Работа победителя заключительного этапа

командной инженерной олимпиады школьников

Олимпиада Национальной технологической инициативы

Профиль «Интеллектуальные робототехнические системы»

Макаренко Илья Сергеевич

Класс: 10

Город: Железногорск

Школа: КГАОУ «Школа космонавтики»

Регион: Красноярский край

Уникальный номер участника: 272

Команда на заключительном этапе: Team STFU

Параллель: 10-11

Результаты заключительного этапа:

Индивидуальная часть													Командная часть					Результат
№	Математика			Информатика							Итого	Макс. балл	Решение задач этапов				Итого	
	1	2	3	1.1	1.2	1.3	2.1	2.2	3.1	3.2			№1	№2	№3	№4		
272	8	17	0	6	9	0	0	0	0	0	40	200	12	38	12	24	86	126

Индивидуальная часть

Персональный лист участника с номером 272:

Олимпиада НТИ

ФИО Макаренко Илья Сергеевич

Город Боготол

Школа № КГАОУ „Школа космонавтики"

Математика

Лист 1:

Командная инженерная олимпиада «Олимпиада НТИ»

Направление UPL

Предмет Математика

Номер участника 282

1	2а	2б	3а	3б	3в	Σ
8	7	10	0	0	0	25

№1

Рассмотрим кол-во молока в стаканах после нескольких первых переливаний и найдём закономерность:

№ ст	I	II
0	0	100
1.	0	100
2.	50	50
3.	25	75
4.	62,5	37,5
5.	31,25	68,75
6.	65,625	34,375

+

Посмотрим изменение первого стакана по чётным индексам:

+50; +12,5; +3,125 ...

Это геометрическая прогрессия

$b_n = 50 \cdot 0,25^{n-1}$, $n \in$ +

Сколько кофе должно быть в каждом стакане, если концентрация будет равна?

200 (Всего кофе) = 100 (мл в первом ст.) · k + 100 (мл. в II ст) · k

$200 = 200 \cdot k + 100 \cdot k$

$k = \frac{2}{3}$ – концентрация кофе.

$\frac{1}{3}$ – концентрация молока =>

В первом стакане должно быть $200 \cdot \frac{1}{3} = 66,(6)$ мл. молока.

=> нам требуется действие, в результате которого в I ст. будет

> 6.5,(6) мл. молока. ~~66,6616+625~~ 5 · 2 = 10

1) 50 + 0 = 50

2) 50 + 12,5 = 62,5

3) 62,5 + 3,125 = 65,625

4) 65,625 + 0,78125 = 66,40625

5) 66,40625 + 0,1953125 = 66,6016625 – результат 2

6) 66,6016625 + 0,0488281125 = 66,6494906 25 –

7) 66,64949062 5 + 0,0122070 3125 = 66,66161765625

Ответ: 10

На 8 переливании.

→ + (8б)

Лист 2:

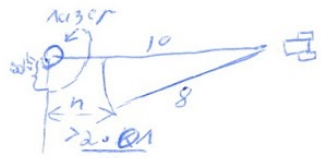
Командная инженерная олимпиада «Олимпиада НТИ»

Направление UPC

Предмет Математика

Номер участника 272

№2

а) 

ол - кругов лазера после полного оборота.

Чтобы успеть приехать в точку, робот должен попасть в окружность (с радиусом $\frac{8}{2\pi}$), внутри которой он может двигаться по окружности быстрее лазера.

Рассмотрим расстояние n до некоторой прямой, выходящей из точки центра и не пересекающей путь робота в первый оборот лазера. оно всегда больше ~~чем равно~~ чем $\frac{2 \cdot ол}{ч}$)

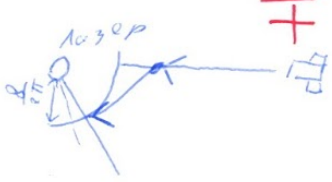
и следовательно, у робота нет шансов попасть в окружность $\frac{8}{2\pi}$. (А почему ему обязательно нужно попасть в этот круг? может он будет двигаться как-то по другому?)

1. И может быть равно $\frac{2 \cdot ол}{ч}$ при скорости робота в 8 клеток при движении к центру под углом 60°, но в условии скорость строго меньше.

Что и требовалось доказать.

б) Да, может. Пример описан в примечании 1: рисунок

(Да, можно так сделать. Но я не нашел объяснение этого?)



После попадания в окружность, робот спокойно добирается до центра, крутясь вокруг него и постепенно приближаясь.

Примечание: при решении я пренебрегал размерами робота, а скорость считал постоянной и не зависящей от каких-либо факторов.

Оценка за задачу №1: 8

Комментарий к решению: Рассуждения верные, но ответ дан неверный.

Оценка за задачу №2: 17

Комментарий к решению: Задача решена не полностью.

Оценка за задачу №3: 0

Комментарий к решению: Решение не предложено.

Информатика

Задача №1

Задача решена для первых двух групп тестов (15 баллов).

Код программы на языке C++, написанный участником и решающий задачу:

```cpp
// task1.cpp: определяет точку входа для консольного приложения.
//

#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;

int main() {
	int n, m, k;
	cin >> n >> m >> k;
	int h[100001], h1[100001];

	for (int i = 0; i < n; i++) {
		cin >> h[i];
		h1[i] = h[i];
	}

	int sum1 = 0, sum2 = 0;
	if (k <= m) {
		sum1 = h[k-1];
	}
	else {

		for (int i = m; i < k - 1; i++) {
			sort(h1+1, h1 + i + 1);
			sum1 += h1[1];
			h1[1] = 1000000;
		}
		sum1 += h1[k - 1];
	}
	if (k - 1 >= n - m) {
		sum2 = h[k-1];
	}
	else {
```

```
            for (int i = n - m; i > k - 1; i--) {
                  sort(h + i, h + n);
                  sum2 += h[i];
                  h[i] = 1000000;
            }
            sum2 += h[k - 1];
      }
      if (sum1 == 0)
            cout << sum2;
      else if (sum2 == 0)
            cout << sum1;
      else
            cout << min(sum1, sum2);
      return 0;
}
```

Задача №2

Задача не решена (0 баллов).

Код программы на языке C++, предложенный участником:

```
#include <iostream>
#include <algorithm>

using namespace std;

int a[100001];
int res(int n) {
      if (n == 0)
            return a[0] ^ a[1];
      if (n == 1)
            return max(a[0] ^ a[1] + a[2] ^ a[3],
                        a[0] ^ a[2] + a[1] ^ a[3]);
      return max(res(n-1)  + a[n*2]^a[n*2+1],
                  res(n-2) + a[n * 2 - 2] ^ a[n * 2] +
                      a[n * 2 - 1] ^ a[n * 2 + 1]);
}

int main()
{
      int n;
      cin >> n;
      for (int i = 0; i < n; i++) {
```

```
            cin >> a[i];
        }

        cout << res(n/2-1);
    return 0;
}
```

Ошибка, которую вернула система тестирования:

```
Failed test #1. Wrong answer
Input:
4
1 1 2 6

Your output:
4
Correct output:
10
```

Задача №3

Решение не было предложено (0 баллов).

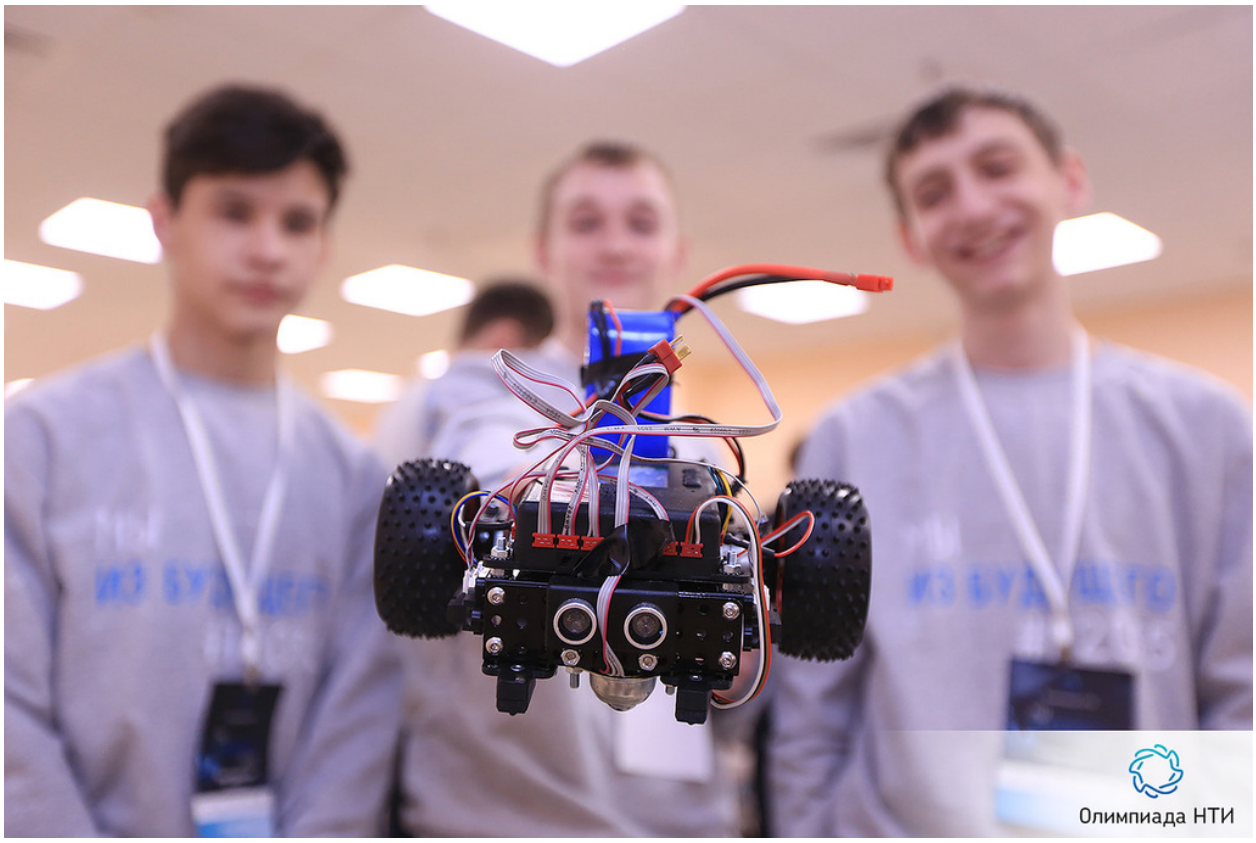
Командная часть

Результаты были получены в рамках выступления команды: Team STFU

Личный состав команды:

- Макаренко Илья Сергеевич
- Фронц Даниил Владимирович
- Чуруксаев Иван

Фотография команды и робототехнического устройства, оснащенного датчиками, перед проверочными испытаниями второго этапа:



Протокол выполнения заданий:

Первый этап:

№ задания	Критерий	Возможные баллы	Полученные баллы
1	Робот покинул сектор старта	6x2	12
2	Робот проехал половину от всего количества секторов от сектора старта до сектора финиша	13x2	0
3	Робот проехал от сектора старта до черной линии (в ходе движения робот пересек черную линию или остановился таким образом, что черная линия пересекает проекцию робота на поле)	6x2	0
4	Робот проехал от сектора старта, заехал в сектор финиша (проекция робота на поле внутри сектора) и остановился	6x2	0
5	Робот проехал от сектора старта, остановился в секции финиша (проекция робота внутри сектора) и вывел на экран верное количество пройденных секций	12x2	0
Дополнительные баллы		19	0
Всего за этап		105	12

Второй этап:

№ задания	Критерий	Возможные баллы	Полученные баллы
6	Робот проехал по периметру вокруг циклической структуры и остановился после определения цикла (допускается проезд по второму кругу до 3х секторов)	7x2	7
7	Робот после верного определения циклической структуры, остановился на 3 секунды, и продолжил движение по другим секторам полигона, не принадлежащим циклической структуре (движение не менее, чем по 5 не принадлежащим циклической структуре)	12x2	0
8	Робот проехал не меньше 7 секторов по полигону и остановился; на экране отображена отображены координаты смещения относительно сектора старта: первое число – смещение по оси X, второе число – смещение по оси Y. При этом ось X совпадает с направлением робота в секторе старта, а ось Y направлена вправо. Данный критерий не будет оцениваться, если робот способен выполнять сразу следующий пункт списка критериев.	6x2	0
9	После остановки и отображения карты на экране устройства отображается 2 числа, определяющее позицию, где произошла остановка: первое число – сектор по оси X, второе число – сектор по оси Y (см. рис. 5)	31x2	31
Дополнительные баллы		19	0
Всего за этап		119	38

Третий этап:

№ задания	Критерий	Возможные баллы	Полученные баллы
10	Робот доехал до сектора сервисного обслуживания (проекция робота внутри сектора) и остановился	6x2	12
11	После остановки робота на экран выведено закодированное значение.	19x2	0
Дополнительные баллы		12	0
Всего за этап		62	12

Четвертый этап:

№ задания	Критерий	Возможные баллы	Полученные баллы

12	После запуска программы, роботе стоит на месте в течение на 10 секунд. На экран выведен путь от сектора старта до сектора финиша. При этом путь выведен в виде координат секций, через которые должен пройти робот. Данный критерий не будет оцениваться, если робот способен выполнять сразу следующий пункт списка критериев.	6x2	0
13	После запуска программы, роботе стоит на месте в течение на 10 секунд. На экран выведен путь от сектора старта до сектора финиша. При этом путь выведен в виде виде алгоритма перемещения (F - одна секция прямо, R- поворот направо, L - поворот налево) с учетом необходимых поворотов в секторе старта.	9x2	18
14	Путь, выведенный после запуска робота, является оптимальным по количеству секторов, необходимых для посещения.	3x2	6
15	Путь перемещения робота из сектора старта до сектора финиша является оптимальным по количеству секторов.	3x2	0
16	Робот доехал до финального сектора и остановился. И издал звуковой сигнал.	6x2	0
Дополнительные баллы		12	0
17	Робот выехал из сектора старта и остановился на 10 секунд в первом (по ходу движения робота) секторе сервисного обслуживания. Перед началом отсчета таймера робот подает звуковой сигнал.	6	0
18	После остановки робота в секторе сервисного обслуживания выведена одна компонента координаты финального сектора вместе с идентификатором оси (буква X или Y). Выведенное число совпадает со штрих-кодом нанесенным в данном секторе сервисного обслуживания.	5	0
19	Робот продолжил движение и остановился на 10 секунд во втором секторе сервисного обслуживания. Перед началом отсчета таймера робот подает звуковой сигнал.	11	0
20	После остановки робота в секторе сервисного обслуживания выведены обе компоненты координаты финального сектора вместе с идентификаторами осей (буква X или Y). Координата совпадает со штрих-кодом нанесенным в данном секторе сервисного обслуживания.	5	0
21	Путь перемещения робота из сектора, где робот локализовался до очередного сектора сервисного обслуживания или финального сектора является оптимальным по количеству секторов.	5	0
22	Робот доехал до финального сектора и остановился. И издал звуковой сигнал.	28	0
Всего за этап		114	24

Программа команды для решения задачи первого этапа:

```
/*
SI:
      range - cm
      angles - radians

*/
// Const block
var pi = 3.141592653589793; // Const value
var cellLen = 40; // Lenght of cell
var cpm = 374; // Raw dara of encoders per one rotation
var robotLen = 22;
var wheelD = 7.6;
var speed = 40;

// Objects' block

function eLeft() {
  return -brick.encoder(E4).readRawData();
}
var eRight = brick.encoder(E3).readRawData;

var mLeft = brick.motor(M4).setPower;
var mRight = brick.motor(M3).setPower;

var lLeft = brick.sensor(A1).read;
var lRight = brick.sensor(A2).read;

var irLeft = brick.sensor(A6).read;
var irRight = brick.sensor(A5).read;
var gyro = brick.gyroscope();

var US = brick.sensor(D2).read;

// Functions

var lightsL = [];
var lightsR = [];
var i_Timer = 0;
var lightTimer = script.timer(10);
lightTimer.timeout.connect(
  function() {
```

```
    lightsL[i_Timer] = lLeft();
    lightsR[i_Timer] = lRight();
    i_Timer++;
  }
);

function turn(angle) {
  eClean();
  var l = Math.abs(angle / 180 * robotLen * pi / 2);
  var limit = l / (pi * wheelD);
  var sgn = angle / Math.abs(angle);
  mLeft(angle * sgn);
  mRight(-angle * sgn);
  var el = Math.abs(eLeft());
  var er = Math.abs(eRight());
  while (el < limit) {
    el = Math.abs(eLeft() / 360);
    script.wait(1);
  }
}

function stabilization(time) {
  print("stabilization");
  var k = 1.2;
  for (var i = 0; i < time; i++) {
    print(gyro.read()[6] / 1000);
    if (gyro.read()[6] / 1000 < 45 && gyro.read()[6] / 1000 > -45) {
      var err = gyro.read()[6] / 1000;
      mLeft(err * k);
      mRight(-err * k);
    }
    if (gyro.read()[6] / 1000 < 135 && gyro.read()[6] / 1000 > 45) {
      var err = gyro.read()[6] / 1000 - 90;
      mLeft(err * k);
      mRight(-err * k);
    }
    if (gyro.read()[6] / 1000 > -135 && gyro.read()[6] / 1000 < -45)
{
      var err = gyro.read()[6] / 1000 + 90;
      mLeft(err * k);
      mRight(-err * k);
    }
    if (gyro.read()[6] / 1000 < -135 || gyro.read()[6] / 1000 > 135)
```

```
{
      var err;
      if (gyro.read()[4] / 1000 < -135)
        err = -(-180 - gyro.read()[6] / 1000);
      else
        err = -(180 - gyro.read()[6] / 1000);
      mLeft(err * k);
      mRight(-err * k);
    }
    script.wait(1);
  }
  mLeft(0);
  mRight(0);
}

function eClean() {
  brick.encoder(E3).reset();
  brick.encoder(E4).reset();
}

function rotate(c) {

  var eLeftNow = eLeft();
  var eRightNow = eRight();

  if (c == 'L') {
    while (((eRight() - eRightNow) - (eLeft() - eLeftNow)) * wheelD /
      cpm <= robotLen / 3) {
      mLeft(-speed);
      mRight(speed);
      script.wait(1);
    }
  }
  if (c == 'R') {
    while (((eRight() - eRightNow) + (eLeft() - eLeftNow)) * wheelD /
      cpm <= robotLen / 3) {
      mRight(-speed);
      mLeft(speed);
      script.wait(1);
    }
  }
}
```

```
function move(c) {
  var eLeftNow = eLeft();
  var eRightNow = eRight();

  if (c == 'F') {
    while (((eRight() - eRightNow) + (eLeft() - eLeftNow)) * pi *
      wheelD / (2 * cpm) <= cellLen - 10) {
      k = 0.25;
      var err = (65 - irRight()) * k;
      if (irRight() < 25)
        err = 0;
      mLeft(speed + err);
      mRight(speed - err);
      script.wait(1);
    }
  }
}

var last_i = 0;

function check() {
  var isL = false;
  var isR = false;
  for (var i = i_Timer; i >= last_i; i--) {
    if (lightsL[i] < 20)
      isL = true;
    if (lightsR[i] < 20)
      isR = true;
  }
  last_i = i_Timer;
  return isL && isR;
}

function main() {
  eClean();

  var cells = 0;
  while (true) {
    brick.motor(M3).forceBreak();
    brick.motor(M4).forceBreak();
    if (check()) {
      brick.display().addLabel(cells, 10, 10);
      brick.display().redraw();
```

```
      break;
    }
    script.wait(1000);
    print(irLeft());
    if (irRight() < 20) {
      rotate('R');
      brick.motor(M3).forceBreak();
      brick.motor(M4).forceBreak();
      script.wait(1000);
      move('F');
      cells++;
      continue;
    }
    if (US() > 25 && irRight() > 35) {
      move('F');
      cells++;
      continue;
    }
    if (US() < 25 && irRight() > 25) {
      rotate('L');
      mLeft(-60);
      mRight(-60);
      script.wait(1000);
      mLeft(40);
      mRight(40);
      script.wait(500);
      continue;
    } else {
      turn(30);
      brick.motor(M3).forceBreak();
      brick.motor(M4).forceBreak();
      script.wait(1000);
      move('F');
      cells++;
      continue;
    }
  }

}


main();
brick.motor(M3).forceBreak();
```

```
brick.motor(M4).forceBreak();
script.wait(10000);
```

Программа команды для решения задач “определение циклической структуры” и “локализация” второго этапа:

```
// MAP
var map = [];
for (var i = 0; i <= 7; i++)
  map[i] = [];

map[0][0] = [
  [0, 1, 1, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];
map[0][1] = [
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1]
];
map[0][2] = [
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
];
map[0][3] = [
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 0],
  [0, 0, 0, 1]
];
map[0][4] = [
  [0, 1, 0, 1],
  [1, 0, 0, 1],
```

```
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
map[0][5] = [
  [0, 1, 1, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1]
];
map[0][6] = "NONE";
map[0][7] = [
  [0, 1, 1, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 0]
];

map[1][0] = [
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1]
];
map[1][1] = "NONE";
map[1][2] = [
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
map[1][3] = [
  [0, 1, 1, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
map[1][4] = "NONE";
```

```
map[1][5] = [
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1]
];
map[1][6] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
map[1][7] = [
  [0, 1, 1, 0],
  [1, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];

map[2][0] = [
  [0, 1, 1, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
map[2][1] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
map[2][2] = [
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
```

```
map[2][3] = [
  [0, 1, 0, 0],
  [1, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];
map[2][4] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 0]
];
map[2][5] = [
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [1, 1, 0, 1]
];
map[2][6] = "NONE";
map[2][7] = [
  [0, 1, 1, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0]
];

map[3][0] = [
  [0, 1, 1, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
map[3][1] = "NONE";
map[3][2] = [
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
```

```
  [1, 1, 0, 1]
];
map[3][3] = [
  [0, 1, 1, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0]
];
map[3][4] = "NONE";
map[3][5] = [
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
map[3][6] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1]
];
map[3][7] = [
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];

map[4][0] = [
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
map[4][1] = [
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
```

```
    [0, 0, 0, 0],
    [0, 1, 0, 1]
  ];
  map[4][2] = [
    [0, 0, 0, 0],
    [0, 1, 0, 1],
    [0, 0, 0, 1],
    [1, 1, 0, 0],
    [0, 0, 0, 0]
  ];
  map[4][3] = [
    [0, 1, 0, 1],
    [0, 0, 0, 1],
    [1, 1, 0, 0],
    [0, 0, 0, 0],
    [1, 1, 0, 1]
  ];
  map[4][4] = [
    [0, 1, 0, 0],
    [0, 1, 0, 1],
    [0, 0, 0, 0],
    [0, 1, 0, 1],
    [0, 1, 0, 0]
  ];
  map[4][5] = [
    [0, 1, 0, 0],
    [0, 1, 0, 1],
    [0, 0, 0, 1],
    [0, 1, 0, 1],
    [1, 0, 0, 0]
  ];
  map[4][6] = "NULL";
  map[4][7] = [
    [0, 1, 1, 1],
    [0, 1, 0, 0],
    [1, 1, 0, 1],
    [0, 0, 0, 1],
    [0, 1, 0, 1]
  ];

  map[5][0] = [
    [0, 1, 1, 1],
    [0, 0, 1, 1],
```

```
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
map[5][1] = "NULL";
map[5][2] = [
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [1, 1, 0, 0]
];
map[5][3] = "NULL";
map[5][4] = [
  [0, 1, 1, 0],
  [0, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
map[5][5] = [
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1]
];
map[5][6] = [
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
map[5][7] = [
  [0, 1, 1, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1]
];

map[6][0] = [
```

```
  [0, 1, 1, 0],
  [1, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
map[6][1] = [
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 0],
  [0, 1, 0, 1]
];
map[6][2] = [
  [0, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
map[6][3] = [
  [0, 1, 1, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
map[6][4] = "NULL";
map[6][5] = [
  [0, 1, 1, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [1, 0, 1, 0]
];
map[6][6] = "NULL";
map[6][7] = [
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
```

```
map[7][0] = [
  [0, 1, 1, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [1, 1, 0, 0]
];
map[7][1] = "NULL";
map[7][2] = [
  [0, 1, 1, 0],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
map[7][3] = [
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
];
map[7][4] = [
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1]
];
map[7][5] = [
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
];
map[7][6] = [
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 0]
```

```
];
map[7][7] = [
  [0, 1, 1, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];


// EoMAP

// ChooseMAP


var chooseMap = [];
for (var i = 0; i < 8; i++)
  chooseMap[i] = [];

chooseMap[0][0] = "2 2";
chooseMap[0][1] = "1 2";
chooseMap[0][2] = "0 2";
chooseMap[0][3] = "5 2";
chooseMap[0][4] = "4 2";
chooseMap[0][5] = "1 0";
chooseMap[0][6] = "NONE";
chooseMap[0][7] = "5 0";

chooseMap[1][0] = "2 3";
chooseMap[1][1] = "NONE";
chooseMap[1][2] = "0 1";
chooseMap[1][3] = "4 2";
chooseMap[1][4] = "NONE";
chooseMap[1][5] = "3 1";
chooseMap[1][6] = "2 0";
chooseMap[1][7] = "5 1";

chooseMap[2][0] = "2 4";
chooseMap[2][1] = "1 4";
chooseMap[2][2] = "4 0";
chooseMap[2][3] = "5 2";
chooseMap[2][4] = "4 4";
chooseMap[2][5] = "7 0";
```

```
chooseMap[2][6] = "NONE";
chooseMap[2][7] = "5 2";

chooseMap[3][0] = "1 4";
chooseMap[3][1] = "NONE";
chooseMap[3][2] = "0 3";
chooseMap[3][3] = "5 3";
chooseMap[3][4] = "NONE";
chooseMap[3][5] = "2 4";
chooseMap[3][6] = "7 2";
chooseMap[3][7] = "5 3";

chooseMap[4][0] = "2 4";
chooseMap[4][1] = "1 6";
chooseMap[4][2] = "5 5";
chooseMap[4][3] = "5 6";
chooseMap[4][4] = "2 2";
chooseMap[4][5] = "7 4";
chooseMap[4][6] = "NULL";
chooseMap[4][7] = "6 3";

chooseMap[5][0] = "2 5";
chooseMap[5][1] = "NULL";
chooseMap[5][2] = "0 7";
chooseMap[5][3] = "NULL";
chooseMap[5][4] = "6 5";
chooseMap[5][5] = "7 7";
chooseMap[5][6] = "6 7";
chooseMap[5][7] = "5 3";

chooseMap[6][0] = "3 7";
chooseMap[6][1] = "4 7";
chooseMap[6][2] = "5 7";
chooseMap[6][3] = "2 4";
chooseMap[6][4] = "NULL";
chooseMap[6][5] = "7 6";
chooseMap[6][6] = "NULL";
chooseMap[6][7] = "4 7";

chooseMap[7][0] = "2 7";
chooseMap[7][1] = "NULL";
chooseMap[7][2] = "6 7";
chooseMap[7][3] = "7 7";
```

```
chooseMap[7][4] = "7 6";
chooseMap[7][5] = "7 5";
chooseMap[7][6] = "6 5";
chooseMap[7][7] = "5 5";


// EoCMAP

var pi = 3.1415926535897931;
var d = 8.25; //8.7; //diameter of wheel, cm
var l = 17.5; // base of robot
var degInRad = 180 / pi;

var alpha = 0;

var readGyro = brick.gyroscope().read

function readYaw() {
  return -readGyro()[6];
}
var fullTurn = 4 * 87; // from experimets. Not 360 degrees, but ....

//кнопки
var stopKey = KeysEnum.Esc;
var startKey = KeysEnum.Left;

//диод
var led = brick.led();

//моторы
var mLeft = brick.motor(M3).setPower;
var mRight = brick.motor(M4).setPower;

//сенсоры ИК
irLeft = brick.sensor(A2).read;
irRight = brick.sensor(A1).read;

//датчики света
lightLeft = brick.sensor(A5).read;
lightRight = brick.sensor(A6).read;

//Сенсор US
usForward = brick.sensor(D2).read;
```

```
usBack = brick.sensor(D1).read;

//энкодеры
var eLeft = brick.encoder(E3);
var eRight = brick.encoder(E4);
var cpr = 374; //560; //374; // количество сигналов на оборот

//длина клетки
var cellLength = 37 * cpr / (pi * d);

//переопределение метода нажатия клавиши
if (0) {
  var key = 0;

  function wasPressed(_key) {
    key = 0;
    if (brick.keys().wasPressed(_key)) {
      key = _key;
      return 1;
    } else {
      return 0;
    }
  }
}

var direction = 0; // absolute angle of direction movement
var directionOld = 0;

print("-----------------------------------------");

led.red();

eLeft.reset();
eRight.reset();

var el = eLeft.readRawData();
var er = eRight.readRawData();
mLeft(0);
mRight(0);
print("Start");

//инициализация и калибровка гироскопа
print("gyro initialaize...");
```

```
brick.gyroscope().calibrate(12000);
script.wait(13000);
print("gyro inited");
script.wait(1000);

var v = 20; // velocity

var ex = 0;
var ey = 0;
var encLeftOld = 0;
var encRightOld = 0;
var encLeft = 0;
var encRight = 0;
var n = 0;

// вычисление абсолютного угла относительно начального положения
function angle() {
  var sgn = 0;
  var _direction = readYaw(); // mgrad
  var dtDirection = _direction - directionOld;

  sgn = directionOld == 0 ? 0 : directionOld /
Math.abs(directionOld);
  n += sgn * Math.floor(Math.abs(dtDirection / 320000));
  direction = _direction + n * 360000;
  directionOld = _direction;
}

var t = 0;

print("Run, robot, run!");


// делаем препывание основной программы с частотой 200Гц, чтобы
посчитать абсолютный угол с гироскопа
var mtimer = script.timer(50);
mtimer.timeout.connect(angle);

// вывод в консоль показаний гироскопа
function printGyro() {
  return;
}
```

```
var ptimer = script.timer(300);
ptimer.timeout.connect(printGyro);


//поворот на угол по гироскопу
function turnDirection(_angle, _v) {
  _angle = _angle * 1000; //toMGrad
  var _vel = _v == undefined ? 10 : _v;
  var angleOfRotate = _angle - direction;
  print("ar=" + angleOfRotate);
  var sgn = angleOfRotate == 0 ? 0 : angleOfRotate / Math.abs(
    angleOfRotate);
  mLeft(_vel * sgn);
  mRight(-_vel * sgn);
  while (Math.abs(angleOfRotate = _angle - direction) > 20000) {
    script.wait(50);
  }
  brick.motor(M3).forceBreak(500);
  brick.motor(M4).forceBreak(500);
  script.wait(500);
}

// проезд в перед на количество ячеек _kceel со скоростью _v выравниваясь
по гироскопу на угол _alpha
function forward(_alpha, _v, _kcell) {
  var _vel = _v == undefined ? 10 : _v;
  var u = 0;
  var el = Math.abs(eLeft.readRawData());
  while (Math.abs(eLeft.readRawData()) < el + (_kcell * cellLength))
{
    u = -0.7 * (_alpha - readYaw() / 1000);
    mLeft(_vel - u);
    mRight(_vel + u);
    script.wait(5);
  }
  brick.motor(M3).forceBreak();
  brick.motor(M4).forceBreak();
  script.wait(300);
}

// проезд назад, чтобы упереться в стенку с целью выравнивания
function backward() {
  mLeft(-50);
```

```
  mRight(-50);
  var eLOld = eLeft.readRawData();
  var eROld = eRight.readRawData();
  var el = eLOld;
  var er = eROld;
  do {
    eLOld = el;
    eROld = er;
    script.wait(1000);
    el = eLeft.readRawData();
    er = eRight.readRawData();
  }
  while (Math.abs(eLOld - el) > 200 && Math.abs(eROld - er) > 200)
  brick.motor(M3).forceBreak();
  brick.motor(M4).forceBreak();
  script.wait(300);

}

var main = function() {
  angle = 0;
  sectors = 0;
  isLastForward = true;
  var currentMap = [];
  for (var i = 0; i <= 4; i++) {
    currentMap[i] = [];
  }
  while (sectors <= 4) {
    if (usForward() < 30)
      currentMap[sectors][0] = 1;
    else
      currentMap[sectors][0] = 0;

    if (irRight() < 30)
      currentMap[sectors][1] = 1;
    else
      currentMap[sectors][1] = 0;

    if (usBack() < 30)
      currentMap[sectors][2] = 1;
    else
      currentMap[sectors][2] = 0;
```

```
if (irLeft() < 30)
  currentMap[sectors][3] = 1;
else
  currentMap[sectors][3] = 0;

if (sectors == 4)
  break;

script.wait(1000);
if (currentMap[sectors][1] == 0) {
  print("Rgt");
  turnDirection(-fullTurn / 4, 40);
  script.wait(1000);
  if (currentMap[sectors][3] == 1) {
    backward();
    brick.gyroscope().calibrate(3000);
    script.wait(3000);
    forward(0, 35, 0.2);
  } else {
    brick.gyroscope().calibrate(3000);
    script.wait(3000);
  }
  forward(0, 35, 1);
} else if (currentMap[sectors][0] == 0) {
  print("Fwd");
  forward(0, 35, 1);
} else if (currentMap[sectors][3] == 0) {
  print("Lft");
  turnDirection(fullTurn / 4, 40);
  backward();
  brick.gyroscope().calibrate(3000);
  script.wait(3000);
  forward(0, 35, 0.2);
  forward(0, 35, 1);
} else if (currentMap[sectors][2] == 0) {
  print("Bck");
  turnDirection(fullTurn / 4, 40);
  script.wait(1000);
  backward();
  brick.gyroscope().calibrate(3000);
  script.wait(3000);
  forward(0, 35, 0.2);
  script.wait(1000);
```

```
    turnDirection(fullTurn / 4, 40);
    script.wait(1000);
    backward();
    brick.gyroscope().calibrate(3000);
    script.wait(3000);
    forward(0, 35, 0.2);
    forward(0, 35, 1);
  }
  sectors++;
}
var isCurrent = true;
var cur = 0;

for (var i = 0; i < 8; i++) {
  for (var j = 0; j < 8; j++) {
    isCurrent = true;
    if (map[i][j] == "NONE" || map[i][j] == "NULL") {
      continue;
      isCurrent = false;
    }
    for (var s = 0; s <= 4; s++) {
      for (var d = 0; d <= 3; d++) {
        if (map[i][j][s][d] != currentMap[s][d]) {
          isCurrent = false;
        }
      }
    }

    if (isCurrent) {
      cur++;
    }

  }
}

if (cur != 1) {
  main();
  return;
}

for (var i = 0; i < 8; i++) {
  for (var j = 0; j < 8; j++) {
    isCurrent = true;
```

```
      if (map[i][j] == "NONE" || map[i][j] == "NULL") {
        continue;
        isCurrent = false;
      }
      for (var s = 0; s <= 4; s++) {
        for (var d = 0; d <= 3; d++) {
          if (map[i][j][s][d] != currentMap[s][d]) {
            isCurrent = false;
          }
        }
      }

      if (isCurrent) {
        brick.display().addLabel(chooseMap[i][j], 10, 10);
        brick.display().redraw();
        script.wait(10000);
        return;
      }

    }
  }
}

main();
```

Программа команды для решения задачи третьего этапа:

```
var pi = 3.1415926535897931;
var d = 8.25; //8.7; //diameter of wheel, cm
var l = 17.5; // base of robot
var degInRad = 180 / pi;

var alpha = 0;

var readGyro = brick.gyroscope().read

function readYaw() {
  return -readGyro()[6];
}
var fullTurn = 4 * 87; // from experimets. Not 360 degrees, but ....

//кнопки
```

```
var stopKey = KeysEnum.Esc;
var startKey = KeysEnum.Left;

//диод
var led = brick.led();

//моторы
var mLeft = brick.motor(M3).setPower;
var mRight = brick.motor(M4).setPower;

//сенсоры ИК
irLeft = brick.sensor(A2).read;
irRight = brick.sensor(A1).read;

//датчики света
lightLeft = brick.sensor(A5).read;
lightRight = brick.sensor(A6).read;

//Сенсор US
usForward = brick.sensor(D2).read;
usBack = brick.sensor(D1).read;

//энкодеры
var eLeft = brick.encoder(E3);
var eRight = brick.encoder(E4);
var cpr = 374; //560; //374; // количество сигналов на оборот

//длина клетки
var cellLength = 37 * cpr / (pi * d);


brick.gyroscope().calibrate(10000);
script.wait(10500);
// To code
var k = 0.5;
var speed = 0;
var grey = 12;
while (lightLeft() < grey && lightRight() < grey) {
  err = brick.gyroscope().read()[6] / 1000 * k;
  mLeft(Math.min(speed + 1, 30) + err);
  mRight(Math.min(speed + 1, 30) - err);
  script.wait(5);
  speed += 0.5;
```

```
}
print("STOP: " + lightLeft() + ' ' + lightRight());
brick.motor(M3).forceBreak();
brick.motor(M4).forceBreak();
script.wait(1000);

var time = 230;
var grey = 45;

print(lightLeft() + ' ' + lightRight());
var first = (lightLeft() + lightRight()) / 2;
if (first > grey) {
  first = 1;
} else {
  first = 0;
}


mLeft(25);
mRight(25);
script.wait(time);
brick.motor(M3).forceBreak();
brick.motor(M4).forceBreak();
script.wait(1000);
print(lightLeft() + ' ' + lightRight());
var second = (lightLeft() + lightRight()) / 2;
if (second > grey) {
  second = 1;
} else {
  second = 0;
}


mLeft(25);
mRight(25);
script.wait(time);
brick.motor(M3).forceBreak();
brick.motor(M4).forceBreak();
script.wait(1000);
print(lightLeft() + ' ' + lightRight());
var third = (lightLeft() + lightRight()) / 2;
if (third > grey) {
  third = 1;
```

```
} else {
  third = 0;
}


mLeft(25);
mRight(25);
script.wait(time);
brick.motor(M3).forceBreak();
brick.motor(M4).forceBreak();
script.wait(1000);
print(lightLeft() + ' ' + lightRight());
var fourth = (lightLeft() + lightRight()) / 2;
if (fourth > grey) {
  fourth = 1;
} else {
  fourth = 0;
}

mLeft(25);
mRight(25);
script.wait(1150);
brick.motor(M3).forceBreak();
brick.motor(M4).forceBreak();
script.wait(1000);

var number = first * 8 + second * 4 + third * 2 + fourth;

print("!!! " + number + " !!!");
brick.display().addLabel(number, 10, 10);
brick.display().redraw();
script.wait(10000);
```

Программа команды для решения задачи “построение маршрута” четвертого этапа:

```
/*
 * Very big function that find optimal way
 * from one cell to other using bfs and non-
 * optimal dfs.
 * Function findRoad(from, to, direction)
 * return string that has format "FFRFLF"
```

```
 * with movements from first cell to second.
 * Can works very long time with big ranges.
 *
 * NOTE: there using div(n, m) function for
 * division n on m.
 */
var yStart = 3;
var xStart = 0;
var direct = 1;
var yFinish = 4;
var xFinish = 4;


function div(val, by) {
  return (val - val % by) / by;
}

var mapGraph = [];

for (var i = 0; i < 8 * 8; i++) {
  mapGraph = [];
}

mapGraph[0 + 0 * 8] = [2, 1, 8];
mapGraph[1 + 0 * 8] = [2, 0, 2];
mapGraph[2 + 0 * 8] = [3, 1, 3, 10];
mapGraph[3 + 0 * 8] = [2, 2, 4];
mapGraph[4 + 0 * 8] = [2, 3, 5];
mapGraph[5 + 0 * 8] = [2, 4, 13];
mapGraph[6 + 0 * 8] = [0];
mapGraph[7 + 0 * 8] = [1, 15];

mapGraph[0 + 1 * 8] = [2, 0, 16];
mapGraph[1 + 1 * 8] = [0];
mapGraph[2 + 1 * 8] = [2, 2, 18];
mapGraph[3 + 1 * 8] = [1, 19];
mapGraph[4 + 1 * 8] = [0];
mapGraph[5 + 1 * 8] = [3, 5, 14, 21];
mapGraph[6 + 1 * 8] = [2, 13, 15];
mapGraph[7 + 1 * 8] = [2, 7, 14];

mapGraph[0 + 2 * 8] = [2, 8, 17];
mapGraph[1 + 2 * 8] = [2, 16, 18];
```

```
mapGraph[2 + 2 * 8] = [3, 10, 17, 26];
mapGraph[3 + 2 * 8] = [3, 11, 20, 27];
mapGraph[4 + 2 * 8] = [2, 19, 21];
mapGraph[5 + 2 * 8] = [3, 13, 20, 29];
mapGraph[6 + 2 * 8] = [0];
mapGraph[7 + 2 * 8] = [1, 31];

mapGraph[0 + 3 * 8] = [1, 32];
mapGraph[1 + 3 * 8] = [0];
mapGraph[2 + 3 * 8] = [2, 18, 34];
mapGraph[3 + 3 * 8] = [1, 19];
mapGraph[4 + 3 * 8] = [0];
mapGraph[5 + 3 * 8] = [3, 21, 30, 37];
mapGraph[6 + 3 * 8] = [2, 29, 31];
mapGraph[7 + 3 * 8] = [3, 23, 30, 39];

mapGraph[0 + 4 * 8] = [3, 24, 33, 40];
mapGraph[1 + 4 * 8] = [2, 32, 34];
mapGraph[2 + 4 * 8] = [4, 26, 33, 35, 42];
mapGraph[3 + 4 * 8] = [2, 34, 36];
mapGraph[4 + 4 * 8] = [3, 35, 37, 44];
mapGraph[5 + 4 * 8] = [3, 29, 36, 45];
mapGraph[6 + 4 * 8] = [0];
mapGraph[7 + 4 * 8] = [1, 31];

mapGraph[0 + 5 * 8] = [1, 32];
mapGraph[1 + 5 * 8] = [0];
mapGraph[2 + 5 * 8] = [2, 34, 50];
mapGraph[3 + 5 * 8] = [0];
mapGraph[4 + 5 * 8] = [2, 36, 45];
mapGraph[5 + 5 * 8] = [4, 37, 44, 46, 53];
mapGraph[6 + 5 * 8] = [2, 45, 47];
mapGraph[7 + 5 * 8] = [2, 46, 55];

mapGraph[0 + 6 * 8] = [2, 49, 56];
mapGraph[1 + 6 * 8] = [2, 48, 50];
mapGraph[2 + 6 * 8] = [4, 42, 49, 51, 58];
mapGraph[3 + 6 * 8] = [2, 50, 59];
mapGraph[4 + 6 * 8] = [0];
mapGraph[5 + 6 * 8] = [1, 45];
mapGraph[6 + 6 * 8] = [0];
mapGraph[7 + 6 * 8] = [2, 47, 63];
```

```
mapGraph[0 + 7 * 8] = [1, 48];
mapGraph[1 + 7 * 8] = [0];
mapGraph[2 + 7 * 8] = [2, 50, 59];
mapGraph[3 + 7 * 8] = [3, 51, 58, 60];
mapGraph[4 + 7 * 8] = [2, 59, 61];
mapGraph[5 + 7 * 8] = [2, 60, 62];
mapGraph[6 + 7 * 8] = [2, 61, 63];
mapGraph[7 + 7 * 8] = [2, 55, 62];

var mapBfs = [];
for (var i = 0; i <= 63; i++) {
  mapBfs[i] = -1;
}
var queue = [];
var queueLast = 1;

function search(cell, width) {
  for (var i = 1; i <= mapGraph[cell][0]; i++) {
    if (mapBfs[mapGraph[cell][i]] == -1) {
      queue[queueLast] = mapGraph[cell][i];
      mapBfs[mapGraph[cell][i]] = width + 1;
      queueLast++;
    }
  }
}

function bfs(cell) {
  var i = 0;
  queue[0] = cell;
  mapBfs[cell] = 0;
  while (i != queueLast) {
    search(queue[i], mapBfs[queue[i]]);
    i++;
  }
}

function bfsClear() {
  for (var i = 0; i < 8 * 8; i++) {
    mapBfs[i] = -1;
    queue = [];
    queueLast = 1;
  }
}
```

```
function dfs(cell, to, range) {
  if (range != 0) {
    for (var i = 1; i <= mapGraph[cell][0]; i++) {
      road = dfs(mapGraph[cell][i], to, range - 1);
      if (road != false) {
        if (cell < 10)
          return '0' + cell + ' ' + road;
        return cell + ' ' + road;
      }
    }
  } else {
    if (cell == to) {
      if (cell < 10)
        return '0' + cell;
      return cell;
    } else {
      return false;
    }
  }
  return 0;
}

function findRoad(from, to, direction) {
  bfs(from);
  var range = mapBfs[to];
  bfsClear();
  var badRoad = dfs(from, to, range);
  var goodRoad = [];
  for (var i = 0; i < (badRoad.length + 1) / 3; i++) {
    goodRoad[i] = [];
    goodRoad[i][0] = div(+(badRoad[i * 3] + badRoad[i * 3 + 1]), 8);
    goodRoad[i][1] = (+(badRoad[i * 3] + badRoad[i * 3 + 1]) % 8);
  }
  var result = "";
  for (var i = 1; i < goodRoad.length; i = i + 1 - 1) {
    if (goodRoad[i][1] == goodRoad[i - 1][1] - 1) {
      if (direction == 2) {
        result += "F";
        i++;
        continue;
      }
      if (direction == 1) {
```

```
      result += "R";
      direction = (direction + 1) % 4;
      continue;
    }
    if (direction == 3) {
      result += "L";
      direction = (direction + 3) % 4;
      continue;
    }
    if (direction == 0) {
      result += "RR";
      direction = (direction + 2) % 4;
      continue;
    }
  }
  if (goodRoad[i][1] == goodRoad[i - 1][1] + 1) {
    if (direction == 0) {
      result += "F";
      i++;
      continue;
    }
    if (direction == 3) {
      result += "R";
      direction = (direction + 1) % 4;
      continue;
    }
    if (direction == 1) {
      result += "L";
      direction = (direction + 3) % 4;
      continue;
    }
    if (direction == 2) {
      result += "RR";
      direction = (direction + 2) % 4;
      continue;
    }
  }
  if (goodRoad[i][0] == goodRoad[i - 1][0] - 1) {
    if (direction == 3) {
      result += "F";
      i++;
      continue;
    }
```

```
      if (direction == 2) {
        result += "R";
        direction = (direction + 1) % 4;
        continue;
      }
      if (direction == 0) {
        result += "L";
        direction = (direction + 3) % 4;
        continue;
      }
      if (direction == 1) {
        result += "RR";
        direction = (direction + 2) % 4;
        continue;
      }
    }
    if (goodRoad[i][0] == goodRoad[i - 1][0] + 1) {
      if (direction == 1) {
        result += "F";
        i++;
        continue;
      }
      if (direction == 0) {
        result += "R";
        direction = (direction + 1) % 4;
        continue;
      }
      if (direction == 2) {
        result += "L";
        direction = (direction + 3) % 4;
        continue;
      }
      if (direction == 3) {
        result += "RR";
        direction = (direction + 2) % 4;
        continue;
      }
    }
  }
  return result;
}
```

```
way = findRoad(yStart * 8 + xStart, yFinish * 8 + xFinish, direct);
print(way);

var pi = 3.1415926535897931;
var d = 8.25; //8.7; //diameter of wheel, cm
var l = 17.5; // base of robot
var degInRad = 180 / pi;

var alpha = 0;

var readGyro = brick.gyroscope().read

function readYaw() {
  return -readGyro()[6];
}
var fullTurn = 4 * 87; // from experimets. Not 360 degrees, but ....

//кнопки
var stopKey = KeysEnum.Esc;
var startKey = KeysEnum.Left;

//диод
var led = brick.led();

//моторы
var mLeft = brick.motor(M3).setPower;
var mRight = brick.motor(M4).setPower;

//сенсоры ИК
irLeft = brick.sensor(A2).read;
irRight = brick.sensor(A1).read;

//датчики света
lightLeft = brick.sensor(A5).read;
lightRight = brick.sensor(A6).read;

//Сенсор US
usForward = brick.sensor(D2).read;
usBack = brick.sensor(D1).read;
//энкодеры
var eLeft = brick.encoder(E3);
var eRight = brick.encoder(E4);
var cpr = 374; //560; //374; // количество сигналов на оборот
```

```
//длина клетки
var cellLength = 35 * cpr / (pi * d);

//переопределение метода нажатия клавиши
if (0) {
  var key = 0;

  function wasPressed(_key) {
    key = 0;
    if (brick.keys().wasPressed(_key)) {
      key = _key;
      return 1;
    } else {
      return 0;
    }
  }
}

var direction_angle = 0; // absolute angle of direction movement
var directionOld = 0;

print("-----------------------------------------");

led.red();

eLeft.reset();
eRight.reset();

var el = eLeft.readRawData();
var er = eRight.readRawData();
mLeft(0);
mRight(0);
print("Start");

//инициализация и калибровка гироскопа
print("gyro initialaize...");
brick.gyroscope().calibrate(4000);
script.wait(4000);
script.wait(1000);

var v = 20; // velocity
```

```
var ex = 0;
var ey = 0;
var encLeftOld = 0;
var encRightOld = 0;
var encLeft = 0;
var encRight = 0;
var n = 0;
var ldata = [];
var edata = [];

// вычисление абсолютного угла относительно начального положения
function angle() {
  var sgn = 0;
  var _direction = readYaw(); // mgrad
  var dtDirection = _direction - directionOld;

  sgn = directionOld == 0 ? 0 : directionOld /
Math.abs(directionOld);
  n += sgn * Math.floor(Math.abs(dtDirection / 320000));
  direction_angle = _direction + n * 360000;
  directionOld = _direction;
}

var t = 0;

print("Run, robot, run!");


// делаем препывание основной программы с частотой 200Гц, чтобы
посчитать абсолютный угол с гироскопа
var mtimer = script.timer(50);
mtimer.timeout.connect(angle);

//поворот на угол по гироскопу
function turnDirection(_angle, _v) {
  _angle = _angle * 1000; //toMGrad
  var _vel = _v == undefined ? 10 : _v;
  var angleOfRotate = _angle - direction_angle;
  print("ar=" + angleOfRotate);
  var sgn = angleOfRotate == 0 ? 0 : angleOfRotate / Math.abs(
    angleOfRotate);
  mLeft(_vel * sgn);
  mRight(-_vel * sgn);
```

```
  while (Math.abs(angleOfRotate = _angle - direction_angle) > 20000)
{
    script.wait(50);
  }
  brick.motor(M3).forceBreak(500);
  brick.motor(M4).forceBreak(500);
  script.wait(500);
}
black = false;
z = 0;
// проезд в перед на количество ячеек _kceel со скоростью _v выравниваясь
по гироскопу на угол _alpha

function forward(_alpha, _v, _kcell) {
  eLeft.reset();
  eRight.reset();
  var _vel = _v == undefined ? 10 : _v;
  var u = 0;
  var el = Math.abs(eLeft.readRawData());
  var irLeftNorm = 0;
  var irRightNorm = 0;
  while (Math.abs(eLeft.readRawData()) < el + (_kcell * cellLength))
{
    err_irLeft = irLeftNorm - irLeft();
    u = -0.7 * (_alpha - readYaw() / 1000);
    z++;
    edata[z] = eLeft.readRawData();
    ldata[z] = (lightLeft() + lightRight()) / 2;
    script.wait(5);
    mLeft(_vel - u);
    mRight(_vel + u);
    script.wait(5);
    print(edata);
    if (usForward() < 7) {
      break;
    }
  }
  brick.motor(M3).forceBreak();
  brick.motor(M4).forceBreak();
  script.wait(300);
}
// проезд назад, чтобы упереться в стенку с целью выравнивания
function backward() {
```

```
  mLeft(-35);
  mRight(-35);
  var eLOld = eLeft.readRawData();
  var eROld = eRight.readRawData();
  var el = eLOld;
  var er = eROld;
  do {
    eLOld = el;
    eROld = er;
    script.wait(1000);
    el = eLeft.readRawData();
    er = eRight.readRawData();
  }
  while (Math.abs(eLOld - el) > 200 && Math.abs(eROld - er) > 200)
  brick.motor(M3).forceBreak();
  brick.motor(M4).forceBreak();
  script.wait(300);
  eLeft.reset();
  eRight.reset();

}

function FORWARD() {
  forward(0, 35, 1);
  script.wait(500);
}

function RIGHT() {
  turnDirection(-fullTurn / 4, 37);
  script.wait(500);
  if (usBack() < 30) {
    backward();
    brick.gyroscope().calibrate(3000);
    script.wait(3100);
    forward(0, 35, 0.2);
    script.wait(100);
  } else {
    brick.gyroscope().calibrate(3000);
    script.wait(3000);
  }
}

function LEFT() {
```

```
  turnDirection(fullTurn / 4, 37);
  script.wait(500);
  if (usBack() < 30) {
    backward();
    brick.gyroscope().calibrate(3000);
    script.wait(3100);
    forward(0, 35, 0.2);
    script.wait(100);
  } else {
    brick.gyroscope().calibrate(3000);
    script.wait(3000);
  }
}

function move_to_finish(way) {
  print("YOO");
  for (var i = 0; i < way.length; i++) {
    if (way[i] == 'F') {
      print("F");
      FORWARD();
      continue;
    }
    if (way[i] == 'R') {
      print("R");
      RIGHT();
      continue;
    }
    if (way[i] == 'L') {
      print("L");
      LEFT();
      continue;
    }
  }
}
brick.display().addLabel(way, 5, 5);
brick.display().redraw();
script.wait(10000);
move_to_finish(way);
```

Программа команды для решения полной финальной задачи:

```
var Map = [];
```

```
var chooseMap = [];
for (var d = 0; d < 4; d++) {
  Map[d] = [];
  chooseMap[d] = [];
  for (var i = 0; i < 8; i++) {
    Map[d][i] = [];
    chooseMap[d][i] = [];
  }
}

//Map 1 элемент-направление, 2 элемент-y, 3 элемент x;
//chooseMap 1 элемент направление, 2 элемент y, 3 элемент x;

Map[0][0][0] = [
  [0, 0, 1, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1]
];
Map[0][0][1] = [
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[0][0][2] = [
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1]
];
Map[0][0][3] = [
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 0],
  [0, 0, 0, 1]
];
Map[0][0][4] = [
  [0, 1, 0, 1],
```

```
  [1, 0, 0, 1],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[0][0][5] = [
  [1, 0, 0, 1],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];
Map[0][0][6] = "NONE";
Map[0][0][7] = [
  [1, 0, 1, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 0]
];

Map[0][1][0] = [
  [1, 0, 1, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[0][1][1] = "NONE";
Map[0][1][2] = [
  [1, 0, 1, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[0][1][3] = [
  [1, 0, 1, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
```

```
Map[0][1][4] = "NONE";
Map[0][1][5] = [
  [0, 0, 1, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1]
];
Map[0][1][6] = [
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [1, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[0][1][7] = [
  [1, 1, 0, 0],
  [1, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];

Map[0][2][0] = [
  [0, 1, 1, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[0][2][1] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[0][2][2] = [
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
```

```
];
Map[0][2][3] = [
  [0, 0, 1, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];
Map[0][2][4] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 0]
];
Map[0][2][5] = [
  [1, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[0][2][6] = "NONE";
Map[0][2][7] = [
  [1, 0, 1, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0]
];

Map[0][3][0] = [
  [1, 0, 1, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[0][3][1] = "NONE";
Map[0][3][2] = [
  [1, 0, 1, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
```

```
  [1, 0, 0, 0],
  [1, 1, 0, 1]
];
Map[0][3][3] = [
  [1, 1, 1, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0]
];
Map[0][3][4] = "NONE";
Map[0][3][5] = [
  [0, 0, 1, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[0][3][6] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1]
];
Map[0][3][7] = [
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1]
];

Map[0][4][0] = [
  [0, 0, 1, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[0][4][1] = [
  [0, 1, 0, 1],
  [0, 0, 0, 0],
```

```
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[0][4][2] = [
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
];
Map[0][4][3] = [
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [1, 1, 0, 0],
  [0, 0, 0, 0],
  [1, 1, 0, 1]
];
Map[0][4][4] = [
  [0, 0, 0, 1],
  [1, 1, 0, 0],
  [0, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[0][4][5] = [
  [1, 0, 0, 0],
  [0, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[0][4][6] = "NONE";
Map[0][4][7] = [
  [1, 1, 1, 0],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];

Map[0][5][0] = [
  [1, 1, 1, 0],
```

```
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[0][5][1] = "NONE";
Map[0][5][2] = [
  [1, 0, 1, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [1, 1, 0, 1]
];
Map[0][5][3] = "NONE";
Map[0][5][4] = [
  [0, 1, 1, 0],
  [0, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[0][5][5] = [
  [0, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 1]
];
Map[0][5][6] = [
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[0][5][7] = [
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
```

```
Map[0][6][0] = [
  [0, 0, 1, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 0]
];
Map[0][6][1] = [
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[0][6][2] = [
  [0, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
Map[0][6][3] = [
  [1, 0, 0, 1],
  [1, 0, 0, 0],
  [1, 0, 0, 1],
  [0, 0, 0, 0],
  [1, 0, 0, 1]
];
Map[0][6][4] = "NONE";
Map[0][6][5] = [
  [1, 1, 1, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[0][6][6] = "NONE";
Map[0][6][7] = [
  [1, 0, 1, 0],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
```

```
];

Map[0][7][0] = [
  [1, 1, 1, 0],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [1, 1, 0, 0]
];
Map[0][7][1] = "NONE";
Map[0][7][2] = [
  [0, 1, 1, 0],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
Map[0][7][3] = [
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
];
Map[0][7][4] = [
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[0][7][5] = [
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
];
Map[0][7][6] = [
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
```

```
  [0, 1, 0, 1]
];
Map[0][7][7] = [
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];


Map[1][0][0] = [
  [0, 1, 1, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];
Map[1][0][1] = [
  [1, 0, 1, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[1][0][2] = [
  [0, 0, 1, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
];
Map[1][0][3] = [
  [1, 0, 1, 0],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[1][0][4] = [
  [1, 0, 1, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 0],
```

```
  [0, 1, 0, 1],
  [1, 1, 0, 0]
];
Map[1][0][5] = [
  [0, 0, 1, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[1][0][6] = "NONE";
Map[1][0][7] = [
  [0, 1, 1, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 0]
];

Map[1][1][0] = [
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[1][1][1] = "NONE";
Map[1][1][2] = [
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[1][1][3] = [
  [0, 1, 1, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[1][1][4] = "NONE";
Map[1][1][5] = [
```

```
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1]
];
Map[1][1][6] = [
  [1, 0, 1, 0],
  [1, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
Map[1][1][7] = [
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 1]
];

Map[1][2][0] = [
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[1][2][1] = [
  [1, 0, 1, 0],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[1][2][2] = [
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1]
];
Map[1][2][3] = [
```

```
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];
Map[1][2][4] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 0]
];
Map[1][2][5] = [
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0]
];
Map[1][2][6] = "NONE";
Map[1][2][7] = [
  [0, 1, 1, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0]
];

Map[1][3][0] = [
  [0, 1, 1, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[1][3][1] = "NONE";
Map[1][3][2] = [
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1]
```

```
];
Map[1][3][3] = [
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0]
];
Map[1][3][4] = "NONE";
Map[1][3][5] = [
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[1][3][6] = [
  [1, 0, 1, 0],
  [1, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[1][3][7] = [
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1]
];

Map[1][4][0] = [
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[1][4][1] = [
  [1, 0, 1, 0],
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0],
```

```
  [1, 1, 0, 1]
];
Map[1][4][2] = [
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0]
];
Map[1][4][3] = [
  [1, 0, 1, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[1][4][4] = [
  [0, 0, 1, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 0]
];
Map[1][4][5] = [
  [0, 0, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[1][4][6] = "NONE";
Map[1][4][7] = [
  [1, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];

Map[1][5][0] = [
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
```

```
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[1][5][1] = "NONE";
Map[1][5][2] = [
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [1, 1, 0, 1]
];
Map[1][5][3] = "NONE";
Map[1][5][4] = [
  [1, 1, 0, 0],
  [0, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[1][5][5] = [
  [0, 0, 0, 0],
  [1, 0, 0, 1],
  [1, 0, 0, 0],
  [1, 0, 0, 0],
  [0, 0, 0, 0]
];
Map[1][5][6] = [
  [1, 0, 1, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[1][5][7] = [
  [0, 0, 1, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1]
];

Map[1][6][0] = [
  [0, 1, 1, 0],
```

```
  [1, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[1][6][1] = [
  [1, 0, 1, 0],
  [1, 1, 0, 1],
  [1, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[1][6][2] = [
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 1],
  [1, 1, 0, 1],
  [1, 0, 0, 1]
];
Map[1][6][3] = [
  [0, 0, 1, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[1][6][4] = "NONE";
Map[1][6][5] = [
  [1, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[1][6][6] = "NONE";
Map[1][6][7] = [
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
```

```
Map[1][7][0] = [
  [1, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [1, 1, 0, 0]
];
Map[1][7][1] = "NONE";
Map[1][7][2] = [
  [1, 1, 0, 0],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
Map[1][7][3] = [
  [1, 0, 0, 0],
  [1, 0, 0, 1],
  [0, 0, 0, 0],
  [1, 0, 0, 1],
  [1, 0, 0, 0]
];
Map[1][7][4] = [
  [1, 0, 1, 0],
  [0, 0, 0, 1],
  [1, 1, 0, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[1][7][5] = [
  [1, 0, 1, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [1, 1, 0, 0],
  [0, 0, 0, 0]
];
Map[1][7][6] = [
  [1, 0, 1, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [1, 1, 0, 0]
];
```

```
Map[1][7][7] = [
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1]
];


Map[2][0][0] = [
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];
Map[2][0][1] = [
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[2][0][2] = [
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
];
Map[2][0][3] = [
  [0, 1, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[2][0][4] = [
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
```

```
];
Map[2][0][5] = [
  [0, 1, 1, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[2][0][6] = "NONE";
Map[2][0][7] = [
  [1, 1, 1, 0],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 0]
];

Map[2][1][0] = [
  [1, 0, 1, 0],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[2][1][1] = "NONE";
Map[2][1][2] = [
  [1, 0, 1, 0],
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1]
];
Map[2][1][3] = [
  [1, 1, 1, 0],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[2][1][4] = "NONE";
Map[2][1][5] = [
  [1, 0, 0, 0],
  [1, 1, 0, 0],
```

```
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 0]
];
Map[2][1][6] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
Map[2][1][7] = [
  [0, 0, 1, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 1]
];

Map[2][2][0] = [
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1]
];
Map[2][2][1] = [
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[2][2][2] = [
  [0, 0, 1, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1]
];
Map[2][2][3] = [
  [1, 0, 0, 0],
  [1, 1, 0, 1],
```

```
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1]
];
Map[2][2][4] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1]
];
Map[2][2][5] = [
  [0, 0, 1, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0]
];
Map[2][2][6] = "NONE";
Map[2][2][7] = [
  [1, 1, 1, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0]
];

Map[2][3][0] = [
  [1, 1, 1, 0],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[2][3][1] = "NONE";
Map[2][3][2] = [
  [1, 0, 1, 0],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[2][3][3] = [
```

```
  [1, 0, 1, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0]
];
Map[2][3][4] = "NONE";
Map[2][3][5] = [
  [1, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
];
Map[2][3][6] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[2][3][7] = [
  [0, 0, 1, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];

Map[2][4][0] = [
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1]
];
Map[2][4][1] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1]
];
```

```
Map[2][4][2] = [
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];
Map[2][4][3] = [
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[2][4][4] = [
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 0]
];
Map[2][4][5] = [
  [0, 0, 1, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1]
];
Map[2][4][6] = "NONE";
Map[2][4][7] = [
  [1, 0, 1, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];

Map[2][5][0] = [
  [1, 0, 1, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
```

```
];
Map[2][5][1] = "NONE";
Map[2][5][2] = [
  [1, 0, 1, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [1, 1, 0, 0]
];
Map[2][5][3] = "NONE";
Map[2][5][4] = [
  [1, 0, 0, 1],
  [1, 0, 0, 0],
  [1, 0, 0, 0],
  [0, 0, 0, 0],
  [1, 0, 0, 1]
];
Map[2][5][5] = [
  [0, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];
Map[2][5][6] = [
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[2][5][7] = [
  [0, 1, 1, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1]
];

Map[2][6][0] = [
  [1, 1, 0, 0],
  [1, 1, 0, 1],
  [1, 0, 0, 1],
```

```
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[2][6][1] = [
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [1, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[2][6][2] = [
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 1]
];
Map[2][6][3] = [
  [0, 1, 1, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[2][6][4] = "NONE";
Map[2][6][5] = [
  [1, 0, 1, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[2][6][6] = "NONE";
Map[2][6][7] = [
  [1, 0, 1, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 0]
];

Map[2][7][0] = [
  [1, 0, 1, 1],
```

```
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [1, 1, 0, 0]
];
Map[2][7][1] = "NONE";
Map[2][7][2] = [
  [1, 0, 0, 1],
  [0, 0, 0, 0],
  [1, 0, 0, 1],
  [1, 0, 0, 0],
  [1, 0, 0, 1]
];
Map[2][7][3] = [
  [0, 0, 0, 1],
  [1, 1, 0, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[2][7][4] = [
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [1, 1, 0, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[2][7][5] = [
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [1, 1, 0, 0],
  [0, 0, 0, 0]
];
Map[2][7][6] = [
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [1, 1, 0, 0]
];
Map[2][7][7] = [
  [0, 0, 1, 1],
```

```
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];


Map[3][0][0] = [
  [1, 0, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[3][0][1] = [
  [1, 0, 1, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[3][0][2] = [
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 0]
];
Map[3][0][3] = [
  [1, 0, 1, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 0],
  [0, 0, 0, 1]
];
Map[3][0][4] = [
  [1, 0, 1, 0],
  [1, 0, 0, 1],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[3][0][5] = [
```

```
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[3][0][6] = "NONE";
Map[3][0][7] = [
  [1, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 0]
];

Map[3][1][0] = [
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[3][1][1] = "NONE";
Map[3][1][2] = [
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1]
];
Map[3][1][3] = [
  [1, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[3][1][4] = "NONE";
Map[3][1][5] = [
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [1, 1, 0, 1],
```

```
  [1, 0, 0, 1]
];
Map[3][1][6] = [
  [1, 0, 1, 0],
  [1, 1, 0, 0],
  [1, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[3][1][7] = [
  [0, 1, 1, 0],
  [1, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];

Map[3][2][0] = [
  [0, 0, 1, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[3][2][1] = [
  [1, 0, 1, 0],
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[3][2][2] = [
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
Map[3][2][3] = [
  [0, 0, 0, 1],
  [1, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
```

```
  [0, 0, 0, 1]
];
Map[3][2][4] = [
  [1, 0, 1, 0],
  [1, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 0]
];
Map[3][2][5] = [
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1]
];
Map[3][2][6] = "NONE";
Map[3][2][7] = [
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0]
];

Map[3][3][0] = [
  [1, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[3][3][1] = "NONE";
Map[3][3][2] = [
  [0, 1, 0, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[3][3][3] = [
  [0, 1, 1, 1],
  [0, 0, 0, 1],
```

```
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [0, 1, 0, 0]
];
Map[3][3][4] = "NONE";
Map[3][3][5] = [
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0]
];
Map[3][3][6] = [
  [1, 0, 1, 0],
  [1, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1]
];
Map[3][3][7] = [
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0]
];

Map[3][4][0] = [
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];
Map[3][4][1] = [
  [1, 0, 1, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[3][4][2] = [
  [0, 0, 0, 0],
```

```
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [1, 1, 0, 0],
  [0, 0, 0, 0]
];
Map[3][4][3] = [
  [1, 0, 1, 0],
  [0, 0, 0, 1],
  [1, 1, 0, 0],
  [0, 0, 0, 0],
  [1, 1, 0, 1]
];
Map[3][4][4] = [
  [1, 0, 0, 0],
  [1, 0, 0, 0],
  [0, 0, 0, 0],
  [1, 0, 0, 1],
  [1, 0, 0, 0]
];
Map[3][4][5] = [
  [0, 1, 0, 0],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 0],
  [1, 1, 0, 1]
];
Map[3][4][6] = "NONE";
Map[3][4][7] = [
  [0, 1, 1, 1],
  [0, 1, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1]
];

Map[3][5][0] = [
  [0, 1, 1, 1],
  [0, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[3][5][1] = "NONE";
```

```
Map[3][5][2] = [
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 1],
  [1, 1, 0, 0]
];
Map[3][5][3] = "NONE";
Map[3][5][4] = [
  [0, 0, 1, 1],
  [0, 0, 0, 0],
  [1, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[3][5][5] = [
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1]
];
Map[3][5][6] = [
  [1, 0, 1, 0],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[3][5][7] = [
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 0, 0, 1]
];

Map[3][6][0] = [
  [1, 0, 0, 1],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 0]
```

```
];
Map[3][6][1] = [
  [1, 0, 1, 0],
  [0, 0, 0, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[3][6][2] = [
  [0, 0, 0, 0],
  [1, 0, 0, 1],
  [1, 0, 0, 0],
  [1, 0, 0, 1],
  [0, 0, 0, 0]
];
Map[3][6][3] = [
  [1, 1, 0, 0],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1]
];
Map[3][6][4] = "NONE";
Map[3][6][5] = [
  [0, 1, 1, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1]
];
Map[3][6][6] = "NONE";
Map[3][6][7] = [
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 0, 0, 0],
  [0, 1, 0, 0]
];

Map[3][7][0] = [
  [0, 1, 1, 1],
  [1, 0, 0, 1],
  [0, 1, 0, 1],
```

```
  [0, 0, 0, 0],
  [1, 1, 0, 0]
];
Map[3][7][1] = "NONE";
Map[3][7][2] = [
  [0, 0, 1, 1],
  [0, 1, 0, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1]
];
Map[3][7][3] = [
  [0, 0, 1, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [1, 1, 0, 1]
];
Map[3][7][4] = [
  [1, 0, 1, 0],
  [0, 1, 0, 1],
  [0, 1, 0, 1],
  [0, 1, 1, 0],
  [0, 1, 0, 1]
];
Map[3][7][5] = [
  [1, 0, 1, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
];
Map[3][7][6] = [
  [1, 0, 1, 0],
  [1, 1, 0, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
  [0, 1, 0, 1]
];
Map[3][7][7] = [
  [0, 1, 1, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0],
```

```
  [0, 1, 0, 1],
  [0, 0, 0, 0]
];



chooseMap[0][0][0] = [0, 2, 2];
chooseMap[0][0][1] = [2, 2, 1];
chooseMap[0][0][2] = [2, 2, 0];
chooseMap[0][0][3] = [1, 2, 5];
chooseMap[0][0][4] = [2, 2, 4];
chooseMap[0][0][5] = [2, 2, 3];
chooseMap[0][0][6] = "NONE";
chooseMap[0][0][7] = [3, 0, 5];

chooseMap[0][1][0] = [2, 2, 1];
chooseMap[0][1][1] = "NONE";
chooseMap[0][1][2] = [3, 1, 0];
chooseMap[0][1][3] = [0, 2, 4];
chooseMap[0][1][4] = "NONE";
chooseMap[0][1][5] = [3, 1, 3];
chooseMap[0][1][6] = [2, 1, 6];
chooseMap[0][1][7] = [2, 1, 5];

chooseMap[0][2][0] = [1, 4, 2];
chooseMap[0][2][1] = [2, 4, 1];
chooseMap[0][2][2] = [2, 4, 0];
chooseMap[0][2][3] = [0, 2, 5];
chooseMap[0][2][4] = [2, 4, 4];
chooseMap[0][2][5] = [2, 4, 3];
chooseMap[0][2][6] = "NONE";
chooseMap[0][2][7] = [3, 2, 5];

chooseMap[0][3][0] = [0, 4, 1];
chooseMap[0][3][1] = "NONE";
chooseMap[0][3][2] = [3, 3, 0];
chooseMap[0][3][3] = [1, 3, 5];
chooseMap[0][3][4] = "NONE";
chooseMap[0][3][5] = [2, 4, 2];
chooseMap[0][3][6] = [3, 2, 7];
chooseMap[0][3][7] = [1, 3, 7];
```

```
chooseMap[0][4][0] = [0, 4, 2];
chooseMap[0][4][1] = [2, 6, 1];
chooseMap[0][4][2] = [2, 6, 0];
chooseMap[0][4][3] = [1, 6, 5];
chooseMap[0][4][4] = [3, 5, 5];
chooseMap[0][4][5] = [0, 4, 5];
chooseMap[0][4][6] = "NONE";
chooseMap[0][4][7] = [2, 3, 6];

chooseMap[0][5][0] = [1, 5, 2];
chooseMap[0][5][1] = "NONE";
chooseMap[0][5][2] = [1, 7, 2];
chooseMap[0][5][3] = "NONE";
chooseMap[0][5][4] = [0, 5, 6];
chooseMap[0][5][5] = [0, 5, 7];
chooseMap[0][5][6] = [2, 7, 6];
chooseMap[0][5][7] = [1, 7, 5];

chooseMap[0][6][0] = [0, 6, 2];
chooseMap[0][6][1] = [0, 7, 4];
chooseMap[0][6][2] = [0, 7, 5];
chooseMap[0][6][3] = [0, 6, 3];
chooseMap[0][6][4] = "NONE";
chooseMap[0][6][5] = [1, 6, 7];
chooseMap[0][6][6] = "NONE";
chooseMap[0][6][7] = [2, 7, 4];

chooseMap[0][7][0] = [1, 7, 2];
chooseMap[0][7][1] = "NONE";
chooseMap[0][7][2] = [0, 7, 6];
chooseMap[0][7][3] = [0, 7, 7];
chooseMap[0][7][4] = [3, 6, 7];
chooseMap[0][7][5] = [3, 5, 7];
chooseMap[0][7][6] = [2, 5, 6];
chooseMap[0][7][7] = [2, 5, 5];

chooseMap[1][0][0] = [0, 2, 2];
chooseMap[1][0][1] = [0, 2, 1];
chooseMap[1][0][2] = [1, 2, 0];
chooseMap[1][0][3] = [1, 1, 0];
chooseMap[1][0][4] = [2, 0, 0];
chooseMap[1][0][5] = [2, 0, 1];
chooseMap[1][0][6] = "NONE";
```

```
chooseMap[1][0][7] = [3, 0, 5];

chooseMap[1][1][0] = [1, 3, 2];
chooseMap[1][1][1] = "NONE";
chooseMap[1][1][2] = [3, 1, 0];
chooseMap[1][1][3] = [0, 2, 5];
chooseMap[1][1][4] = "NONE";
chooseMap[1][1][5] = [3, 1, 3];
chooseMap[1][1][6] = [2, 0, 3];
chooseMap[1][1][7] = [2, 0, 4];

chooseMap[1][2][0] = [1, 4, 2];
chooseMap[1][2][1] = [0, 0, 1];
chooseMap[1][2][2] = [3, 0, 0];
chooseMap[1][2][3] = [0, 2, 5];
chooseMap[1][2][4] = [1, 3, 3];
chooseMap[1][2][5] = [1, 2, 3];
chooseMap[1][2][6] = "NONE";
chooseMap[1][2][7] = [3, 2, 5];

chooseMap[1][3][0] = [0, 4, 1];
chooseMap[1][3][1] = "NONE";
chooseMap[1][3][2] = [3, 3, 0];
chooseMap[1][3][3] = [1, 3, 5];
chooseMap[1][3][4] = "NONE";
chooseMap[1][3][5] = [2, 4, 2];
chooseMap[1][3][6] = [0, 1, 6];
chooseMap[1][3][7] = [3, 1, 5];

chooseMap[1][4][0] = [0, 4, 2];
chooseMap[1][4][1] = [1, 5, 0];
chooseMap[1][4][2] = [1, 4, 0];
chooseMap[1][4][3] = [3, 1, 2];
chooseMap[1][4][4] = [3, 2, 2];
chooseMap[1][4][5] = [3, 3, 2];
chooseMap[1][4][6] = "NONE";
chooseMap[1][4][7] = [2, 3, 6];

chooseMap[1][5][0] = [1, 5, 2];
chooseMap[1][5][1] = "NONE";
chooseMap[1][5][2] = [1, 7, 2];
chooseMap[1][5][3] = "NONE";
chooseMap[1][5][4] = [0, 5, 6];
```

```
chooseMap[1][5][5] = [0, 5, 7];
chooseMap[1][5][6] = [0, 3, 7];
chooseMap[1][5][7] = [3, 3, 5];

chooseMap[1][6][0] = [0, 6, 2];
chooseMap[1][6][1] = [0, 6, 1];
chooseMap[1][6][2] = [3, 6, 0];
chooseMap[1][6][3] = [0, 4, 3];
chooseMap[1][6][4] = "NONE";
chooseMap[1][6][5] = [1, 6, 7];
chooseMap[1][6][6] = "NONE";
chooseMap[1][6][7] = [2, 7, 4];

chooseMap[1][7][0] = [1, 7, 2];
chooseMap[1][7][1] = "NONE";
chooseMap[1][7][2] = [0, 7, 6];
chooseMap[1][7][3] = [0, 7, 2];
chooseMap[1][7][4] = [0, 6, 3];
chooseMap[1][7][5] = [3, 6, 2];
chooseMap[1][7][6] = [2, 7, 2];
chooseMap[1][7][7] = [2, 7, 3];

chooseMap[2][0][0] = [0, 2, 2];
chooseMap[2][0][1] = [0, 2, 1];
chooseMap[2][0][2] = [1, 2, 0];
chooseMap[2][0][3] = [1, 1, 0];
chooseMap[2][0][4] = [2, 0, 0];
chooseMap[2][0][5] = [2, 0, 1];
chooseMap[2][0][6] = ["NONE"];
chooseMap[2][0][7] = [3, 0, 5];

chooseMap[2][1][0] = [1, 1, 2];
chooseMap[2][1][1] = "NONE";
chooseMap[2][1][2] = [0, 0, 5];
chooseMap[2][1][3] = [0, 2, 4];
chooseMap[2][1][4] = "NONE";
chooseMap[2][1][5] = [2, 0, 2];
chooseMap[2][1][6] = [2, 0, 3];
chooseMap[2][1][7] = [2, 1, 5];

chooseMap[2][2][0] = [0, 0, 2];
chooseMap[2][2][1] = [0, 0, 1];
chooseMap[2][2][2] = [0, 0, 4];
```

```
chooseMap[2][2][3] = [3, 2, 3];
chooseMap[2][2][4] = [1, 3, 3];
chooseMap[2][2][5] = [3, 0, 7];
chooseMap[2][2][6] = "NONE";
chooseMap[2][2][7] = [3, 2, 5];

chooseMap[2][3][0] = [0, 4, 1];
chooseMap[2][3][1] = "NONE";
chooseMap[2][3][2] = [0, 0, 3];
chooseMap[2][3][3] = [1, 3, 5];
chooseMap[2][3][4] = "NONE";
chooseMap[2][3][5] = [0, 1, 7];
chooseMap[2][3][6] = [0, 1, 6];
chooseMap[2][3][7] = [2, 3, 5];

chooseMap[2][4][0] = [3, 4, 0];
chooseMap[2][4][1] = [1, 5, 0];
chooseMap[2][4][2] = [3, 0, 2];
chooseMap[2][4][3] = [3, 1, 2];
chooseMap[2][4][4] = [3, 2, 2];
chooseMap[2][4][5] = [1, 4, 7];
chooseMap[2][4][6] = "NONE";
chooseMap[2][4][7] = [2, 3, 6];

chooseMap[2][5][0] = [1, 5, 2];
chooseMap[2][5][1] = "NONE";
chooseMap[2][5][2] = [1, 5, 4];
chooseMap[2][5][3] = "NONE";
chooseMap[2][5][4] = [0, 5, 4];
chooseMap[2][5][5] = [0, 3, 7];
chooseMap[2][5][6] = [0, 3, 6];
chooseMap[2][5][7] = [3, 3, 5];

chooseMap[2][6][0] = [0, 6, 2];
chooseMap[2][6][1] = [0, 6, 1];
chooseMap[2][6][2] = [0, 7, 6];
chooseMap[2][6][3] = [0, 4, 4];
chooseMap[2][6][4] = "NONE";
chooseMap[2][6][5] = [1, 6, 7];
chooseMap[2][6][6] = "NONE";
chooseMap[2][6][7] = [3, 4, 5];

chooseMap[2][7][0] = [1, 7, 2];
```

```
chooseMap[2][7][1] = "NONE";
chooseMap[2][7][2] = [0, 7, 2];
chooseMap[2][7][3] = [3, 4, 2];
chooseMap[2][7][4] = [3, 5, 2];
chooseMap[2][7][5] = [2, 6, 2];
chooseMap[2][7][6] = [3, 6, 3];
chooseMap[2][7][7] = [2, 5, 5];


chooseMap[3][0][0] = [1, 2, 2];
chooseMap[3][0][1] = [2, 2, 1];
chooseMap[3][0][2] = [1, 1, 5];
chooseMap[3][0][3] = [1, 2, 5];
chooseMap[3][0][4] = [2, 2, 4];
chooseMap[3][0][5] = [2, 0, 1];
chooseMap[3][0][6] = "NONE";
chooseMap[3][0][7] = [3, 0, 5];

chooseMap[3][1][0] = [1, 1, 2];
chooseMap[3][1][1] = "NONE";
chooseMap[3][1][2] = [0, 0, 5];
chooseMap[3][1][3] = [0, 2, 4];
chooseMap[3][1][4] = "NONE";
chooseMap[3][1][5] = [1, 1, 8];
chooseMap[3][1][6] = [2, 1, 6];
chooseMap[3][1][7] = [2, 1, 5];

chooseMap[3][2][0] = [1, 4, 2];
chooseMap[3][2][1] = [2, 4, 1];
chooseMap[3][2][2] = [0, 0, 4];
chooseMap[3][2][3] = [1, 4, 5];
chooseMap[3][2][4] = [2, 4, 4];
chooseMap[3][2][5] = [1, 1, 8];
chooseMap[3][2][6] = "NONE";
chooseMap[3][2][7] = [3, 2, 5];

chooseMap[3][3][0] = [0, 4, 1];
chooseMap[3][3][1] = "NONE";
chooseMap[3][3][2] = [0, 0, 3];
chooseMap[3][3][3] = [1, 3, 5];
chooseMap[3][3][4] = "NONE";
chooseMap[3][3][5] = [3, 3, 7];
chooseMap[3][3][6] = [3, 2, 7];
```

```
chooseMap[3][3][7] = [2, 3, 5];

chooseMap[3][4][0] = [1, 6, 2];
chooseMap[3][4][1] = [2, 6, 1];
chooseMap[3][4][2] = [0, 5, 5];
chooseMap[3][4][3] = [0, 5, 6];
chooseMap[3][4][4] = [3, 4, 4];
chooseMap[3][4][5] = [1, 4, 7];
chooseMap[3][4][6] = "NONE";
chooseMap[3][4][7] = [2, 3, 6];

chooseMap[3][5][0] = [1, 5, 2];
chooseMap[3][5][1] = "NONE";
chooseMap[3][5][2] = [1, 5, 4];
chooseMap[3][5][3] = "NONE";
chooseMap[3][5][4] = [0, 5, 6];
chooseMap[3][5][5] = [1, 7, 7];
chooseMap[3][5][6] = [2, 7, 6];
chooseMap[3][5][7] = [3, 3, 5];

chooseMap[3][6][0] = [0, 7, 3];
chooseMap[3][6][1] = [0, 7, 4];
chooseMap[3][6][2] = [3, 6, 2];
chooseMap[3][6][3] = [0, 4, 3];
chooseMap[3][6][4] = "NONE";
chooseMap[3][6][5] = [1, 6, 7];
chooseMap[3][6][6] = "NONE";
chooseMap[3][6][7] = [3, 4, 5];

chooseMap[3][7][0] = [1, 7, 2];
chooseMap[3][7][1] = "NONE";
chooseMap[3][7][2] = [0, 7, 6];
chooseMap[3][7][3] = [0, 7, 7];
chooseMap[3][7][4] = [3, 6, 7];
chooseMap[3][7][5] = [3, 5, 7];
chooseMap[3][7][6] = [2, 5, 6];
chooseMap[3][7][7] = [2, 5, 5];

/*
 * Very big function that find optimal way
 * from one cell to other using bfs and non-
 * optimal dfs.
 * Function findRoad(from, to, direction)
```

```
 * return string that has format "FFRFLF"
 * with movements from first cell to second.
 * Can works very long time with big ranges.
 *
 * NOTE: there using div(n, m) function for
 * division n on m.
 */

function div(val, by) {
  return (val - val % by) / by;
}

var mapGraph = [];

for (var i = 0; i < 8 * 8; i++) {
  mapGraph = [];
}

mapGraph[0 + 0 * 8] = [2, 1, 8];
mapGraph[1 + 0 * 8] = [2, 0, 2];
mapGraph[2 + 0 * 8] = [3, 1, 3, 10];
mapGraph[3 + 0 * 8] = [2, 2, 4];
mapGraph[4 + 0 * 8] = [2, 3, 5];
mapGraph[5 + 0 * 8] = [2, 4, 13];
mapGraph[6 + 0 * 8] = [0];
mapGraph[7 + 0 * 8] = [1, 15];

mapGraph[0 + 1 * 8] = [2, 0, 16];
mapGraph[1 + 1 * 8] = [0];
mapGraph[2 + 1 * 8] = [2, 2, 18];
mapGraph[3 + 1 * 8] = [1, 19];
mapGraph[4 + 1 * 8] = [0];
mapGraph[5 + 1 * 8] = [3, 5, 14, 21];
mapGraph[6 + 1 * 8] = [2, 13, 15];
mapGraph[7 + 1 * 8] = [2, 7, 14];

mapGraph[0 + 2 * 8] = [2, 8, 17];
mapGraph[1 + 2 * 8] = [2, 16, 18];
mapGraph[2 + 2 * 8] = [3, 10, 17, 26];
mapGraph[3 + 2 * 8] = [3, 11, 20, 27];
mapGraph[4 + 2 * 8] = [2, 19, 21];
mapGraph[5 + 2 * 8] = [3, 13, 20, 29];
mapGraph[6 + 2 * 8] = [0];
```

```
mapGraph[7 + 2 * 8] = [1, 31];

mapGraph[0 + 3 * 8] = [1, 32];
mapGraph[1 + 3 * 8] = [0];
mapGraph[2 + 3 * 8] = [2, 18, 34];
mapGraph[3 + 3 * 8] = [1, 19];
mapGraph[4 + 3 * 8] = [0];
mapGraph[5 + 3 * 8] = [3, 21, 30, 37];
mapGraph[6 + 3 * 8] = [2, 29, 31];
mapGraph[7 + 3 * 8] = [3, 23, 30, 39];

mapGraph[0 + 4 * 8] = [3, 24, 33, 40];
mapGraph[1 + 4 * 8] = [2, 32, 34];
mapGraph[2 + 4 * 8] = [4, 26, 33, 35, 42];
mapGraph[3 + 4 * 8] = [2, 34, 36];
mapGraph[4 + 4 * 8] = [3, 35, 37, 44];
mapGraph[5 + 4 * 8] = [3, 29, 36, 45];
mapGraph[6 + 4 * 8] = [0];
mapGraph[7 + 4 * 8] = [1, 31];

mapGraph[0 + 5 * 8] = [1, 32];
mapGraph[1 + 5 * 8] = [0];
mapGraph[2 + 5 * 8] = [2, 34, 50];
mapGraph[3 + 5 * 8] = [0];
mapGraph[4 + 5 * 8] = [2, 36, 45];
mapGraph[5 + 5 * 8] = [4, 37, 44, 46, 53];
mapGraph[6 + 5 * 8] = [2, 45, 47];
mapGraph[7 + 5 * 8] = [2, 46, 55];

mapGraph[0 + 6 * 8] = [2, 49, 56];
mapGraph[1 + 6 * 8] = [2, 48, 50];
mapGraph[2 + 6 * 8] = [4, 42, 49, 51, 58];
mapGraph[3 + 6 * 8] = [2, 50, 59];
mapGraph[4 + 6 * 8] = [0];
mapGraph[5 + 6 * 8] = [1, 45];
mapGraph[6 + 6 * 8] = [0];
mapGraph[7 + 6 * 8] = [2, 47, 63];

mapGraph[0 + 7 * 8] = [1, 48];
mapGraph[1 + 7 * 8] = [0];
mapGraph[2 + 7 * 8] = [2, 50, 59];
mapGraph[3 + 7 * 8] = [3, 51, 58, 60];
mapGraph[4 + 7 * 8] = [2, 59, 61];
```

```
mapGraph[5 + 7 * 8] = [2, 60, 62];
mapGraph[6 + 7 * 8] = [2, 61, 63];
mapGraph[7 + 7 * 8] = [2, 55, 62];

var mapBfs = [];
for (var i = 0; i <= 63; i++) {
  mapBfs[i] = -1;
}
var queue = [];
var queueLast = 1;

function search(cell, width) {
  for (var i = 1; i <= mapGraph[cell][0]; i++) {
    if (mapBfs[mapGraph[cell][i]] == -1) {
      queue[queueLast] = mapGraph[cell][i];
      mapBfs[mapGraph[cell][i]] = width + 1;
      queueLast++;
    }
  }
}

function bfs(cell) {
  var i = 0;
  queue[0] = cell;
  mapBfs[cell] = 0;
  while (i != queueLast) {
    search(queue[i], mapBfs[queue[i]]);
    i++;
  }
}

function bfsClear() {
  for (var i = 0; i < 8 * 8; i++) {
    mapBfs[i] = -1;
    queue = [];
    queueLast = 1;
  }
}

function dfs(cell, to, range) {
  if (range != 0) {
    for (var i = 1; i <= mapGraph[cell][0]; i++) {
      road = dfs(mapGraph[cell][i], to, range - 1);
```

```
      if (road != false) {
        if (cell < 10)
          return '0' + cell + ' ' + road;
        return cell + ' ' + road;
      }
    }
  } else {
    if (cell == to) {
      if (cell < 10)
        return '0' + cell;
      return cell;
    } else {
      return false;
    }
  }
  return 0;
}

function findRoad(from, to, direction) {
  bfs(from);
  var range = mapBfs[to];
  bfsClear();
  var badRoad = dfs(from, to, range);
  var goodRoad = [];
  for (var i = 0; i < (badRoad.length + 1) / 3; i++) {
    goodRoad[i] = [];
    goodRoad[i][0] = div(+(badRoad[i * 3] + badRoad[i * 3 + 1]), 8);
    goodRoad[i][1] = (+(badRoad[i * 3] + badRoad[i * 3 + 1]) % 8);
  }
  var result = "";
  for (var i = 1; i < goodRoad.length; i = i + 1 - 1) {
    if (goodRoad[i][1] == goodRoad[i - 1][1] - 1) {
      if (direction == 2) {
        result += "F";
        i++;
        continue;
      }
      if (direction == 1) {
        result += "R";
        direction = (direction + 1) % 4;
        continue;
      }
      if (direction == 3) {
```

```
      result += "L";
      direction = (direction + 3) % 4;
      continue;
    }
    if (direction == 0) {
      result += "RR";
      direction = (direction + 2) % 4;
      continue;
    }
  }
  if (goodRoad[i][1] == goodRoad[i - 1][1] + 1) {
    if (direction == 0) {
      result += "F";
      i++;
      continue;
    }
    if (direction == 3) {
      result += "R";
      direction = (direction + 1) % 4;
      continue;
    }
    if (direction == 1) {
      result += "L";
      direction = (direction + 3) % 4;
      continue;
    }
    if (direction == 2) {
      result += "RR";
      direction = (direction + 2) % 4;
      continue;
    }
  }
  if (goodRoad[i][0] == goodRoad[i - 1][0] - 1) {
    if (direction == 3) {
      result += "F";
      i++;
      continue;
    }
    if (direction == 2) {
      result += "R";
      direction = (direction + 1) % 4;
      continue;
    }
```

```
      if (direction == 0) {
        result += "L";
        direction = (direction + 3) % 4;
        continue;
      }
      if (direction == 1) {
        result += "RR";
        direction = (direction + 2) % 4;
        continue;
      }
    }
    if (goodRoad[i][0] == goodRoad[i - 1][0] + 1) {
      if (direction == 1) {
        result += "F";
        i++;
        continue;
      }
      if (direction == 0) {
        result += "R";
        direction = (direction + 1) % 4;
        continue;
      }
      if (direction == 2) {
        result += "L";
        direction = (direction + 3) % 4;
        continue;
      }
      if (direction == 3) {
        result += "RR";
        direction = (direction + 2) % 4;
        continue;
      }
    }
  }
  return result;
}

var pi = 3.1415926535897931;
var d = 8.25; //8.7; //diameter of wheel, cm
var l = 17.5; // base of robot
var degInRad = 180 / pi;

var alpha = 0;
```

```
var readGyro = brick.gyroscope().read

function readYaw() {
  return -readGyro()[6];
}
var fullTurn = 4 * 87; // from experimets. Not 360 degrees, but ....

//кнопки
var stopKey = KeysEnum.Esc;
var startKey = KeysEnum.Left;

//диод
var led = brick.led();

//моторы
var mLeft = brick.motor(M3).setPower;
var mRight = brick.motor(M4).setPower;

//сенсоры ИК
irLeft = brick.sensor(A2).read;
irRight = brick.sensor(A1).read;

//датчики света
lightLeft = brick.sensor(A5).read;
lightRight = brick.sensor(A6).read;

//Сенсор US
usForward = brick.sensor(D2).read;
usBack = brick.sensor(D1).read;
//энкодеры
var eLeft = brick.encoder(E3);
var eRight = brick.encoder(E4);
var cpr = 374; //560; //374; // количество сигналов на оборот

//длина клетки
var cellLength = 35 * cpr / (pi * d);

//переопределение метода нажатия клавиши
if (0) {
  var key = 0;

  function wasPressed(_key) {
```

```
    key = 0;
    if (brick.keys().wasPressed(_key)) {
      key = _key;
      return 1;
    } else {
      return 0;
    }
  }
}

var direction_angle = 0; // absolute angle of direction movement
var directionOld = 0;

print("-------------------------------------------");

led.red();

eLeft.reset();
eRight.reset();

var el = eLeft.readRawData();
var er = eRight.readRawData();
mLeft(0);
mRight(0);
print("Start");

//инициализация и калибровка гироскопа
print("gyro initialaize...");
brick.gyroscope().calibrate(4000);
script.wait(4000);
script.wait(1000);

var v = 20; // velocity

var ex = 0;
var ey = 0;
var encLeftOld = 0;
var encRightOld = 0;
var encLeft = 0;
var encRight = 0;
var n = 0;
var ldata = [];
var edata = [];
```

```
// вычисление абсолютного угла относительно начального положения
function angle() {
  var sgn = 0;
  var _direction = readYaw(); // mgrad
  var dtDirection = _direction - directionOld;

  sgn = directionOld == 0 ? 0 : directionOld /
Math.abs(directionOld);
  n += sgn * Math.floor(Math.abs(dtDirection / 320000));
  direction_angle = _direction + n * 360000;
  directionOld = _direction;
}

var t = 0;

print("Run, robot, run!");


// делаем препывание основной программы с частотой 200Гц, чтобы
посчитать абсолютный угол с гироскопа
var mtimer = script.timer(50);
mtimer.timeout.connect(angle);

//поворот на угол по гироскопу
function turnDirection(_angle, _v) {
  _angle = _angle * 1000; //toMGrad
  var _vel = _v == undefined ? 10 : _v;
  var angleOfRotate = _angle - direction_angle;
  print("ar=" + angleOfRotate);
  var sgn = angleOfRotate == 0 ? 0 : angleOfRotate / Math.abs(
    angleOfRotate);
  mLeft(_vel * sgn);
  mRight(-_vel * sgn);
  while (Math.abs(angleOfRotate = _angle - direction_angle) > 20000)
{
    script.wait(50);
  }
  brick.motor(M3).forceBreak(500);
  brick.motor(M4).forceBreak(500);
  script.wait(500);
}
black = false;
```

```
z = 0;
// проезд в перед на количество ячеек _kceel со скоростью _v выравниваясь
по гироскопу на угол _alpha

function forward(_alpha, _v, _kcell) {
  eLeft.reset();
  eRight.reset();
  var _vel = _v == undefined ? 10 : _v;
  var u = 0;
  var el = Math.abs(eLeft.readRawData());
  var irLeftNorm = 0;
  var irRightNorm = 0;
  while (Math.abs(eLeft.readRawData()) < el + (_kcell * cellLength))
{
    err_irLeft = irLeftNorm - irLeft();
    u = -0.7 * (_alpha - readYaw() / 1000);
    z++;
    edata[z] = eLeft.readRawData();
    ldata[z] = (lightLeft() + lightRight()) / 2;
    script.wait(5);
    mLeft(_vel - u);
    mRight(_vel + u);
    script.wait(5);
    if (usForward() < 7) {
      break;
    }
  }
  brick.motor(M3).forceBreak();
  brick.motor(M4).forceBreak();
  script.wait(300);
}
// проезд назад, чтобы упереться в стенку с целью выравнивания
function backward() {
  mLeft(-35);
  mRight(-35);
  var eLOld = eLeft.readRawData();
  var eROld = eRight.readRawData();
  var el = eLOld;
  var er = eROld;
  do {
    eLOld = el;
    eROld = er;
    script.wait(1000);
```

```
    el = eLeft.readRawData();
    er = eRight.readRawData();
  }
  while (Math.abs(eLOld - el) > 200 && Math.abs(eROld - er) > 200)
  brick.motor(M3).forceBreak();
  brick.motor(M4).forceBreak();
  script.wait(300);
  eLeft.reset();
  eRight.reset();

}

function FORWARD() {
  forward(0, 35, 1);
  script.wait(500);
}

function RIGHT() {
  turnDirection(-fullTurn / 4, 37);
  script.wait(500);
  if (usBack() < 30) {
    backward();
    brick.gyroscope().calibrate(3000);
    script.wait(3100);
    forward(0, 35, 0.2);
    script.wait(100);
  } else {
    brick.gyroscope().calibrate(3000);
    script.wait(3000);
  }
}

function LEFT() {
  turnDirection(fullTurn / 4, 37);
  script.wait(500);
  if (usBack() < 30) {
    backward();
    brick.gyroscope().calibrate(3000);
    script.wait(3100);
    forward(0, 35, 0.2);
    script.wait(100);
  } else {
    brick.gyroscope().calibrate(3000);
```

```
    script.wait(3000);
  }
}

function move_to_finish(way) {
  print("YOO");
  for (var i = 0; i < way.length - 1; i++) {
    if (way[i] == 'F') {
      FORWARD();
      if (myDirection == 0)
        myCoords[1]++;
      if (myDirection == 1)
        myCoords[0]++;
      if (myDirection == 2)
        myCoords[1]--;
      if (myDirection == 2)
        myCoords[0]--;
      continue;
    }
    if (way[i] == 'R') {
      RIGHT();
      myDirection = (myDirection + 1) % 4;
      continue;
    }
    if (way[i] == 'L') {
      LEFT();
      myDirection = (myDirection + 3) % 4;
      continue;
    }
  }
}

var readCode = function() {
  mLeft(0);
  mRight(0);
  var d1 = direction_angle;

  var i = 0;
  var _alpha = 0;
  var _vel = 25;
  var u = 0;

  // едем до калибровочной линии
```

```
  while ((lightLeft() < 90) && (lightRight() < 90)) {
    u = -0.6 * (_alpha - readYaw() / 1000);
    mLeft(_vel - u);
    mRight(_vel + u);
    script.wait(5);
  }

  _vel = 15;
  var _encOne = 27; // выяснено имперически
  encLeft = eLeft.readRawData();
  encLeftOld = encLeft;
  var twoInPower = 8;
  var numb = 0;
  var count = 1;
  var sv;
  // считывам значений 4х битов
  while (count < 5) {
    if (usForward() < 5) {
      break;
    }
    u = -0.5 * (_alpha - readYaw() / 1000);
    mLeft(_vel - u);
    mRight(_vel + u);
    encLeft = eLeft.readRawData();
    sv = lightRight();

    // мы считываем значение белая/черная линия с частотой дискретизации
_encOne
    if (Math.abs(encLeft - encLeftOld) >= count * _encOne) {
      count++;
      if (sv > 90) {
        numb += twoInPower;
      }
      twoInPower = twoInPower / 2;
    }
    script.wait(3);

  }
  return (numb);
}

var myCoords = [];
```

```
var myDirection;

function localization() {
  var currentMap = [];
  var sector = 0;
  while (true) {
    print(sector);
    currentMap[sector] = [];
    if (usForward() > 30)
      currentMap[sector][0] = 0;
    else
      currentMap[sector][0] = 1;

    if (irRight() > 30)
      currentMap[sector][1] = 0;
    else
      currentMap[sector][1] = 1;

    if (usBack() > 30)
      currentMap[sector][2] = 0;
    else
      currentMap[sector][2] = 1;

    if (irLeft() > 30)
      currentMap[sector][3] = 0;
    else
      currentMap[sector][3] = 1;

    // End of scan
    if (sector == 4)
      break;

    if (currentMap[sector][1] == 0) {
      RIGHT();
      FORWARD();
    } else if (currentMap[sector][0] == 0) {
      FORWARD();
    } else if (currentMap[sector][3] == 0) {
      LEFT();
      FORWARD();
    } else {
      LEFT();
      LEFT();
```

```
      FORWARD();
    }
    sector++;
  }

  var collisions = 0;
  var coords = []; // d, y, x
  for (var d = 0; d < 4; d++) {
    for (var y = 0; y < 8; y++) {
      for (var x = 0; x < 8; x++) {
        var isNorm = true;
        for (var sector = 0; sector <= 4; sector++) {
          for (var dir = 0; dir < 4; dir++) {
            if (Map[d][y][x] == "NONE") {
              isNorm = false;
              continue;
            }
            if (Map[d][y][x][sector][dir] != currentMap[sector][dir])
              isNorm = false;
          }
        }
        if (isNorm) {
          collisions++;
          coords = [d, x, y];
        }
      }
    }
  }
  print(collisions);
  if (collisions == 1) {
    myDirection = chooseMap[coords[0]][coords[2]][coords[1]][0];
    myCoords[0] = chooseMap[coords[0]][coords[2]][coords[1]][1];
    myCoords[1] = chooseMap[coords[0]][coords[2]][coords[1]][2];
    print(chooseMap[coords[0]][coords[1]][coords[2]]);
  } else {
    localization();
  }
}

localization();
brick.playSound("media/beep.wav")
var road1 = findRoad(myCoords[0] * 8 + myCoords[1], 15, myDirection);
var road2 = findRoad(myCoords[0] * 8 + myCoords[1], 44, myDirection);
```

```
var road;
if (road1 == "") {
  road = road2;
} else if (road2 == "") {
  road = road1;
} else if (road1.length < road2.length) {
  road = road1;
} else {
  road = road2;
}
print(road);
move_to_finish(road);
var firstNumber = readCode();
print(firstNumber);
if (myDirection == 0)
  myCoords[1]++;
if (myDirection == 1)
  myCoords[0]++;
if (myDirection == 2)
  myCoords[1]--;
if (myDirection == 2)
  myCoords[0]--;

brick.playSound("media/beep.wav");
fCoods = [];
fCoods[div(firstNumber, 8)] = firstNumber % 8;
brick.display().addLabel(firstNumber % 8, 40, 40);
brick.display().redraw();
script.wait(10000);
```

```
var road1 = findRoad(myCoords[0] * 8 + myCoords[1], 15, myDirection);
var road2 = findRoad(myCoords[0] * 8 + myCoords[1], 44, myDirection);
var road;
if (road1 == "") {
  road = road2;
} else if (road2 == "") {
  road = road1;
} else if (road1.length < road2.length) {
  road = road1;
} else {
  road = road2;
```

```
}
print(road);
move_to_finish(road);
var secondNumber = readCode();
print(secondNumber);
if (myDirection == 0)
  myCoords[1]++;
if (myDirection == 1)
  myCoords[0]++;
if (myDirection == 2)
  myCoords[1]--;
if (myDirection == 2)
  myCoords[0]--;

brick.playSound("media/beep.wav");
fCoods[div(secondNumber, 8)] = secondNumber % 8;
brick.display().addLabel("X:" + fCoords[1], 10, 10);
brick.display().addLabel("Y:" + fCoords[0], 30, 10);
brick.display().redraw();

script.wait(10000);


road = findRoad(myCoords[0] * 8 + myCoords[1], fCoods[0] * 8 +
fCoods[
  1], myDirection);
move_to_finish(road);
FORWARD();
```