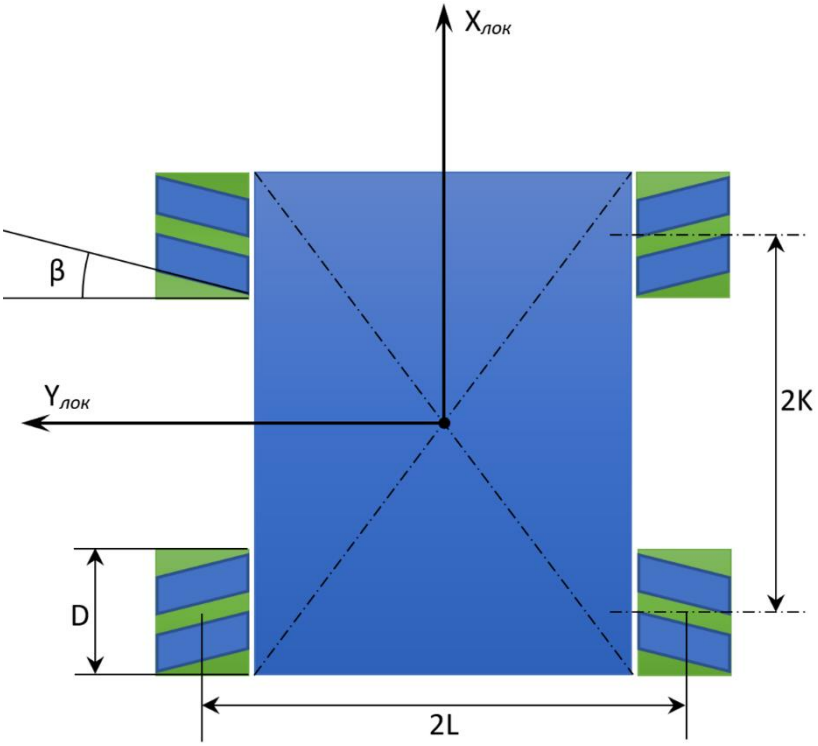
Возрастная категория 9–11 классы

Задача A. Вычисление скоростей mecanum-платформы

Ограничения: по времени 1 сек, по памяти 256 Мб.

Описание робота: mecanum-платформа.



Мобильный робот построен по кинематической схеме с четырьмя колесами Илона (mecanum-wheel) диаметром D мм. Ролики колес отклонены от нормали на β градусов (при β=90 градусов ролики установлены поперек плоскости колеса, как у омни-колес). Обратите внимание, что показан вид робота сверху, то есть в точках касания колес с поверхностью ролики каждого колеса повернуты зеркально.

Расстояние между центрами передних и задних колес $2K$ миллиметров. Между центром левых и правых колес $2L$ миллиметров.

Центр локальной системы координат ($X_{лок}$, $Y_{лок}$) совпадает с геометрическим центром робота и центром между колесами. При этом ось $X_{лок}$ совпадает с направлением робота “вперед”.

Задание:

Робот начинает перемещение с заданными управляющими координатами X, Y и M. Необходимо вычислить скорости вращения колес в рад/с для совершения этого движения.

Формат входных данных

Входные данные состоят из пяти строк.

Первая строка содержит три разделенных пробелами числа, описывающих параметры mecanum-платформы:

Первое число – половина расстояния между центрами левого и правого колес (L), в метрах.

Второе число – половина расстояния между центрами переднего и заднего колес (K), в метрах.

Третье число – угол поворота роликов относительно нормали (β) в градусах.

Вторая строка содержит дробное положительное число от 0.0001 до 1.0 (с точностью до 4 знаков после запятой) – диаметр колес робота (D) в метрах.

Третья строка содержит дробное число от -1.0 до 1.0 (с точностью до 4 знаков после запятой) – составляющая X вектора движения в м/с; положительное значение – движение робота вперед, отрицательное – назад.

Четвертая строка содержит дробное число от -1.0 до 1.0 (с точностью до 4 знаков после запятой) – составляющая Y вектора движения в м/с; положительное значение – движение робота влево, отрицательное – вправо.

Пятая строка содержит дробное число от -1.0 до 1.0 (с точностью до 4 знаков после запятой) – составляющая M (вращение) движения в рад/с; положительное значение – поворот робота против часовой стрелки при взгляде сверху, отрицательное – по часовой стрелке.

Формат выходных данных

Выведите четыре строки с числами, обозначающими скорости вращения колес робота, от -1.0 до 1.0 с точностью до 4 знаков после запятой, положительное значение – вращение колеса вперед, отрицательное – назад.

Первая строка содержит скорость вращение левого переднего колеса в рад/с.

Вторая строка содержит скорость вращения левого заднего колеса в рад/с.

Третья строка содержит скорость вращения правого заднего колеса в рад/с.

Четвертая строка содержит скорость вращения правого переднего колеса в рад/с.

Пример 1

Входные данные:	Выходные данные:
4 4 45	-16.0000
1.0	-14.0000
0.5	16.0000
0.5	18.0000
1.0	

Решение:

Учитывая материалы из видео «Плоскопараллельное движение роботов» и «Роботы на роликовых колесах» можно вывести формулы для вычисления колес:

$$F_{xi} = X - \sqrt{K^2 + L^2} M \sin \alpha_i$$
$$F_{yi} = Y + \sqrt{K^2 + L^2} M \cos \alpha_i$$

$$F_i = \sqrt{F_{xi}^2 + F_{yi}^2}$$

$$\varphi_i = arctan\frac{F_{yi}}{F_{xi}}$$

$$\vartheta_i = F_i \cos\varphi_i - \frac{F_i \sin\varphi_i}{\tan\beta_i}$$

$$\omega_i = \frac{2\vartheta_i}{D}$$

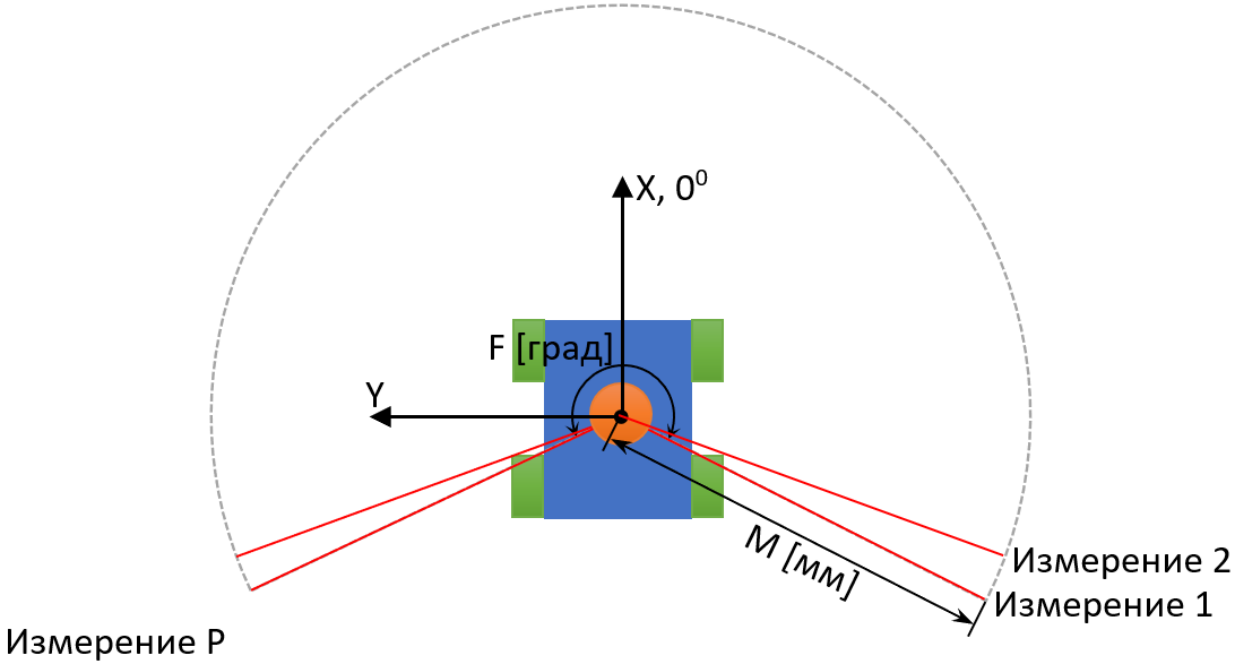
Во всех этих формулах индекс i означает номер колеса, для которого отличаются углы α и β.

Итоговая программа на ЯП Python может иметь следующий вид:

```
from sys import stdin
import sys
import math
import array

L,K,B=map(float,input().split())
B=int(B)
D = float(input())
X = float(input())
Y = float(input())
M = float(input())

alpha=[0 for i in range(4)]
betta=[0 for i in range(4)]
Fx=[0 for i in range(4)]
Fy=[0 for i in range(4)]
Phi=[0 for i in range(4)]
Vel=[0 for i in range(4)]
Ang=[0 for i in range(4)]

alpha[0]=math.atan2(L, K)
alpha[1]=math.atan2(L, -K)
alpha[2]=math.atan2(-L, -K)
alpha[3]=math.atan2(-L, K)
betta[0]=math.radians(B)
betta[1]=math.radians(-B)
betta[2]=math.radians(B)
betta[3]=math.radians(-B)

for i in range(0,4):
    Fx[i] = X - math.sqrt(K*K+L*L)*M*math.sin( alpha[i] )
    Fy[i] = Y + math.sqrt(K*K+L*L)*M*math.cos( alpha[i] )
    F = math.sqrt(Fx[i]**2+Fy[i]**2)
    Phi[i] = math.atan2(Fy[i],Fx[i])
    Vel[i] = F*math.cos(Phi[i]) - F*math.sin(Phi[i]) / math.tan( betta[i] )
    Ang[i]=Vel[i]*2/D

for i in range(0,4):
    print(  float( '{:.4f}'.format(Ang[i]) ) )
```

Задача B. LIDAR

Ограничения: по времени 1 сек, по памяти 256 Мб.



Робот оборудован LIDAR'ом. Угол его сканирования F градусов, на эти F градусов приходится P измерений LIDAR'а.

Максимальное расстояние измерения – до M миллиметров. Если LIDAR не видит объекта, то соответствующее измерение имеет максимальное значение.

Нулевые координаты декартовой системы координат OXY совпадают с центром LIDAR'а, от которого он измеряет расстояние. Ось X этой системы координат совпадает с серединой сектора сканирования LIDAR'а.

Кроме того, в системе присутствует полярная система координат.

Ее полюс совпадает с началом декартовой системы координат и центром LIDAR'а.

А полярная ось (нулевой луч) совпадает с осью X декартовой системы координат.

Задание:

Необходимо перевести координаты каждой найденной точки объекта в декартовы.

При этом измерения LIDAR'а, которые не обнаружили объект, переводить в декартовы координаты не требуется.

Замечание

Готовые библиотеки для решения задачи использовать запрещено.

Примеры входных данных представлены по ссылке и не выводятся в самом задании. Примеры тестовых примеров представлены по ссылке.

Формат входных данных

Первая строка содержит целое число F –угол в градусах ($F \in [1; 360]$).

Вторая строка содержит целое число P – количество измерений ($P \in [1; 4096]$).

Третья строка содержит целое число M – дистанция измерения LIDAR'a в миллиметрах ($M \in [1; 4000]$).

Четвертая строка содержит P целых чисел – показания LIDAR'a.

Формат выходных данных

Первая строка содержит целое число – количество измерений LIDAR'a, в которых обнаружен объект.

Если ни в одном измерении объекты не обнаружены, то строка содержит число 0.

Следующие строки присутствуют только если в первой строке число больше нуля; они содержат через пробел пары чисел с плавающей точкой – координаты X и Y точки обнаруженного объекта с точностью до 0.1 мм.

Вторая строка содержит координаты из первого измерения LIDAR'a, в котором обнаружен объект, третья – из второго измерения, в котором обнаружен объект и т.д.

Решение:

Задача сводится к последовательному переводу координат из полярной в прямоугольную систем координат. Дополнительно вводится условие на проверку не максимальных значений, координаты для которых и преобразуются.

Итоговая программа на ЯП Python может иметь следующий вид:

```
from sys import stdin
import sys
import math

#F,P,M=map(int,input().split())
F=int(input())
P=int(input())
M=int(input())
values = list( map( int, input().split() ) )
xyValues = []
n=0

for i in range(P):
    #print(i)
    if values[i]<M:
        step=P/F
        alpha = i*F/(P-1)-F/2
        #print(alpha)
        X= values[i]*math.cos(math.radians(alpha))
        Y= values[i]*math.sin(math.radians(alpha))
        xyValues.append([0]*2)
        xyValues[n]=[X,Y]
        n+=1

print(n)
for val in xyValues:
    print( float('{:.1f}'.format(val[0])), float( '{:.1f}'.format(val[1])))
```

Задача C. Одометрия материальной точки

Ограничения: по времени 1 сек, по памяти 256 Мб.

В этой задаче робот представлен абстрактной точкой.

Задание:

Робот начинает движение в точке с нулевыми координатами и совершает сложное движение. Каждые *dt* секунд фиксируется смещение робота *dS* и изменение угла поворота *dM*. По известным на каждом шаге движения изменению времени *dt*, пройденному роботом пути *dS* и изменению угла поворота робота *dM* необходимо определить конечные координаты робота.

Формат входных данных

В первой строке записано натуральное число QT ($1 \le QT \le 6 * 10^4$) -- общее количество сделанных измерений.

Следующие QT строк содержат по три числа, разделенных пробелами:

Первое число (дробное, от 0.0001 до 1.0 с точностью до 4 знаков после запятой) – *dt* текущего шага, в секундах.

Второе число (дробное, от -2.0 до 2.0 с точностью до 4 знаков после запятой) – *dS* текущего шага, в метрах.

Третье число (дробное, от -π до π с точностью до 4 знаков после запятой) – *dM* текущего шага, в радианах.

Формат выходных данных

Выведите две строки, содержащих дробные числа, с точностью до четырех знаков после запятой.

В первой строке должна быть указана результирующая *X*-координата робота.

Во второй строке должна быть указана результирующая *Y*-координата робота.

Пример 1

Входные данные:	Выходные данные:
1	-0.0000
1 2 1.5708	2.0000

Решение:

Одометрию материальной точки можно описать формулами

$$\begin{cases} X_n = X_{n-1} + \Delta X \\ Y_n = Y_{n-1} + \Delta Y \\ M_n = M_{n-1} + \Delta M \end{cases}$$

где:

X, *Y* – координаты центра робота в Декартовой системе координат,

M – угол поворота робота, то есть его направление,

n – обозначение текущего шага,

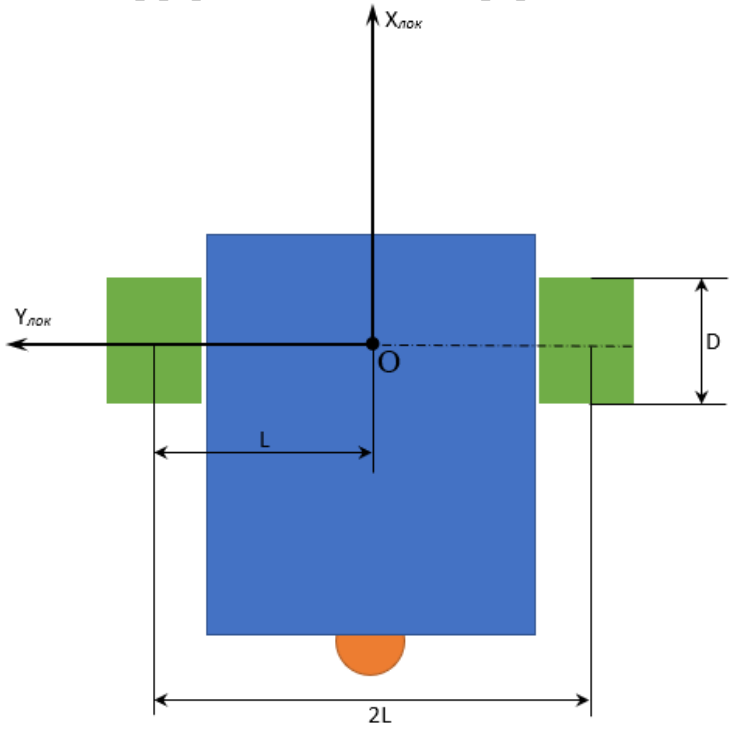
n-1 – обозначение предыдущего шага,
Δ – «приращение», изменение координаты за текущий шаг.
Так как все эти данные поступают в программу как входные, то задача сводится к многократному аккуратному вычислению по формулам.
Итоговая программа на ЯП Python может иметь следующий вид:

```
from sys import stdin
import sys
import math

QT=int(input())
X=0
Y=0
M=0

for i in range(QT):
    dt, dS, dM=map(float,input().split())
    dX = dS*math.cos(M+dM)
    dY = dS*math.sin(M+dM)
    X = X + dX
    Y = Y + dY
    M = M + dM

print(float( '{:.4f}'.format(X) ))
print(float( '{:.4f}'.format(Y) ))
```

Задача D. Одометрия дифф. платформы

Ограничения: по времени 1 сек, по памяти 256 Мб.

Описание робота: дифференциальная платформа.



D – диаметр колес робота

L – полуколея робота, расстояние от геометрического центра робота до центра колеса.

Точка *O* – геометрический центр робота, начало локальной системы координат робота. Точка является серединой оси, соединяющей колеса.

Задание:

Робот начинает движение в точке с нулевыми координатами и совершает сложное движение. Каждые *dt* секунд фиксируется показания энкодеров обоих моторов робота. По известным на каждом шаге движения изменению времени *dt* и показаниям энкодеров необходимо определить конечные координаты робота.

Формат входных данных

Первая строка содержит три разделенных пробелами числа, описывающих параметры привода робота:

Первое число (дробное, положительное, от 0.0001 до 1.0 с точностью до 4 знаков после запятой) – диаметр колес робота (*D*) в метрах.

Второе число (дробное, положительное, от 0.0001 до 1000.0 с точностью до 4 знаков после запятой) – редукция, передаточное число между мотором и колесом (*i*). Показывает, во сколько раз колесо вращается медленнее чем мотор.

Третье число (целое, положительное, четное, от 2 до 1440) – количество «тиков» энкодера на один оборот мотора (*n*).

Вторая строка содержит дробное положительное число от 0.0001 до 1.0 (с точностью до 4 знаков после запятой) – расстояние от центра робота до центра колеса (L) в метрах. «Половина колеи робота».

Третья строка содержит целое число от 1 до 6 * 10^4 – количество измерений (QT).

Следующие QT строк содержат по три разделенных пробелами числа:

Первое число (дробное, положительное, от 0.0001 до 1.0 с точностью до 4 знаков после запятой) – dt текущего шага, в секундах.

Второе число (целое, от -10^6 до 10^6) – показания энкодера левого мотора на текущем шаге (N_{left}).

Третье число (целое, от -10^6 до 10^6) – показания энкодера правого мотора на текущем шаге (N_{right}).

Формат выходных данных

Выведите две строки, содержащих дробные числа, с точностью до четырех знаков после запятой.

Первая строка должна содержать X-координату робота в момент окончания движения.

Вторая строка должна содержать Y-координату робота в момент окончания движения.

Пример 1

Входные данные:	Выходные данные:
5.0 1.0 1 1.0 1 1.0 1 -1	0.0000 0.0000

Пример 2

Входные данные:	Выходные данные:
0.1571 738.7856 360 0.2985 5 0.4176 659 711 0.9776 1364 1688 0.7325 1915 2852 0.2752 2517 3466 0.3579 2943 3779	0.0062 0.0000

Пример 3

Входные данные:	Выходные данные:
0.7456 177.6791 1056	0.0512
0.7913	0.0002
10	
0.4560 399 740	
0.0605 1001 1205	
0.7352 1316 1377	
0.1687 1327 1387	
0.1482 1699 2059	
0.0264 1980 2236	
0.5002 2455 2710	
0.3754 2688 3231	
0.8237 3325 4228	
0.1459 3607 4595	

Решение:

Одометрия дифференциальной платформы описана в видео «Дифференциальная платформа роботов». В текущей задаче можно воспользоваться полными формулами из видео или упрощенными, предполагая, что при малых «шагах» (перемещениях робота) длина хорды примерно равна длине дуги. В этом случае формулы примут вид:

$$\begin{cases} X_n \\ Y_n \\ M_n \end{cases} = \begin{cases} X_{n-1} + \Delta X \\ Y_{n-1} + \Delta Y \\ M_{n-1} + \Delta M \end{cases} = \begin{cases} X_{n-1} + \dfrac{\Delta S_{\text{прав}} + \Delta S_{\text{лев}}}{2} \cos\left(M_{n-1} + \dfrac{\Delta S_{\text{прав}} - \Delta S_{\text{лев}}}{4L}\right) \\ Y_{n-1} + \dfrac{\Delta S_{\text{прав}} + \Delta S_{\text{лев}}}{2} \sin\left(M_{n-1} + \dfrac{\Delta S_{\text{прав}} - \Delta S_{\text{лев}}}{4L}\right) \\ M_{n-1} + \dfrac{\Delta S_{\text{прав}} - \Delta S_{\text{лев}}}{2L} \end{cases}$$

Итоговая программа на ЯП Python может иметь следующий вид:

```
from sys import stdin
import sys
import math

D,i,n=map(float,input().split())
n=int(n)
L=float(input())
QT=int(input())

X=0
Y=0
M=0
Nleft_old=0
Nright_old=0

for stp in range(QT):
    dt, Nleft, Nright=map(float,input().split())
    Nleft=int(Nleft)
```

```
    Nright=int(Nright)
    dSleft = math.pi * D *(Nleft - Nleft_old)/(n*i)
    dSright = math.pi * D *(Nright - Nright_old)/(n*i)
    X = X+(dSright+dSleft)/2*math.cos( M + (dSright-dSleft)/(4*L) )
    Y = Y+(dSright+dSleft)/2*math.sin( M + (dSright-dSleft)/(4*L) )
    M=M+(dSright-dSleft)/(2*L)
    Nleft_old = Nleft
    Nright_old = Nright

print(float( '{:.4f}'.format(X) ))
print(float( '{:.4f}'.format(Y) ))
```

Задача E. Движение по линии ТРИК

Задача:

В симуляторе TRIK Studio подготовить управляющую программу, перемещающую робота вдоль линии и останавливающую его на третьем встреченном перекрестке.

Управляющая программа проверяется на нескольких полях, имеющих разную конфигурацию поворотов, прямых участков и перекрестков. За каждое поле начисляется до 10% баллов. Поле засчитывается при совпадении следующих условий:

- центр робота находится в зоне 300x300 мм, центр которой совпадает с центром третьего по ходу движения робота перекрестка;
- скорости обоих моторов робота равны нулю.

Робот:

Дифференциальная платформа. Робот оснащен четырьмя датчиками отраженного света, направленными вниз:

К порту A1 подключен левый боковой датчик.

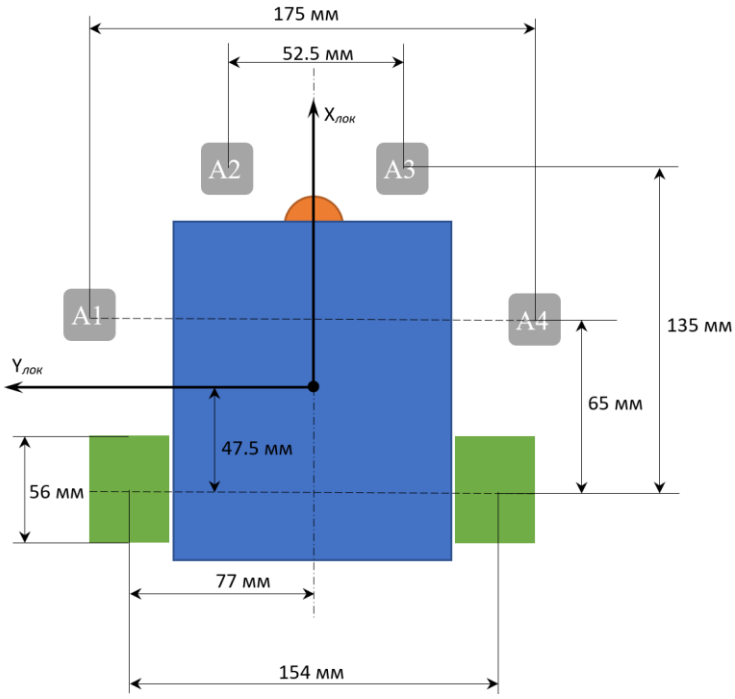
К порту A2 подключен левый передний датчик.

К порту A3 подключен правый передний датчик.

К порту A4 подключен правый боковой датчик.

Размеры модели в симуляторе TRIK Studio представлены на рисунке.

Описание робота ТРИК



Левый мотор робота подключен к порту M4, правый мотор - к порту M3.

Энкодеры робота возвращают угол поворота колес в градусах. Положительные числа обозначают вращение колеса, обеспечивающего движение робота вперед, отрицательные - назад.

Полигон:

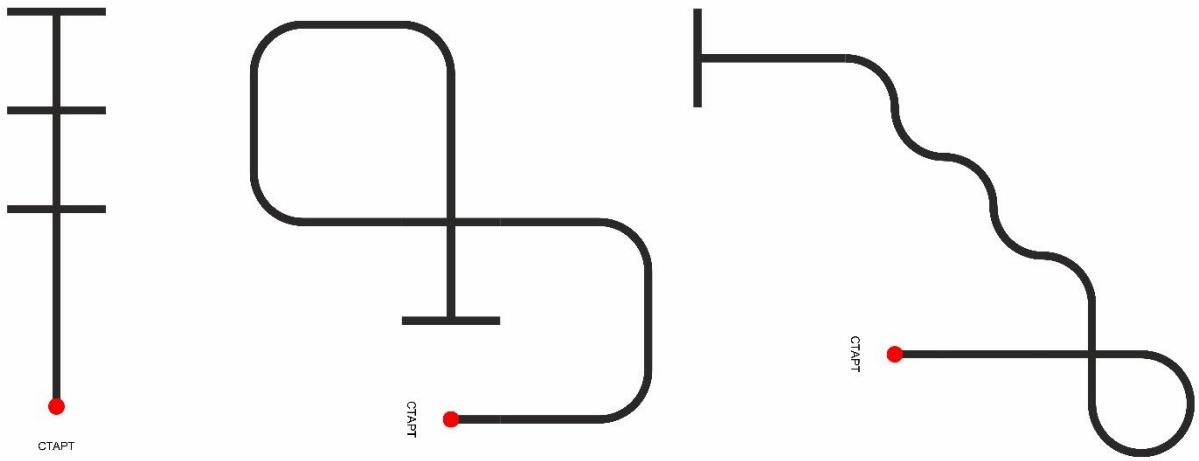
Общий размер полигона составляет 4 х 4 метра.

Стартовая секция и стартовое положение робота - точно в центре полигона.

Ширина линии не менее 25 мм.

Гарантируется, что на старте линия расположена между датчиками A2 и A3. Гарантируется. что первые 300 мм линии - прямой участок. Далее следуют повороты с радиусом не менее 150 мм. Гарантируется, что боковые линии на перекрестках отстоят от основной линии не менее чем на 125 мм и имеют ширину не менее 25 мм.

Пример полей:



Примеры заданий и тестовый проект:

В приложенном архиве находится следующие файлы:

- *Test_field_1_exercise.qrs* / *Test_field_2_exercise.qrs* / *Test_field_3_exercise.qrs* – файлы-упражнения с настроенными тренировочными полями;
- *task_7.py* - файл-программа с исходным кодом участника.
- Файлы-упражнения могут использоваться для отладки программ. После отладки код программы необходимо перенести в файл *task_7.py* и отправить в качестве решения задачи.

Данные: solutions.zip

Решение:

См. решение задачи G «Одометрия дифф. платформы ТРИК» для возрастной группы 9–11 класс, оно подходит.

Задача F. Граф

Ограничения: по времени 3 сек, по памяти 256 Мб.

Задание:

Вам дан неориентированный граф с n вершинами и m ребрами. Вершины пронумерованы от 1 до n.

Вам надо найти количество кратчайших путей из 1 до n. Так как ответ может быть большим, выведите его по модулю $10^9 + 7$.

Система оценки

Каждая группа тестов будет оцениваться отдельно и баллы начисляются в случае, если все тесты группы пройдены. Все тесты разбиты на группы со следующими ограничениями:

Подзадача	Ограничения	Баллы	Необходимые подзадачи
1	*$1 \le n \le 10$*	*30*	–
2	*$1 \le n \le 1000$*	*30*	1
3	–	*40*	1, 2

Формат входных данных

В первой строке записано два целых числа n и m ($2 \le n \le 2*10^5$ и $0 \le m \le 2*10^5$).

Следующие m строк содержат описание рёбер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i – номерами концов ребра ($1 \le b_i, e_i \le n$).

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число – количество кратчайших путей из 1 до n. Если нет пути из 1 в n, то выведите 0.

Пример 1

Входные данные:	Выходные данные:
3 1 2 1 2 3	0

Пример 2

Входные данные:	Выходные данные:
2 1 1 2	1

Пример 3

Входные данные:	Выходные данные:
4 3 1 3 2 3 2 4	1

Входные данные:	Выходные данные:
7 8 1 3 1 4 2 3 2 4 2 5 2 6 5 7 6 7	4

Решение:

Для подсчета кратчайшего расстояния от вершины 1 до вершины *n*, воспользуемся алгоритмом BFS (алгоритм обхода в ширину) за время $O(N + M)$.

Во время работы алгоритма BFS будем вычислять с помощью динамического программирования количество кратчайших путей.

Когда находим кратчайшее расстояние, мы заполняем массив кратчайших расстояний *dist* следующим образом:

Для каждой вершины *t*, у которой есть ребро от *v*:

• если мы еще не посещали *t*, обновляем *dist[t]* на *dist[v]* + 1;

• в противном случае ничего не делаем.

Одновременно будем обновлять массив количества кратчайших путей *dp* следующим образом:

Для каждой вершины *t*, у которой есть ребро от *v*:

• если мы еще не посещали *t*, обновляем *dist[t]* на *dist[v]* + 1, и к *dp[t]* присваиваем *dp[v]*;

• если мы посещали *t* и *dist[t]* == *dist[v]* + 1, то к *dp[t]* добавляем *dp[v]*;

• в противном случае ничего не делаем.

Задача G. Одометрия дифф. платформы ТРИК

Задача:

В симуляторе TRIK Studio подготовить управляющую программу, перемещающую робота вдоль линии и останавливающую его на третьем встреченном перекрестке, посчитав при этом финишные координаты робота.

Управляющая программа проверяется на нескольких полях, имеющих разную конфигурацию поворотов, прямых участков и перекрестков. За каждое поле начисляется до 10% баллов. Поле засчитывается при совпадении следующих условий:

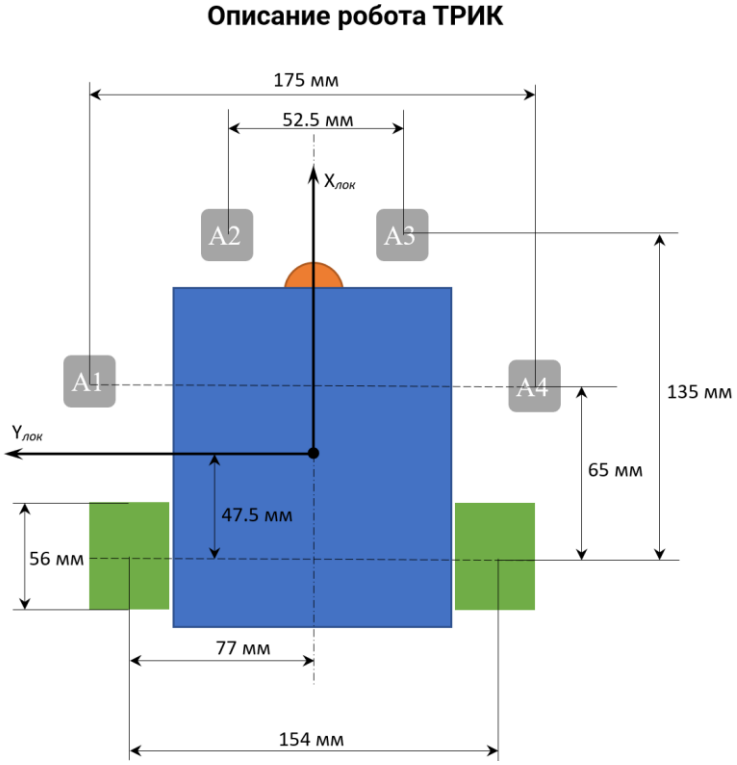
5 баллов начисляются при совпадении условий:

- центр робота находится в зоне 300x300 мм, центр которой совпадает с центром третьего по ходу движения робота перекрестка;
- скорости обоих моторов робота равны нулю.

5 баллов начисляются при совпадении условий:

- перед завершением программы робот вывел на экран не менее двух сообщений в разных строках,
- первое и последнее выведенные на экран робота сообщения содержат целые числа,
- первое выведенное число отличается от эталонной координаты X финишной зоны не более чем на 100 (ошибка вычисления координаты X не превышает 100 мм),
- последнее выведенное число отличается от эталонной координаты Y финишной зоны не более чем на 100 (ошибка вычисления координаты Y не превышает 100 мм).

Робот:



Дифференциальная платформа. Робот оснащен четырьмя датч отраженного света, направленными вниз:

К порту A1 подключен левый боковой датчик.

К порту A2 подключен левый передний датчик.

К порту A3 подключен правый передний датчик.

К порту A4 подключен правый боковой датчик.

Размеры модели в симуляторе TRIK Studio представлены на рисунке.

Левый мотор робота подключен к порту M4, правый мотор - к порту M3.

Энкодеры робота возвращают угол поворота колес в градусах. Положительные числа обозначают вращение колеса, обеспечивающего движение робота вперед, отрицательные - назад.

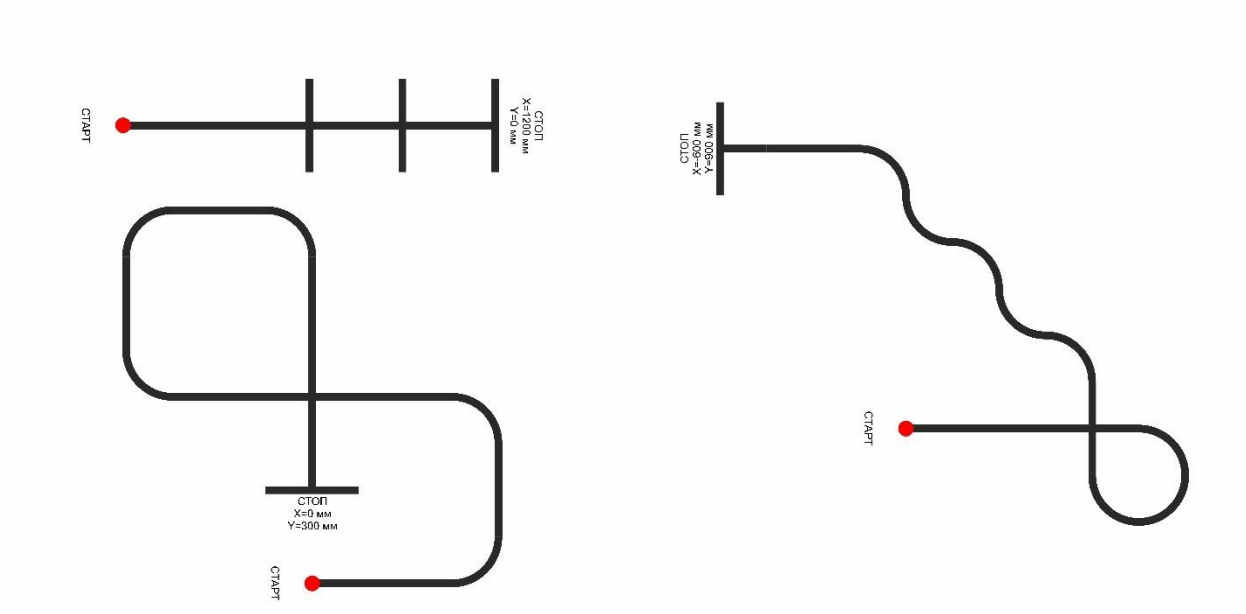
Полигон:

Общий размер полигона составляет 4 x 4 метра.

Стартовая секция и стартовое положение робота - точно в центре полигона.

Ширина линии не менее 25 мм.

Гарантируется, что на старте линия расположена между датчиками A2 и A3. Гарантируется. что первые 300 мм линии - прямой участок. Далее следуют повороты с радиусом не менее 150 мм. Гарантируется, что боковые линии на перекрестках отстоят от основной линии не менее чем на 125 мм и имеют ширину не менее 25 мм.



Примеры заданий и тестовый проект:

В приложенном архиве находится следующие файлы:

- *Test_field_1_exercise.qrs* / *Test_field_2_exercise.qrs* / *Test_field_3_exercise.qrs* – файлы упражнения с настроенными тренировочными полями;
- *task_15.py* – файл программа с исходным кодом участника.

Файлы-упражнения могут использоваться для отладки программ. После отладки код программы необходимо перенести в файл *task_15.py* и отправить в качестве решения задачи.

Данные: solutions.zip

Решение:

Движение по линии может быть реализовано с помощью алгоритма ПИД-регулятора. Для текущей задачи алгоритм можно редуцировать до ПИ- или ПД-регулятора, снизив общую скорость движения робота.

Для вычисления одометрии используются упрощенные формулы из задачи D «Одометрия дифф. платформы».

Итоговая программа на ЯП Python может иметь следующий вид:

```
import sys
import time
import random
import math

class Program():
  __interpretation_started_timestamp__ = time.time() * 1000
  pi = 3.141592653589793
  def execMain(self):
    Diam=0.056 #диаметр колес робота, в метрах
    L=0.077 #полуколея робота, в метрах
    X=0.0 #координата X робота, в метрах
    Y=0.0 #координата Y робота, в метрах
    kf=0.9
    M=0
    kp=1.5
    ki=0
    kd=0
    err=0
    errold=0
    I=0
    n=0
    leftSpd = 0
    rightSpd= 0
    Nleft_old=0
    Nright_old=0
    while n<3:
      s1 = brick.sensor("A1").read()
      s2 = brick.sensor("A2").read()
      s3 = brick.sensor("A3").read()
      s4 = brick.sensor("A4").read()
      s1f = s1
      s4f = s4
      while (s1+s4)<50:
        s1 = brick.sensor("A1").read()
        s2 = brick.sensor("A2").read()
        s3 = brick.sensor("A3").read()
        s4 = brick.sensor("A4").read()
        s1f = kf*s1f+(1-kf)*s1
        s4f = kf*s4f+(1-kf)*s4
        err = s2 - s3
        P=kp*err
        I=I+ki*err
        D=kd*(err-errold)
        U=P+I+D
        leftSpd = 50-U
```

```
            rightSpd= 50+U
            brick.motor("M3").setPower(rightSpd)
            brick.motor("M4").setPower(leftSpd)
            errold = err
            Nleft=brick.encoder("M4").read()
            Nright=brick.encoder("M3").read()
            dSleft = math.pi * Diam *(Nleft - Nleft_old)/360
            dSright = math.pi * Diam *(Nright - Nright_old)/360
            X = X+(dSright+dSleft)/2*math.cos( M + (dSright-dSleft)/(4*L) )
            Y = Y+(dSright+dSleft)/2*math.sin( M + (dSright-dSleft)/(4*L) )
            M=M+(dSright-dSleft)/(2*L)
            Nleft_old = Nleft
            Nright_old = Nright
            script.wait(10)
          n=n+1
          startTime = script.time()
          brick.playTone(200, 300)
          dt = 0
          while dt<800:
            s1 = brick.sensor("A1").read()
            s2 = brick.sensor("A2").read()
            s3 = brick.sensor("A3").read()
            s4 = brick.sensor("A4").read()
            s1f = kf*s1f+(1-kf)*s1
            s4f = kf*s4f+(1-kf)*s4
            err = s2 - s3
            P=kp*err
            I=I+ki*err
            D=kd*(err-errold)
            U=P+I+D
            leftSpd = 40-U
            rightSpd= 40+U
            brick.motor("M3").setPower(rightSpd)
            brick.motor("M4").setPower(leftSpd)
            errold = err
            Nleft=brick.encoder("M4").read()
            Nright=brick.encoder("M3").read()
            dSleft = math.pi * Diam *(Nleft - Nleft_old)/360
            dSright = math.pi * Diam *(Nright - Nright_old)/360
            X = X+(dSright+dSleft)/2*math.cos( M + (dSright-dSleft)/(4*L) )
            Y = Y+(dSright+dSleft)/2*math.sin( M + (dSright-dSleft)/(4*L) )
            M=M+(dSright-dSleft)/(2*L)
            Nleft_old = Nleft
            Nright_old = Nright
            script.wait(10)
            dt=script.time()-startTime
        X=int(X*1000) #перевод координаты в целое число, в миллиметрах
        Y=int(Y*1000) #перевод координаты в целое число, в миллиметрах
        brick.display().addLabel(X, 1, 1, 20) #вывод в первую строку координаты X
        brick.display().redraw()
        brick.display().addLabel(Y, 1, 25,20)#вывод во вторую строку координаты Y
        brick.display().redraw()
        brick.motor("M3").powerOff()
        brick.motor("M4").powerOff()
        script.wait(10)
        brick.stop()
        return

def main():
  program = Program()
  program.execMain()

if __name__ == '__main__':
  main()
```

Задача H. Фигуры

Ограничения: по времени 8 сек, по памяти 256 Мб.

Задание:

Камера робота сделала один черно-белый кадр размером W x H. На кадре зафиксировано от 1 до 30 объектов разных форм (круг, треугольник, квадрат). Необходимо определить количество квадратов, стороны которых параллельны осям координат, повернутых квадратов, кругов и треугольников.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необх. подзадачи
0	0	Тесты из условия	–
1	10	только квадраты, стороны которых параллельны осям координат	–
2	20	только квадраты (могут быть повернуты)	–
3	10	только круги	–
4	15	только треугольники	–
5	45	"–	1-4

Формат входных данных

В первой строке входных данных содержит два натуральных числа W и H ($30 \le W, H \le 500$) – размеры кадра.

В следующих H строках содержится по W символов $a_{i,j} \in \{0, 1\}$ – если 0, то фон, 1 объект.

Гарантируется, что объект состоит не менее чем из 25 единиц.

Формат выходных данных

Выведите через пробел 4 числа – количество квадратов, стороны которых параллельны осям координат, повернутых квадратов, кругов и треугольников.

Пример 1

Входные данные:

```
40 40
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000100000000000000000000000000000
0000000001100000000000000000000000000000
0000000011110000000000000000010000000000
0000000111110000000000000000011000000000
0000001111111000000000000000111100000000
0000011111111000000000000000111111000000
0000000000001000000000000001111111100000
0000000000000000000000000011111111111000
0000000000000000000000000111111111110000
0000000000000000000000001111111111100000
0000000000000000000000011111111111000000
0000000000000000000000111111111110000000
0000000000000000010000000000011110000000
0000000000000001111111000000001100000000
0000000000000011111111111000000100000000
0000000000000111111111111000000000000000
0000000000000111111111111000000000000000
0000000000001111111111111000000000000000
0000000000001111111111111000000000000000
0000000000001111111111111000000000000000
0000000000001111111111111000000000000000
0000000000001111111111111000000000000000
0000000000001111111111111000000000000000
0000111110000111111111111000000000000000
0000111110000111111111111000000000000000
0000111110000111111111111000000000000000
0000111110000000111111100000000000000000
0000111110000000001000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
```

Выходные данные:

```
1 1 1 1
```

Решение:

Разделим решение на два этапа:

1. Выделение объектов: для выделения объектов можно использовать рекурсивный обход (DFS) или BFS. Будем находить первый символ (который не был выделен) и запустим от этого символа DFS. Во время обработки выпишем позиции всех клеток этого объекта для того, чтобы создать дополнительную матрицу, только с этим объектом для дальнейшей обработки.

2. Классификация объектов:

• Для проверки на не повернутый квадрат достаточно проверить, что дополнительная матрица является квадратной и нет клеток фона.

• Для проверки на круг восстановим уравнение окружности, используя координаты центра (x_c, y_c) и радиус r, проверим, что все неповернутые точки удовлетворяют неравенству

$$(x - x_c)^2 + (y - y_c)^2 \leq r^2$$

• Для проверки на повернутый квадрат посчитаем количество компонент, состоящих только из 0. Их должно быть строго 4 (смотрите рисунок), и объект не должен удовлетворять условию, что не выполняется условие окружности.

• Для проверки на треугольник не должны выполняться условия для квадрата и окружности.