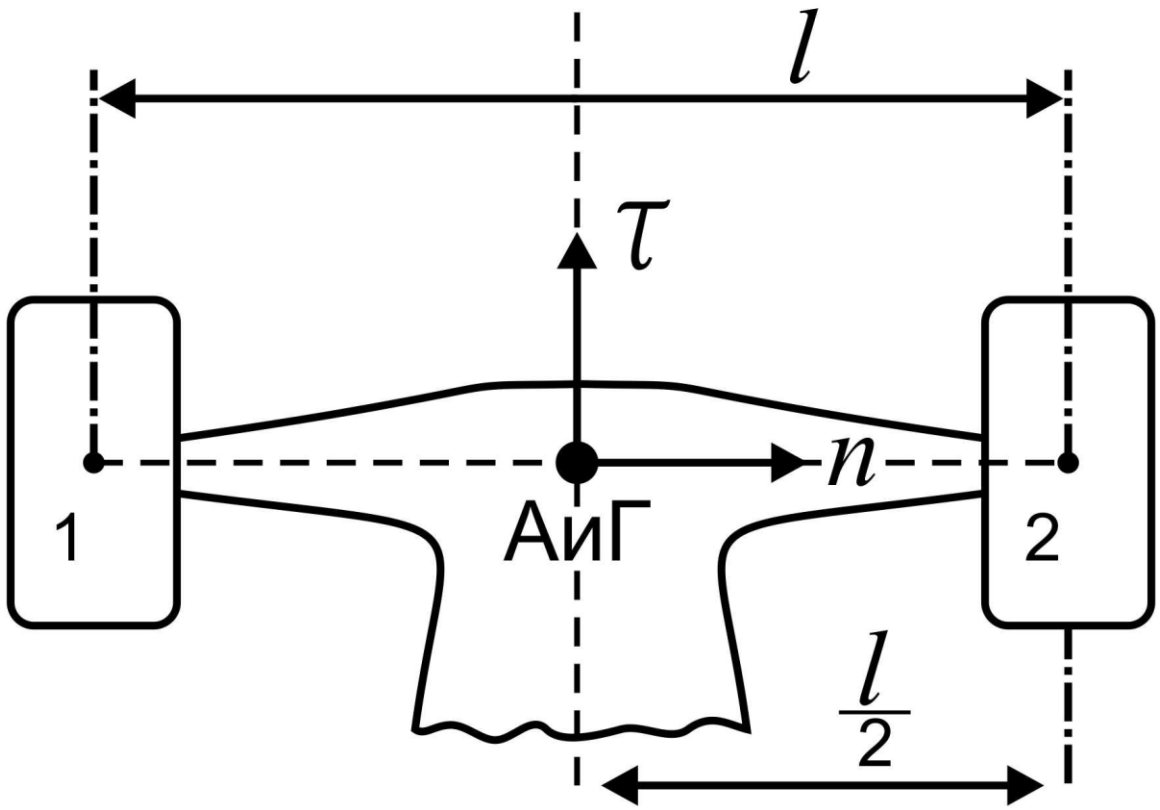
Задача 2.12 (0 баллов)

Условие:

На беспилотной машинке установлены четыре датчика: акселерометр, гироскоп и два датчика скорости.

Акселерометр и гироскоп расположены в одной точке и жёстко закреплены на машинке. Они откалиброваны и стабилизированы относительно вертикальной оси.

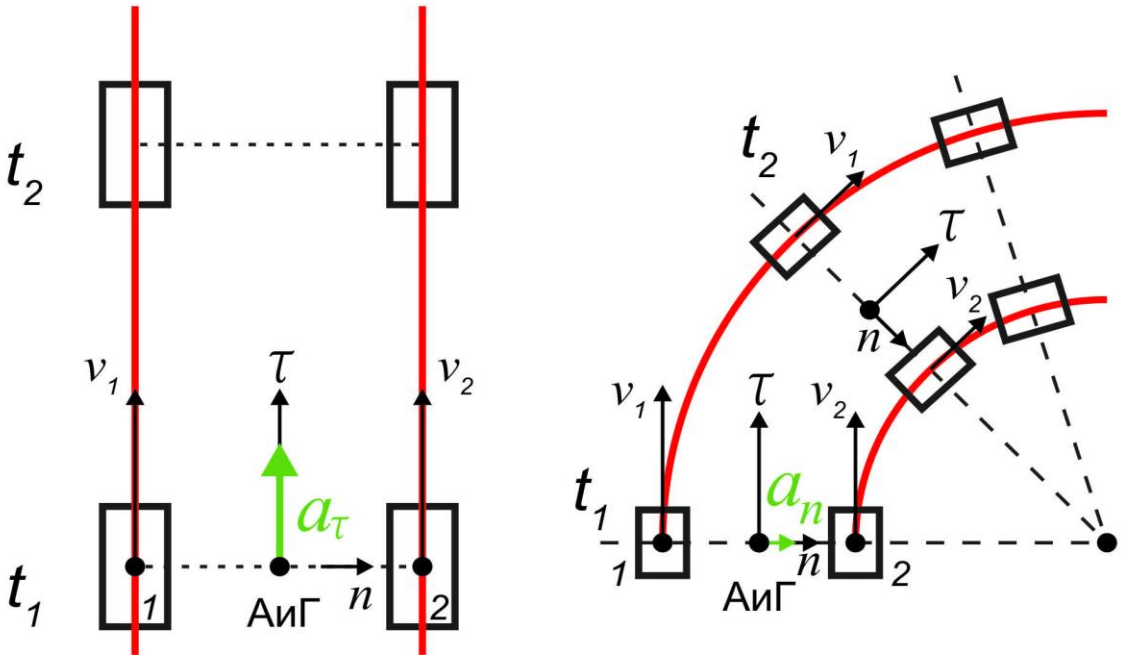
Акселерометр измеряет значение ускорения по двум горизонтальным осям t и n (то есть в системе координат, связанной с машинкой). Гироскоп измеряет значение угловой скорости вращения относительно вертикальной оси. Датчики скорости расположены в центре передних колёс и измеряют модуль значения собственной скорости в горизонтальной плоскости. Расстояние между датчиками скорости 0,2 метра.



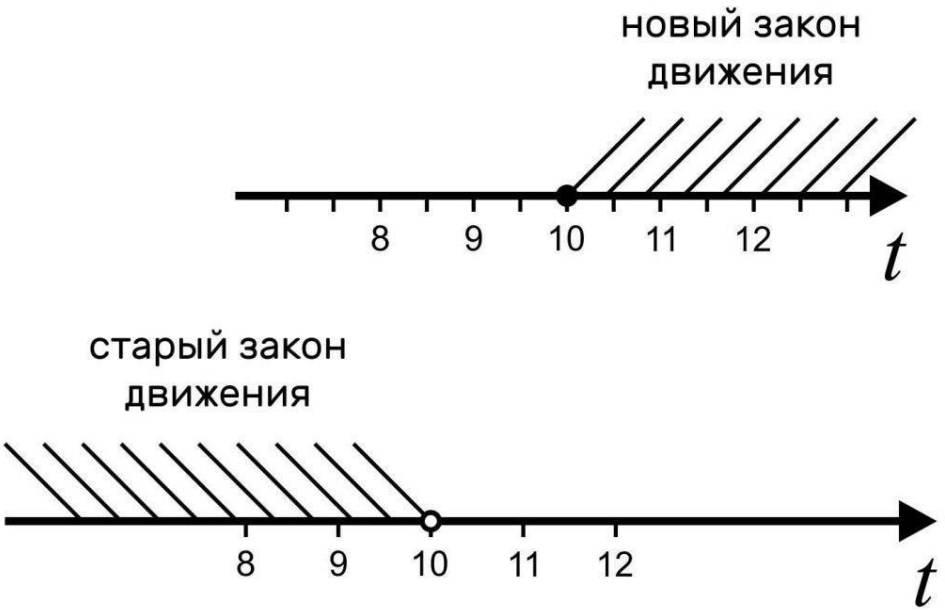
Беспилотная машинка движется в горизонтальной плоскости. На старте машинка неподвижна, направление оси акселерометра t совпадает с направлением оси Y. Стартовые координаты (0, 0).

Машина может двигаться двумя способами:

- по прямой, только вперёд, с ускорением линейно зависящим от времени. Ускорение может принимать отрицательные значения.
- совершая поворот по окружности постоянного радиуса, с постоянной угловой скоростью. При повороте направо угловая скорость положительна, при повороте налево - отрицательна.



Данные со всех датчиков каждую секунду записываются в чёрный ящик. Ускорения записываются в м/с^2, угловая скорость в градусах/с, скорости в м/с.Известно, что смена закона движения машинки происходит мгновенно, и может происходить только в начале секунды.



Задача А

Известны показания акселерометра и гироскопа. Требуется восстановить координаты акселерометра на момент последней записи.

Задача Б

Известны данные с датчиков скорости. Требуется восстановить координаты акселерометра на момент последней записи.

Задача В

Известны данные со всех четырёх датчиков. На старте все датчики исправны. Далее один из датчиков ломается и начинает возвращать показания, не соответствующие действительности. Требуется определить, какой датчик стал давать неверные показания и в какой момент времени.

Тесты в следующих шагах.

Задача 2.13 (1 балл)

Задача A, варианты a (открытый пример) и b (тест)

Условие:

Известны показания акселерометра и гироскопа. Требуется восстановить координаты акселерометра (центра машинки) на момент последней записи. Показания с датчиков записываются с интервалом в 1 с. Общее время не превышает 2 минут.

Формат входных данных:

Первая строка содержит суммарное число измерений, каждая последующая строка содержит данные об измерении в следующем формате: число секунд t с запуска машинки (int, >= 0), тангенциальное ускорение a_t (м/с^2), нормальное ускорение a_n (м/с^2), угловая скорость ω (градусов/с) (все три - double, десятичная часть отделяется точкой), разделенные пробелами.

Формат выходных данных:

Выведите координаты в метрах, десятичная часть чисел отделяется точкой. Абсцисса и ордината разделяются пробелом.

```
Sample Input:
  57
  0 0 0 0
  1 1.5 0 0
  2 3 0 0
  3 4.5 0 0
  4 6 0 0
  5 5 0 0
  6 4 0 0
  7 3 0 0
  8 2 0 0
  9 1 0 0
  10 0 0 0
  11 -1 0 0
  12 -2 0 0
  13 0 14.13716694 30
  14 0 14.13716694 30
  15 0 14.13716694 30
  16 0 14.13716694 30
  17 0 14.13716694 30
  18 0 14.13716694 30
  19 0 0 0
  20 -0.8 0 0
  21 -1.6 0 0
  22 -2.4 0 0
  23 -3.2 0 0
  24 -4 0 0
  25 -3 0 0
  26 -2 0 0
  27 -1 0 0
  28 0 -1.413716694 -10
  29 0 -1.413716694 -10
  30 0 -1.413716694 -10
  31 0 -1.413716694 -10
  32 0 -1.413716694 -10
  33 0 -1.413716694 -10
  34 0 -1.413716694 -10
```

```
  35 0 -1.413716694 -10
  36 0 -1.413716694 -10
  37 0 -1.413716694 -10
  38 0 -0.706858347 -5
  39 0 -0.706858347 -5
  40 0 -0.706858347 -5
  41 0 -0.706858347 -5
  42 0 -0.706858347 -5
  43 0 -0.706858347 -5
  44 0 -0.706858347 -5
  45 0 -0.706858347 -5
  46 0 -0.706858347 -5
  47 0 -0.706858347 -5
  48 0 -0.706858347 -5
  49 0 -0.706858347 -5
  50 0 -0.706858347 -5
  51 0 -0.706858347 -5
  52 0 -0.706858347 -5
  53 0 -0.706858347 -5
  54 0 -0.706858347 -5
  55 0 -0.706858347 -5
  56 0 0 0

Sample Output:
  242.361147340337 131.609581405597
```

Решение:

```
int main ()
{
    int n, i, t;
   double At, An, W, Xo, Yo, Vo=0, V=0, R, X=0, Y=0, S, Sr, Sl, Fi=90,
L=0.1;
    scanf("%d", &n);
    Fi=90*3.14/180;
    for (i=0; i<n; i++)
    {
        Vo=V;
        Xo=X;
        Yo=Y;
        scanf("%d", &t);
        scanf("%lf %lf %lf", &At, &An, &W);
        W=W*( 3.14/180);
        if (W==0) //едет прямо
        {
            //Vo=fabs(Vo);
            //At=fabs(At);
            X=Xo+Vo*cos(Fi)+At*cos(Fi)/2;
                  //      printf("X=%lf\n", X);
            Y=Yo+Vo*sin(Fi)+At*sin(Fi)/2;
                //        printf("Y=%lf\n", Y);
            V=Vo+At;
              //          printf("V=%lf\n\n", V);
            //printf("%lf       %lf\n", Fi*180/3.14, W*180/3.14);
        }
        else //поворачивает
        {
            R=fabs(V/W);
```

```
            //printf("R=%lf\n", R);
            //printf("R=%Lf\n", R);
            S=sqrt(2*R*R-2*R*R*(cos(W)));
            //printf("S=%lf\n", S);
            //Sr=(R+2*L)*W;
            //Sl=R*W;
            //S=(Sr+Sl)/2;
            X=Xo+S*cos(Fi-W/2);
                        //printf("X=%lf\n", X);
            Y=Yo+S*sin(Fi-W/2);
                       // printf("Y=%lf\n", Y);
            Fi=Fi-W;
           // if(Fi<0)
             //   Fi=2*3.14+Fi;
                        //printf("Fir=%lf        W=%lf\n\n",
Fi*180/3.14, W*180/3.14);
        }
    }
    printf("%.20lf %.20lf", X, Y);
    return 0;
}
```

Критерии оценки:

```
#1AB

def generate():
     return [("57\n0 0 0 0\n1 1.5 0 0\n2 3 0 0\n3 4.5 0 0\n4 6 0 0\n5
5 0 0\n6 4 0 0\n7 3 0 0\n8 2 0 0\n9 1 0 0\n10 0 0 0\n11 -1 0 0\n12 -2
0 0\n13 0 14.13716694 30\n14 0 14.13716694 30\n15 0 14.13716694 30\n16
0 14.13716694 30\n17 0 14.13716694 30\n18 0 14.13716694 30\n19 0 0
0\n20 -0.8 0 0\n21 -1.6 0 0\n22 -2.4 0 0\n23 -3.2 0 0\n24 -4 0 0\n25 -
3 0 0\n26 -2 0 0\n27 -1 0 0\n28 0 -1.413716694 -10\n29 0 -1.413716694
-10\n30 0 -1.413716694 -10\n31 0 -1.413716694 -10\n32 0 -1.413716694 -
10\n33 0 -1.413716694 -10\n34 0 -1.413716694 -10\n35 0 -1.413716694 -
10\n36 0 -1.413716694 -10\n37 0 -1.413716694 -10\n38 0 -0.706858347 -
5\n39 0 -0.706858347 -5\n40 0 -0.706858347 -5\n41 0 -0.706858347 -
5\n42 0 -0.706858347 -5\n43 0 -0.706858347 -5\n44 0 -0.706858347 -
5\n45 0 -0.706858347 -5\n46 0 -0.706858347 -5\n47 0 -0.706858347 -
5\n48 0 -0.706858347 -5\n49 0 -0.706858347 -5\n50 0 -0.706858347 -
5\n51 0 -0.706858347 -5\n52 0 -0.706858347 -5\n53 0 -0.706858347 -
5\n54 0 -0.706858347 -5\n55 0 -0.706858347 -5\n56 0 0 0\n",
"242.361147340337 131.609581405597"), ("50\n0 0 0 0\n1 2 0 0\n2 4 0
0\n3 6 0 0\n4 6.333333333 0 0\n5 6.666666667 0 0\n6 7 0 0\n7
7.333333333 0 0\n8 7.666666667 0 0\n9 8 0 0\n10 4 0 0\n11 0
9.267698328 9\n12 0 9.267698328 9\n13 0 9.267698328 9\n14 0
9.267698328 9\n15 0 9.267698328 9\n16 0 9.267698328 9\n17 0
9.267698328 9\n18 0 9.267698328 9\n19 0 9.267698328 9\n20 0
9.267698328 9\n21 0 9.267698328 9\n22 -0.666666667 0 0\n23 -
1.333333333 0 0\n24 -2 0 0\n25 -2.666666667 0 0\n26 -3.333333333 0
0\n27 -4 0 0\n28 -2.666666667 0 0\n29 -1.333333333 0 0\n30 0
6.44026494 9\n31 0 6.44026494 9\n32 0 6.44026494 9\n33 0 6.44026494
9\n34 0 6.44026494 9\n35 0 6.44026494 9\n36 0 6.44026494 9\n37 0
6.44026494 9\n38 0 6.44026494 9\n39 0 6.44026494 9\n40 0 0 0\n41 -
0.666666667 0 0\n42 -1.333333333 0 0\n43 -2 0 0\n44 -2.666666667 0
```

```
0\n45 -3.333333333 0 0\n46 -4 0 0\n47 -2.666666667 0 0\n48 -
1.333333333 0 0\n49 0 0 0\n", "1095.61977236758 113.258225692831")]

def solve(dataset):
     truedatasetclue1 = ("57\n0 0 0 0\n1 1.5 0 0\n2 3 0 0\n3 4.5 0
0\n4 6 0 0\n5 5 0 0\n6 4 0 0\n7 3 0 0\n8 2 0 0\n9 1 0 0\n10 0 0 0\n11
-1 0 0\n12 -2 0 0\n13 0 14.13716694 30\n14 0 14.13716694 30\n15 0
14.13716694 30\n16 0 14.13716694 30\n17 0 14.13716694 30\n18 0
14.13716694 30\n19 0 0 0\n20 -0.8 0 0\n21 -1.6 0 0\n22 -2.4 0 0\n23 -
3.2 0 0\n24 -4 0 0\n25 -3 0 0\n26 -2 0 0\n27 -1 0 0\n28 0 -1.413716694
-10\n29 0 -1.413716694 -10\n30 0 -1.413716694 -10\n31 0 -1.413716694 -
10\n32 0 -1.413716694 -10\n33 0 -1.413716694 -10\n34 0 -1.413716694 -
10\n35 0 -1.413716694 -10\n36 0 -1.413716694 -10\n37 0 -1.413716694 -
10\n38 0 -0.706858347 -5\n39 0 -0.706858347 -5\n40 0 -0.706858347 -
5\n41 0 -0.706858347 -5\n42 0 -0.706858347 -5\n43 0 -0.706858347 -
5\n44 0 -0.706858347 -5\n45 0 -0.706858347 -5\n46 0 -0.706858347 -
5\n47 0 -0.706858347 -5\n48 0 -0.706858347 -5\n49 0 -0.706858347 -
5\n50 0 -0.706858347 -5\n51 0 -0.706858347 -5\n52 0 -0.706858347 -
5\n53 0 -0.706858347 -5\n54 0 -0.706858347 -5\n55 0 -0.706858347 -
5\n56 0 0 0\n", "242.361147340337 131.609581405597")
     truedatasetclue2 = ("50\n0 0 0 0\n1 2 0 0\n2 4 0 0\n3 6 0 0\n4
6.333333333 0 0\n5 6.666666667 0 0\n6 7 0 0\n7 7.333333333 0 0\n8
7.666666667 0 0\n9 8 0 0\n10 4 0 0\n11 0 9.267698328 9\n12 0
9.267698328 9\n13 0 9.267698328 9\n14 0 9.267698328 9\n15 0
9.267698328 9\n16 0 9.267698328 9\n17 0 9.267698328 9\n18 0
9.267698328 9\n19 0 9.267698328 9\n20 0 9.267698328 9\n21 0
9.267698328 9\n22 -0.666666667 0 0\n23 -1.333333333 0 0\n24 -2 0 0\n25
-2.666666667 0 0\n26 -3.333333333 0 0\n27 -4 0 0\n28 -2.666666667 0
0\n29 -1.333333333 0 0\n30 0 6.44026494 9\n31 0 6.44026494 9\n32 0
6.44026494 9\n33 0 6.44026494 9\n34 0 6.44026494 9\n35 0 6.44026494
9\n36 0 6.44026494 9\n37 0 6.44026494 9\n38 0 6.44026494 9\n39 0
6.44026494 9\n40 0 0 0\n41 -0.666666667 0 0\n42 -1.333333333 0 0\n43 -
2 0 0\n44 -2.666666667 0 0\n45 -3.333333333 0 0\n46 -4 0 0\n47 -
2.666666667 0 0\n48 -1.333333333 0 0\n49 0 0 0\n", "1095.61977236758
113.258225692831")
     if dataset == truedatasetclue1[0]:
          return truedatasetclue1[1]
     elif dataset == truedatasetclue2[0]:
          return truedatasetclue2[1]
     else:
          return ""

def check(reply, clue):
   answer = [float(x) for x in reply.split()]
   rightanswer = [float(x) for x in clue.split()]

   if clue == "242.361147340337 131.609581405597":
       result = (abs(answer[0] - rightanswer[0]) < 24/5) and
(abs(answer[1] - rightanswer[1]) < 116/5)
   elif clue == "1095.61977236758 113.258225692831":
       result = (abs(answer[0] - rightanswer[0]) < 177/5) and
(abs(answer[1] - rightanswer[1]) < 212/5)
   else:
       result = (False, "Please contact quiz team. Use comments
below.")

   return result
```

Задача 2.14 (1 балл)

Задача А, вариант с (тест)

Условие:

Известны показания акселерометра и гироскопа. Требуется восстановить координаты акселерометра (центра машинки) на момент последней записи. Показания с датчиков записываются с интервалом в 1 с. Общее время не превышает 2 минут.

Формат входных данных

Первая строка содержит суммарное число измерений, каждая последующая строка содержит данные об измерении в следующем формате: число секунд t с запуска машинки (int, >= 0), тангенциальное ускорение a_t (м/с^2), нормальное ускорение a_n (м/с^2), угловая скорость ω (градусов/с) (все три - double, десятичная часть отделяется точкой), разделенные пробелами.

Формат выходных данных

Выведите координаты в метрах, десятичная часть чисел отделяется точкой. Абсцисса и ордината разделяются пробелом.

Решение:

```
#include <iostream>
#include <iomanip>
#include <math.h>

using namespace std;

int main(int argc, char *argv[])
{
      int sum;
      double vLin=0, x=0, y=0, napr=0,w,an,at,t;
      double pi=M_PI;//3.14;//
      cin>>sum;
      for(int i=0; i<sum-1; i++){
            cin>>t>>at>>an>>w;
            double r=0,rw=0,raw=0;
            if(w==0){
                r=rw=raw=0;
                   double vx=vLin*sin(napr);
                   double vy=vLin*cos(napr);
                   double ax=at*sin(napr);
                   double ay=at*cos(napr);
                   x+=(vx+ax/2);
                   y+=(vy+ay/2);
                   vLin+=at;
           }else{
                  w=w*pi/180.0;
                   //r=vLin/w;
                   //rw=vLin*vLin/an;
                   raw=an/(w*w);
                   r=raw;
                   x=x+r*(cos(napr)-cos(w+napr));
                   y=y-r*(sin(napr)-sin(w+napr));
                   napr+=w;
                   vLin=w*r;
            }
            //cout<<i<<": x="<<x<<" y="<<y<<" v="<<vLin<<" w="<<w<<"
n="<<napr<<" r="<<r<<endl; //<<","<<rw<<","<<raw
      }
```

```
    cout<< fixed << setprecision(12)<<x<<" "<<y<<endl; //242 131
    return 0;
}
```

Критерии оценки:

```
# 1C

def generate():
     return [("73\n0 0 0 0\n1 0.25 0 0\n2 0.5 0 0\n3 0.75 0 0\n4 1 0
0\n5 3 0 0\n6 5 0 0\n7 7 0 0\n8 9 0 0\n9 6 0 0\n10 3 0 0\n11 0
9.293878267 15\n12 0 9.293878267 15\n13 0 9.293878267 15\n14 0
9.293878267 15\n15 0 9.293878267 15\n16 0 9.293878267 15\n17 0
9.293878267 15\n18 0 9.293878267 15\n19 0 9.293878267 15\n20 0
9.293878267 15\n21 0 9.293878267 15\n22 0 9.293878267 15\n23 0
9.293878267 15\n24 0 9.293878267 15\n25 0 9.293878267 15\n26 0
9.293878267 15\n27 0 9.293878267 15\n28 0 9.293878267 15\n29 0 0 0\n30
0.5 0 0\n31 1 0 0\n32 1.5 0 0\n33 2 0 0\n34 1.2 0 0\n35 0.4 0 0\n36 -
0.4 0 0\n37 -1.2 0 0\n38 -2 0 0\n39 -1.5 0 0\n40 -1 0 0\n41 -0.5 0
0\n42 0 -3.717551307 -6\n43 0 -3.717551307 -6\n44 0 -3.717551307 -
6\n45 0 -3.717551307 -6\n46 0 -3.717551307 -6\n47 0 -3.717551307 -
6\n48 0 -3.717551307 -6\n49 0 -3.717551307 -6\n50 0 -3.717551307 -
6\n51 0 -3.717551307 -6\n52 0 -3.717551307 -6\n53 0 -3.717551307 -
6\n54 0 -3.717551307 -6\n55 0 -3.717551307 -6\n56 0 -3.717551307 -
6\n57 0 -3.717551307 -6\n58 0 -3.717551307 -6\n59 0 -3.717551307 -
6\n60 0 -3.717551307 -6\n61 0 -3.717551307 -6\n62 0 -3.717551307 -
6\n63 0 -3.717551307 -6\n64 0 -3.717551307 -6\n65 0 -3.717551307 -
6\n66 0 -3.717551307 -6\n67 0 -3.717551307 -6\n68 0 -3.717551307 -
6\n69 0 -3.717551307 -6\n70 0 -3.717551307 -6\n71 0 -3.717551307 -
6\n72 0 -3.717551307 -6\n", "-364.899988485708 -680.600069085771")]

def solve(dataset):
     truedatasetclue = ("73\n0 0 0 0\n1 0.25 0 0\n2 0.5 0 0\n3 0.75 0
0\n4 1 0 0\n5 3 0 0\n6 5 0 0\n7 7 0 0\n8 9 0 0\n9 6 0 0\n10 3 0 0\n11
0 9.293878267 15\n12 0 9.293878267 15\n13 0 9.293878267 15\n14 0
9.293878267 15\n15 0 9.293878267 15\n16 0 9.293878267 15\n17 0
9.293878267 15\n18 0 9.293878267 15\n19 0 9.293878267 15\n20 0
9.293878267 15\n21 0 9.293878267 15\n22 0 9.293878267 15\n23 0
9.293878267 15\n24 0 9.293878267 15\n25 0 9.293878267 15\n26 0
9.293878267 15\n27 0 9.293878267 15\n28 0 9.293878267 15\n29 0 0 0\n30
0.5 0 0\n31 1 0 0\n32 1.5 0 0\n33 2 0 0\n34 1.2 0 0\n35 0.4 0 0\n36 -
0.4 0 0\n37 -1.2 0 0\n38 -2 0 0\n39 -1.5 0 0\n40 -1 0 0\n41 -0.5 0
0\n42 0 -3.717551307 -6\n43 0 -3.717551307 -6\n44 0 -3.717551307 -
6\n45 0 -3.717551307 -6\n46 0 -3.717551307 -6\n47 0 -3.717551307 -
6\n48 0 -3.717551307 -6\n49 0 -3.717551307 -6\n50 0 -3.717551307 -
6\n51 0 -3.717551307 -6\n52 0 -3.717551307 -6\n53 0 -3.717551307 -
6\n54 0 -3.717551307 -6\n55 0 -3.717551307 -6\n56 0 -3.717551307 -
6\n57 0 -3.717551307 -6\n58 0 -3.717551307 -6\n59 0 -3.717551307 -
6\n60 0 -3.717551307 -6\n61 0 -3.717551307 -6\n62 0 -3.717551307 -
6\n63 0 -3.717551307 -6\n64 0 -3.717551307 -6\n65 0 -3.717551307 -
6\n66 0 -3.717551307 -6\n67 0 -3.717551307 -6\n68 0 -3.717551307 -
6\n69 0 -3.717551307 -6\n70 0 -3.717551307 -6\n71 0 -3.717551307 -
6\n72 0 -3.717551307 -6\n", "-364.899988485708 -680.600069085771")
     if dataset == truedatasetclue[0]:
          return truedatasetclue[1]
```

```
      else:
            return ""

def check(reply, clue):
      answer = [float(x) for x in reply.split()]
      rightanswer = [float(x) for x in clue.split()]

      return (abs(answer[0] - rightanswer[0]) < 134/5) and
(abs(answer[1] - rightanswer[1]) < 207/5)
```

Задача 2.15 (1 балл)

Задача A, вариант d (тест)

Условие:

Известны показания акселерометра и гироскопа. Требуется восстановить координаты акселерометра (центра машинки) на момент последней записи. Показания с датчиков записываются с интервалом в 1 с. Общее время не превышает 2 минут.

Формат входных данных

Первая строка содержит суммарное число измерений, каждая последующая строка содержит данные об измерении в следующем формате: число секунд t с запуска машинки (int, >= 0), тангенциальное ускорение a_t (м/с^2), нормальное ускорение a_n (м/с^2), угловая скорость ω (градусов/с) (все три - double, десятичная часть отделяется точкой), разделенные пробелами.

Формат выходных данных

Выведите координаты в метрах, десятичная часть чисел отделяется точкой. Абсцисса и ордината разделяются пробелом.

Решение:

```
#include <iostream>
#include <cmath>
using namespace std;

struct DANNIE
{
      int t;
      double omega;
      double tang;
      double boost;
      double speed;
      double x;
      double y;
      double R;
      double  fi;

};

int main() {
      const double pi = 3.1415926535;
      int n = 0;
      cin >> n;

      DANNIE *mas = new DANNIE[n];
      for (int i = 0; i < n; i++) {
```

```
            cin >> mas[i].t >> mas[i].tang >> mas[i].boost >>
mas[i].omega;
            mas[i].omega = mas[i].omega*(pi / 180);
      }
      mas[0].x = 0;
      mas[0].y = 0;
      mas[0].fi = 0;
      mas[0].speed = 0;
      for (int i = 1; i < n; i++) {
            mas[i].speed = mas[i - 1].speed + mas[i].tang;
      }
      for (int i = 1; i < n; i++) {

            mas[i].fi = mas[i - 1].fi + mas[i].omega;

      }
      for (int i = 1; i < n; i++) {
            mas[i].x = mas[i - 1].x +
sin(mas[i].fi)*((mas[i].speed+mas[i-1].speed)/2);
            mas[i].y = mas[i - 1].y + cos(mas[i].fi)*((mas[i].speed +
mas[i - 1].speed) / 2);
      }

      cout << mas[n-1].x << " " << mas[n - 1].y;

      delete [] mas;
      return 0;
}
```

Критерии оценки:

```
# 1D

def generate():
      return [("51\n0 0 0 0\n1 1 0 0\n2 2 0 0\n3 3 0 0\n4 2.8 0 0\n5
2.6 0 0\n6 2.4 0 0\n7 2.2 0 0\n8 2 0 0\n9 1.8 0 0\n10 1.6 0 0\n11 1.4
0 0\n12 1.2 0 0\n13 1 0 0\n14 0.8 0 0\n15 0.6 0 0\n16 0.4 0 0\n17 0.2
0 0\n18 0 -14.13716694 -30\n19 0 -14.13716694 -30\n20 0 -14.13716694 -
30\n21 0 -14.13716694 -30\n22 0 -14.13716694 -30\n23 0 -14.13716694 -
30\n24 0 -4.71238898 -10\n25 0 -4.71238898 -10\n26 0 -4.71238898 -
10\n27 0 -4.71238898 -10\n28 0 -4.71238898 -10\n29 0 -4.71238898 -
10\n30 0 -4.71238898 -10\n31 0 -4.71238898 -10\n32 0 -4.71238898 -
10\n33 0 0 0\n34 2 0 0\n35 4 0 0\n36 3 0 0\n37 2 0 0\n38 1 0 0\n39 0 -
8.927359124 -15\n40 0 -8.927359124 -15\n41 0 -8.927359124 -15\n42 0 -
8.927359124 -15\n43 0 -8.927359124 -15\n44 0 -8.927359124 -15\n45 0 -
8.927359124 -15\n46 0 -8.927359124 -15\n47 0 -8.927359124 -15\n48 0 -
8.927359124 -15\n49 0 -8.927359124 -15\n50 0 -8.927359124 -15\n",
"224.166201561776 402.806206167492")]

def solve(dataset):
      truedatasetclue = ("51\n0 0 0 0\n1 1 0 0\n2 2 0 0\n3 3 0 0\n4
2.8 0 0\n5 2.6 0 0\n6 2.4 0 0\n7 2.2 0 0\n8 2 0 0\n9 1.8 0 0\n10 1.6 0
0\n11 1.4 0 0\n12 1.2 0 0\n13 1 0 0\n14 0.8 0 0\n15 0.6 0 0\n16 0.4 0
0\n17 0.2 0 0\n18 0 -14.13716694 -30\n19 0 -14.13716694 -30\n20 0 -
14.13716694 -30\n21 0 -14.13716694 -30\n22 0 -14.13716694 -30\n23 0 -
14.13716694 -30\n24 0 -4.71238898 -10\n25 0 -4.71238898 -10\n26 0 -
4.71238898 -10\n27 0 -4.71238898 -10\n28 0 -4.71238898 -10\n29 0 -
```

```
4.71238898 -10\n30 0 -4.71238898 -10\n31 0 -4.71238898 -10\n32 0 -
4.71238898 -10\n33 0 0 0\n34 2 0 0\n35 4 0 0\n36 3 0 0\n37 2 0 0\n38 1
0 0\n39 0 -8.927359124 -15\n40 0 -8.927359124 -15\n41 0 -8.927359124 -
15\n42 0 -8.927359124 -15\n43 0 -8.927359124 -15\n44 0 -8.927359124 -
15\n45 0 -8.927359124 -15\n46 0 -8.927359124 -15\n47 0 -8.927359124 -
15\n48 0 -8.927359124 -15\n49 0 -8.927359124 -15\n50 0 -8.927359124 -
15\n", "224.166201561776 402.806206167492")

     if dataset == truedatasetclue[0]:
          return truedatasetclue[1]
     else:
          return ""

def check(reply, clue):
     answer = [float(x) for x in reply.split()]
     rightanswer = [float(x) for x in clue.split()]

     return (abs(answer[0] - rightanswer[0]) < 70/5) and
(abs(answer[1] - rightanswer[1]) < 116/5)
```

Задача 2.16 (1 балл)

Задача Б, вариант с (тест)

Условие:

Известны данные с датчиков скорости колес. Требуется восстановить координаты акселерометра (центра машинки) на момент последней записи. Показания с датчиков записываются с интервалом в 1 с. Общее время не превышает 2 минут.

Формат входных данных

Первая строка содержит суммарное число измерений, каждая последующая строка содержит данные об измерении в следующем формате: число секунд t с запуска машинки (int, >= 0), скорость v_1 (м/с), скорость v_2 (м/с) (обе - double, десятичная часть отделяется точкой), разделенные пробелами.

Формат выходных данных

Выведите координаты в метрах, десятичная часть чисел отделяется точкой. Абсцисса и координата разделяются пробелом.

Решение:

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

int main ()
{
    int n, i, t;
   long double At, Aty, W, X0=0, a, Y0=0, V=0,V0=0, R, X=0, Y=0, S,
Sr, Sl, Fi=0, L=0.1,Vd0=0,Vb0=0,Vd,Vb, r=0;
    scanf("%d", &n);
    for (i=0; i<n; i++)
    {
        V0=V;
        X0=X;
        Y0=Y;
        scanf("%d", &t);
        scanf("%Lf %Lf", &Vd, &Vb); //11 59.0157 58.9843
```

```
        if(Vd<Vb)
        {
            a=Vd;
            Vd=Vb;
            Vb=a;
            r=1;
        }
        if (Vd0==Vb0)
        {
           // if(Vd==Vb)
           // {
                V0=(Vd0+Vb0)/2;
                At=(Vd+Vb)/2-V0;
                X=X0+V0*sin(Fi)+At*sin(Fi)/2;
                Y=Y0+V0*cos(Fi)+At*cos(Fi)/2;
                V=V0+At;

        }
        else
        {
            R=0.1*(Vb0+Vd0)/(Vd0-Vb0);
            if(r==1)
            {
              W=(-1)*(Vd0+Vb0)/2/R;
            }
            else
            {
                W=(Vd0+Vb0)/2/R;
            }
                S=sqrt(2*R*R-2*R*R*(cos(W)));
                X=X0+S*sin(Fi+W/2);
                Y=Y0+S*cos(Fi+W/2);
                Fi=Fi+W;
        }
            //}
            //else
            //{
            //    R=abs(0.1*(Vb+Vd)/(Vd-Vb));
             //   if(r==1)
             //   {
              //    //      W=(-1)*(Vd+Vb)/2/R;
              //    //  }
              //    //  else
              //    //  {
              //    //      W=(Vd+Vb)/2/R;
              //    //  }
               // // S=sqrt(2*R*R-2*R*R*(cos(W)));
               // // X=Xo+S*sin(Fi+W/2);
               // // Y=Yo+S*cos(Fi+W/2);
               // // Fi=Fi+W;
            //}
        //}                                                   //   Vd*R-
0.1*Vd=Vb*R+Vb*0.1
       // else                                                 //
R=0.1*(Vb+Vd)/(Vd-Vb);
        // {
            // R=abs(0.1*(Vb+Vd)/(Vd-Vb));
```

```
            // if(r==1)
             //{
             //    W=(-1)*(Vd+Vb)/2/R;
            // }
            // else
             //{
             //    W=(Vd+Vb)/2/R;
            // }
           //  S=sqrt(2*R*R-2*R*R*(cos(W)));
          //   X=Xo+S*sin(Fi+W/2);
          //   Y=Yo+S*cos(Fi+W/2);
         //    Fi=Fi+W;
        // }
         Vd0=Vd;
         Vb0=Vb;
         r=0;
         //printf("% d %.12Lf %.12Lf %.12Lf\n", t-1, X, Y,
(Fi*180/3.14));
    }
    printf("%.12Lf %.12Lf", X, Y);
    return 0;
}
```

Критерии оценки:

```
# 2BA

def generate():
     return [("50\n0 0 0\n1 1 1\n2 4 4\n3 9 9\n4 15.1667 15.1667\n5
21.6667 21.6667\n6 28.5 28.5\n7 35.6667 35.6667\n8 43.1667 43.1667\n9
51 51\n10 57 57\n11 59.0157 58.9843\n12 59.0157 58.9843\n13 59.0157
58.9843\n14 59.0157 58.9843\n15 59.0157 58.9843\n16 59.0157
58.9843\n17 59.0157 58.9843\n18 59.0157 58.9843\n19 59.0157
58.9843\n20 59.0157 58.9843\n21 59 59\n22 58.6667 58.6667\n23 57.6667
57.6667\n24 56 56\n25 53.6667 53.6667\n26 50.6667 50.6667\n27 47
47\n28 43.6667 43.6667\n29 41.6667 41.6667\n30 41.0157 40.9843\n31
41.0157 40.9843\n32 41.0157 40.9843\n33 41.0157 40.9843\n34 41.0157
40.9843\n35 41.0157 40.9843\n36 41.0157 40.9843\n37 41.0157
40.9843\n38 41.0157 40.9843\n39 41.0157 40.9843\n40 41 41\n41 40.6667
40.6667\n42 39.6667 39.6667\n43 38 38\n44 35.6667 35.6667\n45 32.6667
32.6667\n46 29 29\n47 25.6667 25.6667\n48 23.6667 23.6667\n49 23
23\n", "1095.61977236758 113.258225692831"), ("57\n0 0 0\n1 0.75
0.75\n2 3 3\n3 6.75 6.75\n4 12 12\n5 17.5 17.5\n6 22 22\n7 25.5
25.5\n8 28 28\n9 29.5 29.5\n10 30 30\n11 29.5 29.5\n12 28 28\n13
27.05235988 26.94764012\n14 27.05235988 26.94764012\n15 27.05235988
26.94764012\n16 27.05235988 26.94764012\n17 27.05235988
26.94764012\n18 27.05235988 26.94764012\n19 27 27\n20 26.6 26.6\n21
25.4 25.4\n22 23.4 23.4\n23 20.6 20.6\n24 16.1 16.1\n25 12.6 12.6\n26
10.1 10.1\n27 8.6 8.6\n28 8.082546707 8.117453293\n29 8.082546707
8.117453293\n30 8.082546707 8.117453293\n31 8.082546707
8.117453293\n32 8.082546707 8.117453293\n33 8.082546707
8.117453293\n34 8.082546707 8.117453293\n35 8.082546707
8.117453293\n36 8.082546707 8.117453293\n37 8.082546707
8.117453293\n38 8.091273354 8.108726646\n39 8.091273354
8.108726646\n40 8.091273354 8.108726646\n41 8.091273354
8.108726646\n42 8.091273354 8.108726646\n43 8.091273354
```

```
8.108726646\n44 8.091273354 8.108726646\n45 8.091273354
8.108726646\n46 8.091273354 8.108726646\n47 8.091273354
8.108726646\n48 8.091273354 8.108726646\n49 8.091273354
8.108726646\n50 8.091273354 8.108726646\n51 8.091273354
8.108726646\n52 8.091273354 8.108726646\n53 8.091273354
8.108726646\n54 8.091273354 8.108726646\n55 8.091273354
8.108726646\n56 8.1 8.1\n", "242.361147340337 131.609581405597")]

def solve(dataset):
     truedatasetclue1 = ("50\n0 0 0\n1 1 1\n2 4 4\n3 9 9\n4 15.1667
15.1667\n5 21.6667 21.6667\n6 28.5 28.5\n7 35.6667 35.6667\n8 43.1667
43.1667\n9 51 51\n10 57 57\n11 59.0157 58.9843\n12 59.0157 58.9843\n13
59.0157 58.9843\n14 59.0157 58.9843\n15 59.0157 58.9843\n16 59.0157
58.9843\n17 59.0157 58.9843\n18 59.0157 58.9843\n19 59.0157
58.9843\n20 59.0157 58.9843\n21 59 59\n22 58.6667 58.6667\n23 57.6667
57.6667\n24 56 56\n25 53.6667 53.6667\n26 50.6667 50.6667\n27 47
47\n28 43.6667 43.6667\n29 41.6667 41.6667\n30 41.0157 40.9843\n31
41.0157 40.9843\n32 41.0157 40.9843\n33 41.0157 40.9843\n34 41.0157
40.9843\n35 41.0157 40.9843\n36 41.0157 40.9843\n37 41.0157
40.9843\n38 41.0157 40.9843\n39 41.0157 40.9843\n40 41 41\n41 40.6667
40.6667\n42 39.6667 39.6667\n43 38 38\n44 35.6667 35.6667\n45 32.6667
32.6667\n46 29 29\n47 25.6667 25.6667\n48 23.6667 23.6667\n49 23
23\n", "1095.61977236758 113.258225692831")
     truedatasetclue2 = ("57\n0 0 0\n1 0.75 0.75\n2 3 3\n3 6.75
6.75\n4 12 12\n5 17.5 17.5\n6 22 22\n7 25.5 25.5\n8 28 28\n9 29.5
29.5\n10 30 30\n11 29.5 29.5\n12 28 28\n13 27.05235988 26.94764012\n14
27.05235988 26.94764012\n15 27.05235988 26.94764012\n16 27.05235988
26.94764012\n17 27.05235988 26.94764012\n18 27.05235988
26.94764012\n19 27 27\n20 26.6 26.6\n21 25.4 25.4\n22 23.4 23.4\n23
20.6 20.6\n24 16.1 16.1\n25 12.6 12.6\n26 10.1 10.1\n27 8.6 8.6\n28
8.082546707 8.117453293\n29 8.082546707 8.117453293\n30 8.082546707
8.117453293\n31 8.082546707 8.117453293\n32 8.082546707
8.117453293\n33 8.082546707 8.117453293\n34 8.082546707
8.117453293\n35 8.082546707 8.117453293\n36 8.082546707
8.117453293\n37 8.082546707 8.117453293\n38 8.091273354
8.108726646\n39 8.091273354 8.108726646\n40 8.091273354
8.108726646\n41 8.091273354 8.108726646\n42 8.091273354
8.108726646\n43 8.091273354 8.108726646\n44 8.091273354
8.108726646\n45 8.091273354 8.108726646\n46 8.091273354
8.108726646\n47 8.091273354 8.108726646\n48 8.091273354
8.108726646\n49 8.091273354 8.108726646\n50 8.091273354
8.108726646\n51 8.091273354 8.108726646\n52 8.091273354
8.108726646\n53 8.091273354 8.108726646\n54 8.091273354
8.108726646\n55 8.091273354 8.108726646\n56 8.1 8.1\n",
"242.361147340337 131.609581405597")
     if dataset == truedatasetclue1[0]:
          return truedatasetclue1[1]
     elif dataset == truedatasetclue2[0]:
          return truedatasetclue2[1]
     else:
          return ""

def check(reply, clue):
   answer = [float(x) for x in reply.split()]
   rightanswer = [float(x) for x in clue.split()]

   if clue == "242.361147340337 131.609581405597":
```

```
       result = (abs(answer[0] - rightanswer[0]) < 24/5) and
(abs(answer[1] - rightanswer[1]) < 116/5)
   elif clue == "1095.61977236758 113.258225692831":
       result = (abs(answer[0] - rightanswer[0]) < 177/5) and
(abs(answer[1] - rightanswer[1]) < 212/5)
   else:
       result = (False, "Please contact quiz team. Use comments
below.")

   return result
```

Задача 2.17 (1 балл)

Задача Б, вариант d (тест)

Условие:

Известны данные с датчиков скорости колес. Требуется восстановить координаты акселерометра (центра машинки) на момент последней записи. Показания с датчиков записываются с интервалом в 1 с. Общее время не превышает 2 минут.

Формат входных данных

Первая строка содержит суммарное число измерений, каждая последующая строка содержит данные об измерении в следующем формате: число секунд t с запуска машинки (int, >= 0), скорость v_1 (м/с), скорость v_2 (м/с) (обе - double, десятичная часть отделяется точкой), разделенные пробелами.

Формат выходных данных

Выведите координаты в метрах, десятичная часть чисел отделяется точкой. Абсцисса и ордината разделяются пробелом.

Sample input:

```
 0,00    0,0000   0,0000
 1,00    0,5000   0,5000
 2,00    2,0000   2,0000
 3,00    4,5000   4,5000
 4,00    7,4000   7,4000
 5,00    10,1000  10,1000
 6,00    12,6000  12,6000
 7,00    14,9000  14,9000
 8,00    17,0000  17,0000
 9,00    18,9000  18,9000
 10,00   20,6000  20,6000
 11,00   22,1000  22,1000
 12,00   23,4000  23,4000
 13,00   24,5000  24,5000
 14,00   25,4000  25,4000
```

15,00	26,1000	26,1000
16,00	26,6000	26,6000
17,00	26,9000	26,9000
18,00	26,9476	27,0524
19,00	26,9476	27,0524
20,00	26,9476	27,0524
21,00	26,9476	27,0524
22,00	26,9476	27,0524
23,00	26,9476	27,0524
24,00	26,9825	27,0175
25,00	26,9825	27,0175
26,00	26,9825	27,0175
27,00	26,9825	27,0175
28,00	26,9825	27,0175
29,00	26,9825	27,0175
30,00	26,9825	27,0175
31,00	26,9825	27,0175
32,00	26,9825	27,0175
33,00	27,0000	27,0000
34,00	28,0000	28,0000
35,00	31,0000	31,0000
36,00	34,5000	34,5000
37,00	37,0000	37,0000
38,00	38,5000	38,5000
39,00	38,9738	39,0262
40,00	38,9738	39,0262
41,00	38,9738	39,0262
42,00	38,9738	39,0262
43,00	38,9738	39,0262
44,00	38,9738	39,0262
45,00	38,9738	39,0262
46,00	38,9738	39,0262

```
 47,00    38,9738  39,0262

 48,00    38,9738  39,0262

 49,00    38,9738  39,0262

 50,00    38,9738  39,0262
```

Sample output:

```
 253,562

 440,239
```

Решение:

```
#include <iostream>
#include <math.h>

using namespace std;

int main(int argc, char *argv[])
{
      int sum;
      float t, x=0, y=0, napr=0, vp, vl;
      float pi=3.141527;
      cin>>sum;
      for(int i=0; i<sum; i++){
            cin>>t>>vp>>vl;
            if(vp==vl){
                  x+=vp*sin(napr);
                  y+=vp*cos(napr);
           }else{
                 float r=(0.2/((vp/vl)-1))+0.1;
                 float V=(vp+vl)/2;
                 float w=V/r;
                  x=(x+r*cos(napr))+r*sin(w+napr-(pi/2));
                  y=(y-r*sin(napr))+r*cos(w+napr-(pi/2));
                  napr+=w;
            }
            //cout<<t<<": x="<<x<<" y="<<y<<"napr="<<napr<<endl<<endl;
      }

    cout<<x<<" "<<y<<endl; //1095.61977236758 113.258225692831
    return 0;
}
```

Критерии оценки:

```
# 2D

def generate():
      return [("51\n0 0 0\n1 0.5 0.5\n2 2 2\n3 4.5 4.5\n4 7.4 7.4\n5
10.1 10.1\n6 12.6 12.6\n7 14.9 14.9\n8 17 17\n9 18.9 18.9\n10 20.6
20.6\n11 22.1 22.1\n12 23.4 23.4\n13 24.5 24.5\n14 25.4 25.4\n15 26.1
26.1\n16 26.6 26.6\n17 26.9 26.9\n18 26.9476 27.0524\n19 26.9476
```

```
27.0524\n20 26.9476 27.0524\n21 26.9476 27.0524\n22 26.9476
27.0524\n23 26.9476 27.0524\n24 26.9825 27.0175\n25 26.9825
27.0175\n26 26.9825 27.0175\n27 26.9825 27.0175\n28 26.9825
27.0175\n29 26.9825 27.0175\n30 26.9825 27.0175\n31 26.9825
27.0175\n32 26.9825 27.0175\n33 22.1 22.1\n34 23.1 23.1\n35 26.1
26.1\n36 29.6 29.6\n37 32.1 32.1\n38 33.6 33.6\n39 34.0738 34.1262\n40
34.0738 34.1262\n41 34.0738 34.1262\n42 34.0738 34.1262\n43 34.0738
34.1262\n44 34.0738 34.1262\n45 34.0738 34.1262\n46 34.0738
34.1262\n47 34.0738 34.1262\n48 34.0738 34.1262\n49 34.0738
34.1262\n50 34.0738 34.1262\n", "224.166201561776 402.806206167492")]

def solve(dataset):
     truedatasetclue = ("51\n0 0 0\n1 0.5 0.5\n2 2 2\n3 4.5 4.5\n4
7.4 7.4\n5 10.1 10.1\n6 12.6 12.6\n7 14.9 14.9\n8 17 17\n9 18.9
18.9\n10 20.6 20.6\n11 22.1 22.1\n12 23.4 23.4\n13 24.5 24.5\n14 25.4
25.4\n15 26.1 26.1\n16 26.6 26.6\n17 26.9 26.9\n18 26.9476 27.0524\n19
26.9476 27.0524\n20 26.9476 27.0524\n21 26.9476 27.0524\n22 26.9476
27.0524\n23 26.9476 27.0524\n24 26.9825 27.0175\n25 26.9825
27.0175\n26 26.9825 27.0175\n27 26.9825 27.0175\n28 26.9825
27.0175\n29 26.9825 27.0175\n30 26.9825 27.0175\n31 26.9825
27.0175\n32 26.9825 27.0175\n33 22.1 22.1\n34 23.1 23.1\n35 26.1
26.1\n36 29.6 29.6\n37 32.1 32.1\n38 33.6 33.6\n39 34.0738 34.1262\n40
34.0738 34.1262\n41 34.0738 34.1262\n42 34.0738 34.1262\n43 34.0738
34.1262\n44 34.0738 34.1262\n45 34.0738 34.1262\n46 34.0738
34.1262\n47 34.0738 34.1262\n48 34.0738 34.1262\n49 34.0738
34.1262\n50 34.0738 34.1262\n", "224.166201561776 402.806206167492")

     if dataset == truedatasetclue[0]:
          return truedatasetclue[1]
     else:
          return ""

def check(reply, clue):
     answer = [float(x) for x in reply.split()]
     rightanswer = [float(x) for x in clue.split()]

     return (abs(answer[0] - rightanswer[0]) < 70/5) and
(abs(answer[1] - rightanswer[1]) < 116/5)
```

Задача 2.18 (1 балл)

Задача B, вариант c (тест)

Условие:

Известны данные со всех четырёх датчиков. На старте все датчики исправны. Далее один из датчиков ломается и начинает давать показания, не соответствующие действительности. Сломанный датчик ошибается минимум на единицу. Требуется определить, какой датчик стал давать неверные показания и в какой момент времени. Показания с датчиков записываются с интервалом в 1 с. Общее время не превышает 2 минут.

Формат входных данных

Первая строка содержит суммарное число измерений, каждая последующая строка содержит данные об измерении в следующем формате: число секунд t с запуска машинки (int, >= 0), тангенциальное ускорение a_t (м/с^2), нормальное ускорение a_n (м/с^2), угловая скорость ω (градусов/с), скорость v_1 (м/с), скорость v_2 (м/с) (все пять - double, десятичная часть отделяется точкой), разделенные пробелами.

Формат выходных данных

Выведите обозначение датчика и целое число секунд *t*, при котором было впервые зафиксировано ложное показание, разделенное пробелом. Датчики обозначаются следующим образом:

- акселерометр: A
- гироскоп: G
- первый датчик скорости: V1
- второй датчик скорости: V2

Решение:

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




int main(int argc, char *argv[])
{


//FILE * fo;
//fo = fopen("testot.txt","wt");




const double Pi = 3.14159265358979323846264338327950288;
//                3.14159265358979323846264338327950288
int n;
//FILE * fi;
//fi = fopen("test.txt","rt");

scanf("%d", &n );


int t[n]; //time
int p[n]; //process
double at[n];
double an[n];
double omg[n]; // w
double r[n];
int s;
double vl[n];
vl[0] = 0;
double vr[n];
vr[0] = 0;
double x [n];
x[0]=0.;
double y [n];
y[0]=0.;
double alfa[n];
```

```
alfa [0] = 90.;
//  double jerk[n];
// jerk[0] = 0;
int i;
for(i=0; i<n; ++i)
{
scanf("%d", &t[i] );
scanf("%lf", &at[i] );
scanf("%lf", &an[i] );
scanf("%lf", &omg[i] );
scanf("%lf", &vl[i] );
scanf("%lf", &vr[i] );
}

 // ------------------------------------------------
for(i=0; i<n; ++i) // if,if,if -> switch
{

      if ((at[i] != 0)&&(an[i] != 0))
      {
            printf("%s%d","A ", i);
            exit(0) ;
      }
      if ((omg[i] != 0)&&((at[i]!= 0)&&(vl[i]==vr[i])))
      {
            printf("%s%d", "G ", i);
            exit(0) ;
      }
      if ((omg[i] == 0)&&(an[i] != 0))
      {
            printf("%s%d","A ", i);
            exit(0) ;
      }
      if (((omg[i] > 0)&&(vl[i] <= vr[i]))||((omg[i] < 0)&&(vl[i] >=
vr[i])))
      {
            if (abs(vl[i] -
(omg[i]*Pi/180*(an[i]/(omg[i]*omg[i]*Pi*Pi/180/180)-0.1))) > 0.001 )
            {
                  printf("%s%d","V1 ", i);
                  exit(0) ;
            }
            if (abs(vr[i] -
(omg[i]*Pi/180*(an[i]/(omg[i]*omg[i]*Pi*Pi/180/180)+0.1))) > 0.001 )
            {
                  printf("%s%d","V2 ", i);
                  exit(0) ;
            }
      }



}
```

```
return 0;
}
```

Критерии оценки:

```
# 3DA

def generate():
     return [("51\n0 0.00 0 0.00 0.0000 0.0000\n1 1.00 0 0.00 0.5000
0.5000\n2 2.00 0 0.00 2.0000 2.0000\n3 3.00 0 0.00 4.5000 4.5000\n4
2.80 0 0.00 7.4000 7.4000\n5 2.60 0 0.00 10.1000 10.1000\n6 2.40 0
0.00 12.6000 12.6000\n7 2.20 0 0.00 14.9000 14.9000\n8 2.00 0 0.00
17.0000 17.0000\n9 1.80 0 0.00 18.9000 18.9000\n10 1.60 0 0.00 20.6000
20.6000\n11 1.40 0 0.00 22.1000 22.1000\n12 1.20 0 0.00 23.4000
23.4000\n13 1.00 0 0.00 24.5000 24.5000\n14 0.80 0 0.00 25.4000
25.4000\n15 0.60 0 0.00 26.1000 26.1000\n16 0.40 0 0.00 26.6000
26.6000\n17 0.20 0 0.00 26.9000 26.9000\n18 0.00 -14.13716694 -30.00
26.9476 27.0524\n19 0.00 -14.13716694 -30.00 26.9476 27.0524\n20 0.00
-14.13716694 -30.00 26.9476 27.0524\n21 0.00 -14.13716694 -30.00
26.9476 27.0524\n22 0.00 -14.13716694 -30.00 26.9476 27.0524\n23 0.00
-14.13716694 -30.00 26.9476 27.0524\n24 0.00 -4.71238898 -10.00
26.9825 27.0175\n25 0.00 -4.71238898 -10.00 26.9825 27.0175\n26 0.00 -
4.71238898 -10.00 26.9825 27.0175\n27 0.00 -4.71238898 -10.00 26.9825
27.0175\n28 0.00 -4.71238898 -10.00 26.9825 27.0175\n29 0.00 -
4.71238898 -10.00 26.9825 27.0175\n30 0.00 -4.71238898 -10.00 26.9825
27.0175\n31 0.00 -4.71238898 -10.00 26.9825 27.0175\n32 0.00 -
4.71238898 -10.00 26.9825 27.0175\n33 0.00 0 0.00 22.1000 22.1000\n34
2.00 0 0.00 23.1000 23.1000\n35 4.00 0 0.00 26.1000 26.1000\n36 3.00 0
0.00 29.6000 29.6000\n37 2.00 0 0.00 32.1000 32.1000\n38 1.00 0 0.00
33.6000 33.6000\n39 0.00 -8.927359124 -15.00 34.0738 34.0738\n40 0.00
-8.927359124 -15.00 34.0738 34.0738\n41 0.00 -8.927359124 -15.00
34.0738 34.0738\n42 0.00 -8.927359124 -15.00 34.0738 34.0738\n43 0.00
-8.927359124 -15.00 34.0738 34.0738\n44 0.00 -8.927359124 -15.00
34.0738 34.0738\n45 0.00 -8.927359124 -15.00 34.0738 34.0738\n46 0.00
-8.927359124 -15.00 34.0738 34.0738\n47 0.00 -8.927359124 -15.00
34.0738 34.0738\n48 0.00 -8.927359124 -15.00 34.0738 34.0738\n49 0.00
-8.927359124 -15.00 34.0738 34.0738\n50 0.00 -8.927359124 -15.00
34.0738 34.0738\n", "V2 39"), ("57\n0 0 0 0 0 0\n1 1.5 0 0 0.75
0.75\n2 3 0 0 3 3\n3 4.5 0 0 6.75 6.75\n4 6 0 0 12 12\n5 5 0 0 17.5
17.5\n6 4 0 0 22 22\n7 3 0 0 25.5 25.5\n8 2 0 0 28 28\n9 1 0 0 29.5
29.5\n10 0 0 0 30 30\n11 -1 0 0 29.5 29.5\n12 -2 0 0 28 28\n13 0
14.13716694 30 27.05235988 26.94764012\n14 0 14.13716694 30
27.05235988 26.94764012\n15 0 14.13716694 30 27.05235988
26.94764012\n16 0 14.13716694 30 27.05235988 26.94764012\n17 0
14.13716694 30 27.05235988 26.94764012\n18 0 14.13716694 30
27.05235988 26.94764012\n19 0 0 30 27 27\n20 -0.8 0 0 26.6 26.6\n21 -
1.6 0 0 25.4 25.4\n22 -2.4 0 0 23.4 23.4\n23 -3.2 0 0 20.6 20.6\n24 -4
0 0 16.1 16.1\n25 -3 0 0 12.6 12.6\n26 -2 0 0 10.1 10.1\n27 -1 0 0 8.6
8.6\n28 0 -1.413716694 -10 8.082546707 8.117453293\n29 0 -1.413716694
-10 8.082546707 8.117453293\n30 0 -1.413716694 -10 8.082546707
8.117453293\n31 0 -1.413716694 -10 8.082546707 8.117453293\n32 0 -
1.413716694 -10 8.082546707 8.117453293\n33 0 -1.413716694 -10
8.082546707 8.117453293\n34 0 -1.413716694 -10 8.082546707
8.117453293\n35 0 -1.413716694 -10 8.082546707 8.117453293\n36 0 -
1.413716694 -10 8.082546707 8.117453293\n37 0 -1.413716694 -10
8.082546707 8.117453293\n38 0 -0.706858347 -5 8.091273354
8.108726646\n39 0 -0.706858347 -5 8.091273354 8.108726646\n40 0 -
```

```
0.706858347 -5 8.091273354 8.108726646\n41 0 -0.706858347 -5
8.091273354 8.108726646\n42 0 -0.293141653 -5 8.091273354
8.108726646\n43 0 -0.293141653 -5 8.091273354 8.108726646\n44 0 -
0.293141653 -5 8.091273354 8.108726646\n45 0 -0.293141653 -5
8.091273354 8.108726646\n46 0 -0.293141653 -5 8.091273354
8.108726646\n47 0 -0.293141653 -5 8.091273354 8.108726646\n48 0 -
0.293141653 -5 8.091273354 8.108726646\n49 0 -0.293141653 -5
8.091273354 8.108726646\n50 0 -0.293141653 -5 8.091273354
8.108726646\n51 0 -0.293141653 -5 8.091273354 8.108726646\n52 0 -
0.293141653 -5 8.091273354 8.108726646\n53 0 -0.293141653 -5
8.091273354 8.108726646\n54 0 -0.293141653 -5 8.091273354
8.108726646\n55 0 -0.293141653 -5 8.091273354 8.108726646\n56 0 0 0
8.1 8.1\n", "A 42")]

def solve(dataset):
     truedatasetclue1 = ("51\n0 0.00 0 0.00 0.0000 0.0000\n1 1.00 0
0.00 0.5000 0.5000\n2 2.00 0 0.00 2.0000 2.0000\n3 3.00 0 0.00 4.5000
4.5000\n4 2.80 0 0.00 7.4000 7.4000\n5 2.60 0 0.00 10.1000 10.1000\n6
2.40 0 0.00 12.6000 12.6000\n7 2.20 0 0.00 14.9000 14.9000\n8 2.00 0
0.00 17.0000 17.0000\n9 1.80 0 0.00 18.9000 18.9000\n10 1.60 0 0.00
20.6000 20.6000\n11 1.40 0 0.00 22.1000 22.1000\n12 1.20 0 0.00
23.4000 23.4000\n13 1.00 0 0.00 24.5000 24.5000\n14 0.80 0 0.00
25.4000 25.4000\n15 0.60 0 0.00 26.1000 26.1000\n16 0.40 0 0.00
26.6000 26.6000\n17 0.20 0 0.00 26.9000 26.9000\n18 0.00 -14.13716694
-30.00 26.9476 27.0524\n19 0.00 -14.13716694 -30.00 26.9476
27.0524\n20 0.00 -14.13716694 -30.00 26.9476 27.0524\n21 0.00 -
14.13716694 -30.00 26.9476 27.0524\n22 0.00 -14.13716694 -30.00
26.9476 27.0524\n23 0.00 -14.13716694 -30.00 26.9476 27.0524\n24 0.00
-4.71238898 -10.00 26.9825 27.0175\n25 0.00 -4.71238898 -10.00 26.9825
27.0175\n26 0.00 -4.71238898 -10.00 26.9825 27.0175\n27 0.00 -
4.71238898 -10.00 26.9825 27.0175\n28 0.00 -4.71238898 -10.00 26.9825
27.0175\n29 0.00 -4.71238898 -10.00 26.9825 27.0175\n30 0.00 -
4.71238898 -10.00 26.9825 27.0175\n31 0.00 -4.71238898 -10.00 26.9825
27.0175\n32 0.00 -4.71238898 -10.00 26.9825 27.0175\n33 0.00 0 0.00
22.1000 22.1000\n34 2.00 0 0.00 23.1000 23.1000\n35 4.00 0 0.00
26.1000 26.1000\n36 3.00 0 0.00 29.6000 29.6000\n37 2.00 0 0.00
32.1000 32.1000\n38 1.00 0 0.00 33.6000 33.6000\n39 0.00 -8.927359124
-15.00 34.0738 34.0738\n40 0.00 -8.927359124 -15.00 34.0738
34.0738\n41 0.00 -8.927359124 -15.00 34.0738 34.0738\n42 0.00 -
8.927359124 -15.00 34.0738 34.0738\n43 0.00 -8.927359124 -15.00
34.0738 34.0738\n44 0.00 -8.927359124 -15.00 34.0738 34.0738\n45 0.00
-8.927359124 -15.00 34.0738 34.0738\n46 0.00 -8.927359124 -15.00
34.0738 34.0738\n47 0.00 -8.927359124 -15.00 34.0738 34.0738\n48 0.00
-8.927359124 -15.00 34.0738 34.0738\n49 0.00 -8.927359124 -15.00
34.0738 34.0738\n50 0.00 -8.927359124 -15.00 34.0738 34.0738\n", "V2
39")
     truedatasetclue2 = ("57\n0 0 0 0 0 0\n1 1.5 0 0 0.75 0.75\n2 3 0
0 3 3\n3 4.5 0 0 6.75 6.75\n4 6 0 0 12 12\n5 5 0 0 17.5 17.5\n6 4 0 0
22 22\n7 3 0 0 25.5 25.5\n8 2 0 0 28 28\n9 1 0 0 29.5 29.5\n10 0 0 0
30 30\n11 -1 0 0 29.5 29.5\n12 -2 0 0 28 28\n13 0 14.13716694 30
27.05235988 26.94764012\n14 0 14.13716694 30 27.05235988
26.94764012\n15 0 14.13716694 30 27.05235988 26.94764012\n16 0
14.13716694 30 27.05235988 26.94764012\n17 0 14.13716694 30
27.05235988 26.94764012\n18 0 14.13716694 30 27.05235988
26.94764012\n19 0 0 30 27 27\n20 -0.8 0 0 26.6 26.6\n21 -1.6 0 0 25.4
25.4\n22 -2.4 0 0 23.4 23.4\n23 -3.2 0 0 20.6 20.6\n24 -4 0 0 16.1
16.1\n25 -3 0 0 12.6 12.6\n26 -2 0 0 10.1 10.1\n27 -1 0 0 8.6 8.6\n28
0 -1.413716694 -10 8.082546707 8.117453293\n29 0 -1.413716694 -10
```

```
8.082546707 8.117453293\n30 0 -1.413716694 -10 8.082546707
8.117453293\n31 0 -1.413716694 -10 8.082546707 8.117453293\n32 0 -
1.413716694 -10 8.082546707 8.117453293\n33 0 -1.413716694 -10
8.082546707 8.117453293\n34 0 -1.413716694 -10 8.082546707
8.117453293\n35 0 -1.413716694 -10 8.082546707 8.117453293\n36 0 -
1.413716694 -10 8.082546707 8.117453293\n37 0 -1.413716694 -10
8.082546707 8.117453293\n38 0 -0.706858347 -5 8.091273354
8.108726646\n39 0 -0.706858347 -5 8.091273354 8.108726646\n40 0 -
0.706858347 -5 8.091273354 8.108726646\n41 0 -0.706858347 -5
8.091273354 8.108726646\n42 0 -0.293141653 -5 8.091273354
8.108726646\n43 0 -0.293141653 -5 8.091273354 8.108726646\n44 0 -
0.293141653 -5 8.091273354 8.108726646\n45 0 -0.293141653 -5
8.091273354 8.108726646\n46 0 -0.293141653 -5 8.091273354
8.108726646\n47 0 -0.293141653 -5 8.091273354 8.108726646\n48 0 -
0.293141653 -5 8.091273354 8.108726646\n49 0 -0.293141653 -5
8.091273354 8.108726646\n50 0 -0.293141653 -5 8.091273354
8.108726646\n51 0 -0.293141653 -5 8.091273354 8.108726646\n52 0 -
0.293141653 -5 8.091273354 8.108726646\n53 0 -0.293141653 -5
8.091273354 8.108726646\n54 0 -0.293141653 -5 8.091273354
8.108726646\n55 0 -0.293141653 -5 8.091273354 8.108726646\n56 0 0 0
8.1 8.1\n", "A 42")
      if dataset == truedatasetclue1[0]:
            return truedatasetclue1[1]
      elif dataset == truedatasetclue2[0]:
            return truedatasetclue2[1]
      else:
            return ""

def check(reply, clue):
      answer = reply.split()
      rightanswer = clue.split()

      return (answer[0] == rightanswer[0]) and (answer[1] ==
rightanswer[1])
```

Задача 2.19 (1 балл)

Задача B, вариант d (тест)

Условие:

Известны данные со всех четырёх датчиков. На старте все датчики исправны. Далее один из датчиков ломается и начинает давать показания, не соответствующие действительности. Сломанный датчик ошибается минимум на единицу. Требуется определить, какой датчик стал давать неверные показания и в какой момент времени. Показания с датчиков записываются с интервалом в 1 с. Общее время не превышает 2 минут.

Формат входных данных

Первая строка содержит суммарное число измерений, каждая последующая строка содержит данные об измерении в следующем формате: число секунд t с запуска машинки (int, >= 0), тангенциальное ускорение a_t (м/с^2), нормальное ускорение a_n (м/с^2), угловая скорость ω (градусов/с), скорость v_1 (м/с), скорость v_2 (м/с) (все пять - double, десятичная часть отделяется точкой), разделенные пробелами.

Формат выходных данных

Выведите обозначение датчика и целое число секунд t, при котором было впервые зафиксировано ложное показание, разделенное пробелом. Датчики обозначаются следующим образом:

- акселерометр: A
- гироскоп: G
- первый датчик скорости: V1
- второй датчик скорости: V2

Sample input:

```
 0,00    0,00    0        0,00    0,0000   0,0000
 1,00    1,00    0        0,00    0,5000   0,5000
 2,00    2,00    0        0,00    2,0000   2,0000
 3,00    3,00    0        0,00    4,5000   4,5000
 4,00    2,80    0        0,00    7,4000   7,4000
 5,00    2,60    0        0,00    10,1000  10,1000
 6,00    2,40    0        0,00    12,6000  12,6000
 7,00    2,20    0        0,00    14,9000  14,9000
 8,00    2,00    0        0,00    17,0000  17,0000
 9,00    1,80    0        0,00    18,9000  18,9000
 10,00   1,60    0        0,00    20,6000  20,6000
 11,00   1,40    0        0,00    22,1000  22,1000
 12,00   1,20    0        0,00    23,4000  23,4000
 13,00   1,00    0        0,00    24,5000  24,5000
 14,00   0,80    0        0,00    25,4000  25,4000
 15,00   0,60    0        0,00    26,1000  26,1000
 16,00   0,40    0        0,00    26,6000  26,6000
 17,00   0,20    0        0,00    26,9000  26,9000
 18,00   0,00    -        -30,00  26,9476  27,0524
                 14,1372
 19,00   0,00    -        -30,00  26,9476  27,0524
                 14,1372
 20,00   0,00    -        -30,00  26,9476  27,0524
                 14,1372
 21,00   0,00    -        -30,00  26,9476  27,0524
                 14,1372
 22,00   0,00    -        -30,00  26,9476  27,0524
                 14,1372
 23,00   0,00    -        -30,00  26,9476  27,0524
                 14,1372
```

24,00	0,00	-4,71239	-10,00	26,9825	27,0175
25,00	0,00	-4,71239	-10,00	26,9825	27,0175
26,00	0,00	-4,71239	-10,00	26,9825	27,0175
27,00	0,00	-4,71239	-10,00	26,9825	27,0175
28,00	0,00	-4,71239	-10,00	26,9825	27,0175
29,00	0,00	-4,71239	-10,00	26,9825	27,0175
30,00	0,00	-4,71239	-10,00	26,9825	27,0175
31,00	0,00	-4,71239	-10,00	26,9825	27,0175
32,00	0,00	-4,71239	-10,00	26,9825	27,0175
33,00	0,00	0	0,00	27,0000	27,0000
34,00	2,00	0	0,00	28,0000	28,0000
35,00	4,00	0	0,00	31,0000	31,0000
36,00	3,00	0	0,00	34,5000	34,5000
37,00	2,00	0	0,00	37,0000	37,0000
38,00	1,00	0	0,00	38,5000	38,5000
39,00	0,00	-8,92736	-15,00	34,0738	34,0738
40,00	0,00	-8,92736	-15,00	34,0738	34,0738
41,00	0,00	-8,92736	-15,00	34,0738	34,0738
42,00	0,00	-8,92736	-15,00	34,0738	34,0738
43,00	0,00	-8,92736	-15,00	34,0738	34,0738
44,00	0,00	-8,92736	-15,00	34,0738	34,0738
45,00	0,00	-8,92736	-15,00	34,0738	34,0738
46,00	0,00	-	-15,00	34,0738	34,0738

```
                  8,92736

 47,00    0,00     -        -15,00   34,0738  34,0738
                  8,92736

 48,00    0,00     -        -15,00   34,0738  34,0738
                  8,92736

 49,00    0,00     -        -15,00   34,0738  34,0738
                  8,92736

 50,00    0,00     -        -15,00   34,0738  34,0738
                  8,92736
Sample Output:
  V2 39
```

Решение:

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
      int N, sec = 0;
      double At, An, AngleG = 0, AngleAn = 0, AngleV = 0, Gyro, v1 =
0, v2 = 0, V = 0, Va = 0, X = 0, Y = 0 ;
      cin >> N;

      for (int i = 0; i < N; i++) {
            cin >> sec >> At >> An >> Gyro >> v1 >>v2;
            V = (v1 + v2) / 2;
            Va += At;

            AngleG = Gyro* 3.14 / 180;
            if (V!=0) {
                  AngleAn = An / V;
            }
            AngleV = (v1 - v2) / 0.2;

            if (abs(AngleG) - abs(AngleV) < 0.002 && abs(AngleG) -
abs(AngleAn) > 0.002) {
                  cout << "A " << sec;
                  break;
            }
            if (abs(AngleAn) - abs(AngleV) < 0.002 && abs(AngleG) -
abs(AngleAn) > 0.002) {
                  cout << "G " << sec;
                  break;
            }

            if (abs(AngleG)-abs(AngleV) > 0.002 && abs(AngleG) -
abs(AngleAn) < 0.002) {
                  if (AngleG > 0) {
                        cout << "V1 "<< sec;
                        break;
                  }
                  else
```

```
                {
                      cout << "V2 " << sec;
                      break;
                }
          }
     }
    return 0;
}
```

Критерии оценки:

```
# 3C

def generate():
     return [("73\n0 0 0 0 0.0000 0.0000\n1 0.25 0 0 0.1250 0.1250\n2
0.5 0 0 0.5000 0.5000\n3 0.75 0 0 1.1250 1.1250\n4 1 0 0 2.0000
2.0000\n5 3 0 0 4.0000 4.0000\n6 5 0 0 8.0000 8.0000\n7 7 0 0 14.0000
14.0000\n8 9 0 0 22.0000 22.0000\n9 6 0 0 29.5000 29.5000\n10 3 0 0
34.0000 34.0000\n11 0 9.293878267 15 35.5262 35.4738\n12 0 9.293878267
15 35.5262 35.4738\n13 0 9.293878267 15 35.5262 35.4738\n14 0
9.293878267 15 35.5262 35.4738\n15 0 9.293878267 15 35.5262
35.4738\n16 0 9.293878267 15 35.5262 35.4738\n17 0 9.293878267 15
35.5262 35.4738\n18 0 9.293878267 15 35.5262 35.4738\n19 0 9.293878267
15 35.5262 35.4738\n20 0 9.293878267 15 35.5262 35.4738\n21 0
9.293878267 15 35.5262 35.4738\n22 0 9.293878267 15 35.5262
35.4738\n23 0 9.293878267 15 35.5262 35.4738\n24 0 9.293878267 15
35.5262 35.4738\n25 0 9.293878267 15 35.5262 35.4738\n26 0 9.293878267
15 35.5262 35.4738\n27 0 9.293878267 15 35.5262 35.4738\n28 0
9.293878267 15 35.5262 35.4738\n29 0 0 -1 35.5000 35.5000\n30 0.5 0 -1
35.7500 35.7500\n31 1 0 -1 36.5000 36.5000\n32 1.5 0 -1 37.7500
37.7500\n33 2 0 -1 39.5000 39.5000\n34 1.2 0 -1 41.1000 41.1000\n35
0.4 0 -1 41.9000 41.9000\n36 -0.4 0 -1 41.9000 41.9000\n37 -1.2 0 -1
41.1000 41.1000\n38 -2 0 -1 39.5000 39.5000\n39 -1.5 0 -1 37.7500
37.7500\n40 -1 0 -1 36.5000 36.5000\n41 -0.5 0 -1 35.7500 35.7500\n42
0 -3.717551307 -6 35.4895 35.5105\n43 0 -3.717551307 -6 35.4895
35.5105\n44 0 -3.717551307 -6 35.4895 35.5105\n45 0 -3.717551307 -6
35.4895 35.5105\n46 0 -3.717551307 -6 35.4895 35.5105\n47 0 -
3.717551307 -6 35.4895 35.5105\n48 0 -3.717551307 -6 35.4895
35.5105\n49 0 -3.717551307 -6 35.4895 35.5105\n50 0 -3.717551307 -6
35.48952802 35.51047198\n51 0 -3.717551307 -6 35.48952802
35.51047198\n52 0 -3.717551307 -6 35.48952802 35.51047198\n53 0 -
3.717551307 -6 35.48952802 35.51047198\n54 0 -3.717551307 -6
35.48952802 35.51047198\n55 0 -3.717551307 -6 35.48952802
35.51047198\n56 0 -3.717551307 -6 35.48952802 35.51047198\n57 0 -
3.717551307 -6 35.48952802 35.51047198\n58 0 -3.717551307 -6
35.48952802 35.51047198\n59 0 -3.717551307 -6 35.48952802
35.51047198\n60 0 -3.717551307 -6 35.48952802 35.51047198\n61 0 -
3.717551307 -6 35.48952802 35.51047198\n62 0 -3.717551307 -6
35.48952802 35.51047198\n63 0 -3.717551307 -6 35.48952802
35.51047198\n64 0 -3.717551307 -6 35.48952802 35.51047198\n65 0 -
3.717551307 -6 35.48952802 35.51047198\n66 0 -3.717551307 -6
35.48952802 35.51047198\n67 0 -3.717551307 -6 35.48952802
35.51047198\n68 0 -3.717551307 -6 35.48952802 35.51047198\n69 0 -
3.717551307 -6 35.48952802 35.51047198\n70 0 -3.717551307 -6
35.48952802 35.51047198\n71 0 -3.717551307 -6 35.48952802
35.51047198\n72 0 -3.717551307 -6 35.48952802 35.51047198\n", "G 29")]
```

```
def solve(dataset):
     truedatasetclue = ("73\n0 0 0 0 0.0000 0.0000\n1 0.25 0 0 0.1250
0.1250\n2 0.5 0 0 0.5000 0.5000\n3 0.75 0 0 1.1250 1.1250\n4 1 0 0
2.0000 2.0000\n5 3 0 0 4.0000 4.0000\n6 5 0 0 8.0000 8.0000\n7 7 0 0
14.0000 14.0000\n8 9 0 0 22.0000 22.0000\n9 6 0 0 29.5000 29.5000\n10
3 0 0 34.0000 34.0000\n11 0 9.293878267 15 35.5262 35.4738\n12 0
9.293878267 15 35.5262 35.4738\n13 0 9.293878267 15 35.5262
35.4738\n14 0 9.293878267 15 35.5262 35.4738\n15 0 9.293878267 15
35.5262 35.4738\n16 0 9.293878267 15 35.5262 35.4738\n17 0 9.293878267
15 35.5262 35.4738\n18 0 9.293878267 15 35.5262 35.4738\n19 0
9.293878267 15 35.5262 35.4738\n20 0 9.293878267 15 35.5262
35.4738\n21 0 9.293878267 15 35.5262 35.4738\n22 0 9.293878267 15
35.5262 35.4738\n23 0 9.293878267 15 35.5262 35.4738\n24 0 9.293878267
15 35.5262 35.4738\n25 0 9.293878267 15 35.5262 35.4738\n26 0
9.293878267 15 35.5262 35.4738\n27 0 9.293878267 15 35.5262
35.4738\n28 0 9.293878267 15 35.5262 35.4738\n29 0 0 -1 35.5000
35.5000\n30 0.5 0 -1 35.7500 35.7500\n31 1 0 -1 36.5000 36.5000\n32
1.5 0 -1 37.7500 37.7500\n33 2 0 -1 39.5000 39.5000\n34 1.2 0 -1
41.1000 41.1000\n35 0.4 0 -1 41.9000 41.9000\n36 -0.4 0 -1 41.9000
41.9000\n37 -1.2 0 -1 41.1000 41.1000\n38 -2 0 -1 39.5000 39.5000\n39
-1.5 0 -1 37.7500 37.7500\n40 -1 0 -1 36.5000 36.5000\n41 -0.5 0 -1
35.7500 35.7500\n42 0 -3.717551307 -6 35.4895 35.5105\n43 0 -
3.717551307 -6 35.4895 35.5105\n44 0 -3.717551307 -6 35.4895
35.5105\n45 0 -3.717551307 -6 35.4895 35.5105\n46 0 -3.717551307 -6
35.4895 35.5105\n47 0 -3.717551307 -6 35.4895 35.5105\n48 0 -
3.717551307 -6 35.4895 35.5105\n49 0 -3.717551307 -6 35.4895
35.5105\n50 0 -3.717551307 -6 35.48952802 35.51047198\n51 0 -
3.717551307 -6 35.48952802 35.51047198\n52 0 -3.717551307 -6
35.48952802 35.51047198\n53 0 -3.717551307 -6 35.48952802
35.51047198\n54 0 -3.717551307 -6 35.48952802 35.51047198\n55 0 -
3.717551307 -6 35.48952802 35.51047198\n56 0 -3.717551307 -6
35.48952802 35.51047198\n57 0 -3.717551307 -6 35.48952802
35.51047198\n58 0 -3.717551307 -6 35.48952802 35.51047198\n59 0 -
3.717551307 -6 35.48952802 35.51047198\n60 0 -3.717551307 -6
35.48952802 35.51047198\n61 0 -3.717551307 -6 35.48952802
35.51047198\n62 0 -3.717551307 -6 35.48952802 35.51047198\n63 0 -
3.717551307 -6 35.48952802 35.51047198\n64 0 -3.717551307 -6
35.48952802 35.51047198\n65 0 -3.717551307 -6 35.48952802
35.51047198\n66 0 -3.717551307 -6 35.48952802 35.51047198\n67 0 -
3.717551307 -6 35.48952802 35.51047198\n68 0 -3.717551307 -6
35.48952802 35.51047198\n69 0 -3.717551307 -6 35.48952802
35.51047198\n70 0 -3.717551307 -6 35.48952802 35.51047198\n71 0 -
3.717551307 -6 35.48952802 35.51047198\n72 0 -3.717551307 -6
35.48952802 35.51047198\n", "G 29")

     if dataset == truedatasetclue[0]:
          return truedatasetclue[1]
     else:
          return ""

def check(reply, clue):
     answer = reply.split()
     rightanswer = clue.split()

     return (answer[0] == rightanswer[0]) and (answer[1] ==
rightanswer[1])
```