§2 Второй отборочный этап

Второй отборочный этап проводится в командном формате в сети интернет, работы оцениваются автоматически средствами системы онлайн-тестирования. Продолжительность второго этапа составляет 2 недели. Задачи носят междисциплинарный характер и в более простой форме воссоздают инженерную задачу заключительного этапа. Решение задач подразумевает написание программ на языке Python или C++. Решение каждой задачи дает определенное количество баллов. Баллы зачисляются в полном объеме за правильное решение задачи. В данном этапе можно получить суммарно от 0 до 25 баллов.

2.1. Программирование робота

Вам предстоит написать программу, моделирующую управление промышленным роботом, который занимается сбором грузов на складе. Робот программируется на языке Python. Для управления используется набор команд, который будет поробно описан в двух следующих подзадачах.

Вы управляете роботом, который реализован как экземпляр `Robot`. Все действия и информацию о роботе можно получить, вызывая методы этого экземпляра обращаясь к его атрибутам. Робот перемещается по складу, реализованному как экземпляр `Stock` (см. далее), представляющему из себя поле, разбитое на квадратные клетки, на которых расположены ящики и препятствия. Робот может перемещаться по горизонтали или вертикали на соседние клетки и собирать ящики. Задача — собрать все ящики, которые находятся на складе.

Далее дается подробное описание API, использующегося для управления роботом.

У экземпляра Robot есть следующие аттрибуты:

- `Robot.x`, `Robot.y` - координаты по "вертикали" и "горизонтали" соответственно. Начальные кординаты - (1, 1); могут принимать значения от 1 до 10.
- `Robot.direction` - текущее направление движения робота. Может принимать значения `'R'`, `'L'`, `'U'`, `'D'` (относительно игрового поля). Начальное значение - `'R'`.
- `Robot.collected` - количество собранных на данный момент ящиков. Изначально равно нулю.

Важно! Обращение к атрибутам в вашем коде, в отличие от методов, не расходует энергии, так как при этом робот не «совершает» действий.

Есть следующие методы для управления роботом (вызов каждого из методов

расходует одну единицу энергии робота):

- `Robot.step_forward()` - переместиться на одн клетку вперед.
- `Robot.step_backward()` - переместиться на одну клетку назад.
- `Robot.turn_right()` - поворот на 90 градусов направо.
- `Robot.turn_left()` - поворот на 90 нрадусов налево.
- `Robot.look_self()` - определить, что находится на клекте под роботом. Возвращает 0 в случае, если клетка пустая, и 2 в случае, если на ней находится ящик.
- `Robot.look_forward()` - определить, что находится на клетке перед роботом. Возвращает 0 в случае, если клетка пустая, 1, если на ней расположено препятствие, 2, если на ней находится ящик, и 3, если перед роботом стена.
- `Robot.collect()` - забрать ящик с клетки, если он присутствует. Иначе произойдет ошибка.

Пример склада (экземпляр `Stock`, см. далее):

```
+ - - - - - - - - - - +
| > x . . . # x . . . |
| . # . . . . . # . x |
| # . . . . . . . . . |
| . # . . . . . . . . |
| . . # x # . . . . . |
| . . . . . . . . . . |
| . # . x . . . . . . |
| . . . . . . x . x . |
| x . . . # . # . . . |
| . . x . . x . . . . |
+ - - - - - - - - - - +
```

Здесь `'+'`, `'-'`, `'|'` - стены, `'>'` - начальное положение робота (на этой клетке так же может быть расположен ящик!), `'.'` - пустые клетки, `'x'` - блоки-препятствия, и `'#'` - ящики.

Теперь поговорим о складе. Склад реализован в форме экземпляра класса `Stock`. Размеры склада - 10x10, со всех сторон он окружен стенами. При каждом решении задачи склад генерируется случайным образом. При выполнении дейтвий роботом состояние состояние экземпляра Stock меняется автоматически. У экземпляра Stock есть следующие аттрибуты:

`Stock.field` - двумерный массив `NumPy` размерами 12x12, который представляет собой карту склада (включая стены). Массив заполнен числами от 0 до 3; 0 - пустая клетка, 1 - клетка, занятая препятствием; 2 - клетка, в которой находится ящик; 3 - стена склада. Можно обращаться к элементам массива по их индексам, в том числе, передавая в качестве индексов координаты робота, например:

- `Stock.field[1, 1]` - то, что находится в клетке в верхнем левом углу склада.
- `Stock.field[Robot.x, Robot.y]` - то, что находится на клетке с текущим положением робота.
- `Stock.size` - размер склада, исключая стены.
- `Stock.targets` - текущее количество ящиков на поле.
- `Stock.blocks` - число блоков на карте.

Важно! Атрибуты `Stock.field`, методы `Robot.look_self()`, `Robot.look_forward()` оперируют целочисленными значениями 0, 1, 2, 3!

Задача 2.1.1 (1 балл)

Это простая задача, которая позволит вам освоить основные команды управления роботом. В данной задаче требуется добраться до ящика, который расположен на клетке с координатами (10, 10), подобрать на ней ящик и вернуть робота в начальную точку.

Образец программы:

```
1. while True:
2.     if Robot.look_self() == 2:
3.         Robot.collect()
4.         break
5.     else:
6.         Robot.step_forward()
7.         continue
```

Этот образец не является правильным решением задачи!

Поле для задачи:

```
+ - - - - - - - - - - +
| > . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
```

```
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . . |
| . . . . . . . . . # |
+ - - - - - - - - - - +
```

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для проверки результата используется следующий код на языке Python:

```
def check(reply, clue):
    rep = [int(i) for i in reply.split()]
    slv = [int(i) for i in clue.split()]

    gathered = []
    checked = []

    for i in range(int(len(rep)/2)):
        gathered.append((rep[i*2], rep[i*2+1]))
    for i in range(int(len(slv)/2)):
        checked.append((slv[i*2], slv[i*2+1]))

    for elem in checked:

        if (elem in gathered):
            pass
        else:
            return False

    return True
```

РЕШЕНИЕ:

Пример программы, решающей данную задачу на языке Python:

```
1. for i in range(9):
2.    Robot.step_forward()
3. Robot.turn_right()
4. for i in range(9):
5.     Robot.step_forward()
6. Robot.collect()
```

Задача 2.1.2 (3 балла)

На складе размерами 10x10, окруженном стенами, в случайном порядке

расположены 10 ящиков и 10 препятствий. Напишите программу, которая позволит собрать все ящики, находящиеся на складе. Ваша программа должна пройти десять тестов со случайно сгенерированными складами. Пример склада приведен на изображении.

Пример поля:

```
+ - - - - - - - - - - +
| > . . . . x . # . . |
| . . . . . . . . # . |
| # . . x . # . . . . |
| . x . # . . . . . x |
| . # . . x . . . . # |
| x # . . . . x . . x |
| . . . . . . . . . . |
| . . # . . . . . . . |
| . x . . . . . . . . |
| . . . # . . . x . . |
+ - - - - - - - - - - +
```

Ограничение на время выполнения программы: 15 с

Ограничение на используемый программой объем памяти: 256 Мб

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для проверки результата используется следующий код на языке Python:

```
def check(reply, clue):
    rep = [int(i) for i in reply.split()]
    slv = [int(i) for i in clue.split()]

    gathered = []
    checked = []

    for i in range(int(len(rep)/2)):
        gathered.append((rep[i*2], rep[i*2+1]))
    for i in range(int(len(slv)/2)):
        checked.append((slv[i*2], slv[i*2+1]))

    for elem in checked:
        if (elem in gathered):
            pass
        else:
            return False
```

```
    return True
```

РЕШЕНИЕ:

Пример программы, решающей данную задачу на языке Python:

```
01. task = [(1,1)]
02. for xk, x in enumerate(list(Stock.field)):
03.    for yk, y in enumerate(x):
04.        if y == 2:
05.            task.append((xk, yk))
06.
07. inf = float('inf')
08.
09. def volna(xy, impmaps):
10.     global Stock
11.
12.     ans = {}
13.     nn = {}
14.     nn[xy] = []
15.     queue = set([xy])
16.     ms = []
17.     while len(queue) > 0:
18.         elem = queue.pop()
19.         for px, py in ((+1,0), (-1,0), (0,+1), (0,-1)):
20.             nx, ny = elem[0] + px, elem[1] + py
21.             if (nx, ny) in nn or Stock.field[nx][ny] == 3 or Stock.field[nx]
[ny] == 1: continue
22.             nn[(nx, ny)] = nn[elem] + [elem]
23.             queue.add((nx, ny))
24.
25.     for i in impmaps:
26.         ans[i] = nn.get(i, inf)
27.         if i in nn: ms.append(i)
28.
29.    return(ans, ms)
30.
31. history = {}
32. neighbors = {}
33.
34. for z in task:
35.     history[z], neighbors[z] = volna(z, task )
36.
```

```
37. kk = {}
38. for key, i in enumerate(task):
39.     kk[i] = key
40.
41. nvv = {}
42. def rf(xy, hist):
43.     donev = tuple(sorted(set(hist)))
44.     nvv.setdefault(xy, {})
45.     if donev in nvv[xy]:
46.         return()
47.     if nvv[xy].get(donev, -1) == -1 or len(hist + (xy,)) <
len(nvv[xy].get(donev, 0)):
48.         nvv[xy][donev] = hist + (xy,)
49.     for i in neighbors[task[xy]]:
50.         rf(kk[i], hist+(xy,))
51.
52. rf(0, ())
53. for i in nvv:
54.     i = nvv[i]
55.     if tuple(range(11)) in i:
56.         transport = i[tuple(range(11))]
57.         break
58.
59. def rd(n, rd):
60.     if rd == n: return(1)
61.         if tuple(sorted([n, rd])) == ('L', 'R') or tuple(sorted([n, rd])) ==
('D', 'U'): return(-1)
62.
63.         p = list('URDL')
64.         ar =  p.index(n)
65.         if p[(ar+1)%4] == rd:
66.             Robot.turn_right()
67.         else:
68.             Robot.turn_left()
69.         return(1)
70.
71. def navig(a, b):
72.     ra = history[a][b] + [b]
73.     for i in range(1, len(ra) ):
74.         bast = ra[i-1]
75.         goal = ra[i]
```

```
76.         if bast[0] < goal[0]: #DOWN
77.             u = rd(Robot.direction, 'D')
78.         elif bast[0] > goal[0]: #UP
79.             u = rd(Robot.direction, 'U')
80.         elif bast[1] < goal[1]: #R
81.             u = rd(Robot.direction, 'R')
82.         elif bast[1] > goal[1]:#l
83.             u = rd(Robot.direction, 'L')
84.
85.         if u == 1:
86.             Robot.step_forward()
87.         else:
88.             Robot.step_backward()
89.
90. for tsk in range(1, len(transport)):
91.     back = transport[tsk-1]
92.     now = transport[tsk]
93.     navig(task[back], task[now])
94.     if Robot.look_self() == 2: Robot.collect()
```

Задача 2.1.3 (5 баллов)

Данная задача не отличается от предыдущей, кроме одного условия. К экземпляру `Robot` был добавлен атрибут, показывающий количество запасенной роботом энергии:

`Robot.energy`

начальное значение `Robot.energy` = 1000. Каждое отдельное действие приводит к снижению запаса энергии на единицу. Список действий, снижающих запас энергии на 1:

- `Robot.step_forward()`
- `Robot.step_backward()`
- `Robot.turn_left()`
- `Robot.turn_right()`
- `Robot.look_self()`
- `Robot.look_forward()`
- `Robot.collect()`

Если запас энергии упадет ниже нуля (`Robot.energy` < 0), то робот остановится.

Ваша программа должна пройти десять тестов со случайно сгенерированными складами, во время каждого теста робот должен собрать все ящики со склада, пока не

истощился запас энергии. Пример склада приведен на изображении.

Пример поля:

```
+ - - - - - - - - - - +
| > . . . . x . # . . |
| . . . . . . . . # . |
| # . . x . # . . . . |
| . x . # . . . . . x |
| . # . . x . . . . # |
| x # . . . . x . . x |
| . . . . . . . . . . |
| . . # . . . . . . . |
| . x . . . . . . . . |
| . . . # . . . x . . |
+ - - - - - - - - - - +
```

Ограничение на время выполнения программы: 20 с

Ограничение на используемый программой объем памяти: 256 Мб

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для проверки результата используется следующий код на языке Python:

```
def check(reply, clue):
    rep = [int(i) for i in reply.split()]
    slv = [int(i) for i in clue.split()]

    gathered = []
    checked = []

    for i in range(int(len(rep)/2)):
        gathered.append((rep[i*2], rep[i*2+1]))
    for i in range(int(len(slv)/2)):
        checked.append((slv[i*2], slv[i*2+1]))

    for elem in checked:
        if (elem in gathered):
            pass
        else:
            return False
```

```
    return True
```

РЕШЕНИЕ:

Обе задачи можно решить одним алгоритмом, вторая является более сложным вариантом первой. Программа на языке Python, решающая данную задачу:

```
01. task = [(1,1)]
02. for xk, x in enumerate(list(Stock.field)):
03.    for yk, y in enumerate(x):
04.        if y == 2:
05.            task.append((xk, yk))
06.
07. inf = float('inf')
08.
09. def volna(xy, impmaps):
10.     global Stock
11.
12.     ans = {}
13.     nn = {}
14.     nn[xy] = []
15.     queue = set([xy])
16.     ms = []
17.     while len(queue) > 0:
18.         elem = queue.pop()
19.         for px, py in ((+1,0), (-1,0), (0,+1), (0,-1)):
20.             nx, ny = elem[0] + px, elem[1] + py
21.             if (nx, ny) in nn or Stock.field[nx][ny] == 3 or Stock.field[nx]
[ny] == 1: continue
22.             nn[(nx, ny)] = nn[elem] + [elem]
23.             queue.add((nx, ny))
24.
25.     for i in impmaps:
26.         ans[i] = nn.get(i, inf)
27.         if i in nn: ms.append(i)
28.
29.    return(ans, ms)
30.
31. history = {}
32. neighbors = {}
33.
34. for z in task:
35.     history[z], neighbors[z] = volna(z, task )
```

```
36.
37. kk = {}
38. for key, i in enumerate(task):
39.     kk[i] = key
40.
41. nvv = {}
42. def rf(xy, hist):
43.     donev = tuple(sorted(set(hist)))
44.     nvv.setdefault(xy, {})
45.     if donev in nvv[xy]:
46.         return()
47.     if nvv[xy].get(donev, -1) == -1 or len(hist + (xy,)) <
len(nvv[xy].get(donev, 0)):
48.         nvv[xy][donev] = hist + (xy,)
49.     for i in neighbors[task[xy]]:
50.         rf(kk[i], hist+(xy,))
51.
52. rf(0, ())
53. for i in nvv:
54.     i = nvv[i]
55.     if tuple(range(11)) in i:
56.         transport = i[tuple(range(11))]
57.         break
58.
59. def rd(n, rd):
60.     if rd == n: return(1)
61.         if tuple(sorted([n, rd])) == ('L', 'R') or tuple(sorted([n, rd])) ==
('D', 'U'): return(-1)
62.
63.         p = list('URDL')
64.         ar =  p.index(n)
65.         if p[(ar+1)%4] == rd:
66.             Robot.turn_right()
67.         else:
68.             Robot.turn_left()
69.         return(1)
70.
71. def navig(a, b):
72.     ra = history[a][b] + [b]
73.     for i in range(1, len(ra) ):
74.         bast = ra[i-1]
```

```
75.         goal = ra[i]
76.         if bast[0] < goal[0]: #DOWN
77.             u = rd(Robot.direction, 'D')
78.         elif bast[0] > goal[0]: #UP
79.             u = rd(Robot.direction, 'U')
80.         elif bast[1] < goal[1]: #R
81.             u = rd(Robot.direction, 'R')
82.         elif bast[1] > goal[1]:#l
83.             u = rd(Robot.direction, 'L')
84.
85.         if u == 1:
86.             Robot.step_forward()
87.         else:
88.             Robot.step_backward()
89.
90. for tsk in range(1, len(transport)):
91.     back = transport[tsk-1]
92.     now = transport[tsk]
93.     navig(task[back], task[now])
94.     if Robot.look_self() == 2: Robot.collect()
```

2.2 Приём цифрового ИК-сигнала

В этой задаче предлагается смоделировать работу АЦП и расшифровку ИК-сигнала.

На вход программе подаётся последовательность чисел с плавающей десятичной точкой (float), описывающих последовательные измерения напряжения (в вольтах) на выходе датчика (приёмного фотоэлемента). Числа находятся в диапазоне от 0 до 5 вольт.

На выходе программа должна выдать содержимое (в виде текстовой строки) принятого пакета, если таковой обнаружен, или пустую строку, если не обнаружен, или строку «CRC_ERROR», если пакет был обнаружен и декодирован, но контрольная сумма не сошлась или отсутствует проверочный бит.

Упомянутая входная последовательность состоит из следующих элементов:

- «0» – последовательность значений, близких к нулю, меньших, чем V/5;
- «1» – последовательность значений, близких к V, и больших, чем 4/5 V.

Последовательность начинается и заканчивается последовательностью «0» («паузой»), при декодировании они должны быть проигнорированы. Если в сигнале присутствует пакет, то он состоит из «0» и «1», при этом все биты «0» и «1» в пакете имеют примерно одинаковую длину (возможен разброс не более чем на 1/16 от длины бита). Между

паузами и самим пакетом может присутствовать сигнал, не несущий полезной информации.

Полностью структура последовательности следующая:

1. Стартовая пауза (последовательность «0», может отсутствовать), должна быть проигнорирована.
2. Произвольная комбинация значений в диапазоне от 0 до 5 В (шум), которая должна быть проигнорирована.
3. Калибровочная последовательность из 4 байтов 0xAA (10101010), то есть 16 раз повторённая комбинация «1»-«0».
4. Один проверочный байт 0x53 (01010011).
5. Один байт длины контента.
6. Столько байт контента, сколько указано в байте длины. В качестве контента используется связный текст (латиницей) в кодировке ASCII с использованием значений от 32 до 127.
7. Байт контрольный суммы, являющийся XOR-суммой всех байтов контента и байта длины.
8. Произвольная комбинация значений в диапазоне от 0 до 5 (шум), которая должна быть проигнорирована.
9. Финальная пауза (последовательность «0», может отсутствовать).

Задача 2.2.1 (5 баллов)

Напишите программу на языке C++, которая расшифровывает последовательность, описанную в предыдущем шаге, и пишет в стандыртный вывод закодированное сообщение, если оно было успешно принято и декодировано, пустую строку «», если в принятом сигнале отсутствует сообщение (не удалось найти калибровочную последовательность), или строку «`CRC_ERROR`», если при проверке контрольной суммы/проверочного байта произошла ошибка.

Доступ к сигналу, заданному набором чисел с плавающей точкой, можно получить, вызвав метод

```
int Signal.get(float *);
```

класса `Signal`. Каждый вызов данного метода записывает следующее в последовательности значение напряжения в переданную по указателю переменную типа `float`. Если последовательность (т. е. входящий сигнал) еще не кончилась, метод возвращает 0. Если вся последовательность была считана и следующее значение отсутствует, то будет возвращена 1. В задаче переданный сигнал представлен объектом `signal`.

Ограничение на время исполнения программы: 5 с

Ограничение на объем используемой памяти: 256 Мб

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
class Signal ():
    def __init__ (self, text):
        self.text = text
        self.arr = [int(i) for i in ''.join(['{0:08b}'.format(ord(i)) for i in
text])]
        self.byted = [ord(i) for i in text]
    def output (self):
        print (''.join(str(i) for i in self.arr), '\n', self.text)

    def scatter (self, i):
        if i == 1:
            return -1
        else:
            return 1

    def sample (self, volt = 5, freq = rd.randint(8, 32)):
        shift = int(freq / 16);
        self.bits = []
        self.keyargs = []
        for i in self.arr:
            if rd.randint(0, 1) == 0: bit = (freq + rd.randint(0, shift))
            else: bit = (freq - rd.randint(0, shift))

            self.keyargs.append(bit)
            for j in [i]*bit:
                self.bits.append(j)
        self.signal = [(i + self.scatter(i)*rd.random()/5.) * volt for i in
self.bits]

def mygenerate (text, cal = 'y', chk = 'y', lng = 'y', bit = 'y', txt = 'y'):
    t_len = len(text)
    if t_len < 32:
        text = text + ' '*(32 - t_len)

    templ = Signal(text)
```

```
    if lng == 'y':
        length = len(templ.byted)
    else:
        length = rd.randint(32,127)
    if chk == 'y':
        checks = length
        for i in templ.byted:
            checks ^= i
    else:
        checks = rd.randint(0, 255)
    if cal == 'y':
        calib = '\xaa'*4
    else:
        calib = ['\xaa'*2 + chr(rd.randint(0, 255)) + '\xaa',\
            '\xaa' + chr(rd.randint(0, 255)) + '\xaa'*2,\
            '\xaa' + chr(rd.randint(0, 255))*2 + '\xaa',\
            chr(rd.randint(0, 255)) + '\xaa'*3,\
            '\xaa'*3 + chr(rd.randint(0, 255)),\
            chr(rd.randint(0, 255))*3 + '\xaa']

        calib = calib[rd.randint(0,5)]
    if bit == 'y':
        ctrl = '\x53'
    else:
        ctrl = chr(rd.randint(80, 95))

    if txt != 'y':
        text = ''.join([chr(rd.randint(0x21, 0x7e)) for i in
range(rd.randint(32, 127))])

    zero_beg = rd.randint(0, 255)
    zero_fin = rd.randint(0, 255)

    beg = ''.join(chr(0)*zero_beg) + ''.join([chr(rd.randint(0x21, 0x7e)) for i
in range(255-zero_beg + rd.randint(0, 64))])
    fin = ''.join([chr(rd.randint(0x21, 0x7e)) for i in range(255-zero_fin +
rd.randint(0, 64))]) + ''.join(chr(0)*zero_fin)

    return beg + calib + ctrl + chr(length) + text + chr(checks) + fin
    #beg + cal + ctrl + chr(length) + text + chr(checks) + fin
def conv(i):
```

```
    if  0 <= i <=1: return 0
    elif 4 <=i <= 5: return 1
    else: return -1

    # get sample arr {0, 1} and ptr, returns [ptr, bitlen] if ok
    # or [ptr, 0] if not ok
def calibration(sample, ptr):
    n_changes = 0
    bitlen = []
    ptr_buf = ptr

    for i in range(2):
        if  ptr + 1 > len(sample): return [ptr, 0]
        try:
            while sample[ptr] == sample[ptr+1]: ptr += 1
        except IndexError:
            #if there's no calibration sequense
            return (len(sample), 0)
        ptr += 1
    next_ptr = ptr
    ptr = ptr_buf

    while (n_changes < 32):
        while  ptr+1 < len(sample) and sample[ptr] == sample[ptr+1]: ptr += 1
        ptr += 1
        bitlen.append(ptr - ptr_buf)
        ptr_buf = ptr
        n_changes += 1

    flag = True
    for i in bitlen[1:-1]:
        if abs(i - np.mean(bitlen[1:-1])) > np.mean(bitlen[1:-1])/4.:
            flag = False
            break

    if flag == False: return [next_ptr, 0]
    if bitlen[-1] > 1.5*np.mean(bitlen[1:-1]): ptr -=
int(round(np.mean(bitlen[1:-1]),0))
    return [ptr, int(round(np.mean(bitlen[1:-1],0)))]

def getbyte (samp, i, calib):
```

```
    l_samp = len(samp)
    n_bits = 0
    digits = []
    fin = i+calib*9
    while (i+1 < l_samp and n_bits < 8):
        buf = i
        cnt = 0
        while i+1 < fin and samp[i] == samp[i+1]:
            i+=1
            cnt+=1
        i+=1
        cnt += 1
        #print int(round(cnt/float(calib),0))i
        dis = int(round(cnt/float(calib), 0))
        digits += [samp[buf]]*dis
        n_bits += dis
    bitstr = ''.join(str(i) for i in digits)
    l = len (bitstr)
    litera = int(bitstr[:8], 2)
    return [i - calib*(l-8), litera]

def start (samp, i, calib):
    cnt = 0
    buf = 0x00
    while (buf != 0x53 and cnt < calib*16):
        i_curr = i
        Null, buf = getbyte(samp, i, calib)
        if buf == 0x53:
            i = Null
            break
        else:
            i += 1
        cnt += 1
    if cnt == calib*16: return [0,'CRC_ERROR']
    return [i, chr(buf)]

def generate():
    num_tests = 10
    tests = []
    sig = Signal (mygenerate("Test"*24))
    sig.sample()
```

```
    tests.append(' '.join(str(i) for i in sig.signal) + '\n')

    sig = Signal (mygenerate("There is broken calibration.", cal = 'n'))
    sig.sample()
    tests.append(' '.join(str(i) for i in sig.signal) + '\n')

    sig = Signal (mygenerate("There is broken start bit.", bit = 'n'))
    sig.sample()
    tests.append(' '.join(str(i) for i in sig.signal) + '\n')

    sig = Signal (mygenerate("There is random message.", txt = 'n'))
    sig.sample()
    tests.append(' '.join(str(i) for i in sig.signal) + '\n')

    sig = Signal (mygenerate("There is wrong length.", lng = 'n'))
    sig.sample()
    tests.append(' '.join(str(i) for i in sig.signal) + '\n')

    sig = Signal (mygenerate("And also wrong checksum.", chk = 'n'))
    sig.sample()
    tests.append(' '.join(str(i) for i in sig.signal) + '\n')

    sig = Signal (mygenerate("Yet another correct test. Shall I compare thee to
a summer's day? Thou art more lovely and more temperate"))
    sig.sample()
    tests.append(' '.join(str(i) for i in sig.signal) + '\n')

    sig = Signal (mygenerate("Everything is broken.", chk = 'n', lng = 'n', bit
= 'n', txt = 'n'))
    sig.sample()
    tests.append(' '.join(str(i) for i in sig.signal) + '\n')

    return tests

def check(reply, clue):
    sys.stderr.write(clue)
    sys.stderr.write(reply)
    return reply == clue
```

РЕШЕНИЕ:

Программа на языке C++, решающая данную задачу:

```
1. #include <iostream>
```

```
 2. using namespace std;
 3. const float V = 5.0;
 4. int readBit(float d){
 5.   if(d < V/5.)
 6.       return 0;
 7.   if(d > V*4./5.)
 8.       return 1;
 9.   return -1;
10. }
11. float tmp=0;
12. float safeGetSignal(){
13.   if(signal.get(&tmp))
14.       exit(0);
15.   else
16.       return tmp;
17. }
18. int tickLen=0;
19. long int offset = 0;
20. bool lastSkipBit = 0;
21. void skipBit(){
22.   bool b = (bool) readBit(safeGetSignal());
23.   if(b!=lastSkipBit)
24.       offset = 0;
25.   lastSkipBit = b;
26.     offset++;
27. }
28. bool readBit(){
29.   offset++;
30.   return (bool) readBit(safeGetSignal());
31. }
32. bool readKnownBit(){
33.   while((offset+tickLen/2)%tickLen > 0 )
34.       skipBit();
35.   return readBit();
36. }
37. int readByte(){
38.   char b = 0;
39.   for(int i=0;i<8;i++){
40.       b |= readKnownBit() << 7-i;
41.   }
42.   return b;
```

```
43. }
44. bool checkPattern(int* hl, int* ll){
45.   int highLen = 0;
46.   int lowLen = 0;
47.   while (readBit(safeGetSignal()) == 1)
48.       highLen++;
49.   if (readBit(tmp) == 0) {
50.       while (readBit(safeGetSignal()) == 0)
51.           lowLen++;
52.       if (abs(highLen - lowLen) < min(lowLen, highLen) / 10.f) {
53.           *hl += highLen;
54.           *ll += lowLen;
55.           return true;
56.       }
57.   }
58.   return false;
59. }
60. long int searchStartMark(){
61.   int highLen = 0;
62.   int lowLen = 0;
63.   bool flag;
64.   while(true){
65.       flag = true;
66.       lowLen = 0;
67.       highLen = 0;
68.       while(readBit(safeGetSignal())!=1);
69.       for(int i=0;i<10;i++){
70.           if(!checkPattern(&highLen,&lowLen)) {
71.               flag = false;
72.               break;
73.           }
74.       }
75.       if(flag) {
76.           return highLen + lowLen;
77.         }
78.     }
79. }
80. #include <string.h>
81. #include <cmath>
82. bool fall =0;
83. float f_ = 0;
```

```
int main() {
 signal.get(&f_);
 offset = searchStartMark();
 tickLen = (int) (round(offset / 20.));
 bool lastBit = readKnownBit();
 for(int i=0;i<15;i++){
     bool b = readKnownBit();
     if(b == lastBit)
         break;
     else
         lastBit = b;
 }
 offset -- ;
 int check = readByte();
 if(check!=0x53) {
     exit(0);
 }
 int len = readByte();
 char *content = new char[len];
 int myCRC = len;

 for(int i=0;i<len;i++){
     content[i] = (char) readByte();
     myCRC ^= content[i];
 }
 int CRC = readByte();
 if(CRC != myCRC){
     for(int i=0;i<len;i++){
     }
     cout << "CRC_ERROR" << endl;
 }else{
     for(int i=0;i<len;i++){
         cout << content[i];
     }
     cout  << endl;
 }
 return fall;
}
```

2.3 Дорожная разметка

Ваша задача — написать программу, которая управляет автомобилем, едущим по

дорожной разметке. Разметка моделируется набором строк, передающихся на стандартный ввод программы. Каждая строка является последовательностью символов типа char '0' и '1' (без пробелов между ними и кавычек!), строки разделены символом переноса строки '\n'. Робот обладает набором пяти сенсоров, каждый из которых считывает один из идущих подряд пяти символов. Ниже приведен пример трека. Положение сенсоров обозначено символами `vvvvv`:

```
           vvvvv
000000000000100
000000000001000
000000000001000
000000000000100
000000000001000
000000000010000
000000000001000
000000000001000
000000000000100
000000000000100
```

в данном примере сенсоры должны считать набор символов {0, 0, 1, 0, 0}.

Задача 2.3.1 (1 балл)

В первом задании необходимо написать определение функции `sensor()`. Эта функция в качестве аргументов принимает С-подобную строку `char track[]`, переменную `int car`, указывающее на положение центрального датчика относительно края дороги (нумерация идет от начала строки, с нуля), и массив `int values[5]`, в который в правильном порядке должны быть записаны значения, считанные датчиками с полос дороги. Объявление функции:

```
int sensor(char track[], int car, int values[]);
```

функция должна возвращать значение 1 в случае успешного завершения и -1 в случае ошибки (один из сенсоров за пределами трассы, символ отличается от '0' или '1'). Вы можете воспользоваться следующими константами, которые были определены в закрытой части программы:

```
#define ERROR -1
#define OK     1
```

Напишите определение функции `sensor()`. Вам нужно дополнить уже имеющийся

шаблон кода, нет необходимости в создании функции `main()`, программа проверки сделает это за вас.

Формат входных данных:

первое значение — значение, передаваемое `int car`;

следующая за ним строка — `char  track[]`;

Привет ввода:

```
11
01001100111101010010
```

Формат выходных данных:

строка, состоящая из цифр, разделенных пробелами. Первое значение - число, которое возвращает функция. Остальные значения - показатели датчиков из массива int values[].

Пример вывода:

```
1 1 1 1 0 1
```

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import random as rd
import numpy as np

def strgen(type = 'normal'):
    if type == 'normal':
        width = rd.randint (10, 30)
        track = []
        return str(rd.randint(2, width - 4))+'\n'+ ''.join(str(rd.randint(0,1))
for i in range(width)) + '\n'

    else:
        width = rd.randint (10, 30)
        track = []
        return str(rd.randint(0, width - 1))+'\n'+ ''.join(str(rd.randint(0,9))
for i in range(width)) + '\n'

def generate():
    num_tests = 20
    tests = []
    for test in range(num_tests):
```

```
        text = strgen() + '\n'
        tests.append(text)
    for test in range(num_tests):
        text = strgen('unusual') + '\n'
        tests.append(text)
    return tests

def check(reply, clue):
    r = int(reply.split()[0])
    c = int(clue.split()[0])
    if c == r and r == -1:
        return True
    for i in range(6):
        if clue[i*2] != reply[i*2]:
                    return False

    return True
```

РЕШЕНИЕ:

Программа на языке C++, решающая данную задачу:

```
01. int sensor(char track[], int car, int values[])
02. {
03.       int count;
04.       for (count = 0; track[count] != '\0'; count++);
05.       if (car < 2 || car > count - 4) return ERROR;
06.
07.       for (int i = 0; i < 5; i++)
08.       {
09.             values[i] = (int) (track[car - 2 + i] - '0');
10.             if (values[i] != 0 && values[i] != 1) return ERROR;
11.       }
12.       return OK;
13. }
```

Задача 2.3.2 (1 балл)

Напишите функцию, которая будет удерживать машину на дорожной разметке в виде одной линии. При этом центр машины (центральный датчик из предыдущего задания) должен совпадать с разметкой.

В данном задании необходимо написать определение функции `int sngl_line()`, которая принимает в качестве аргументов набор значений, полученный от сенсоров в виде

массива `int sens[5]`, и аргумент, переданный по указателю — `int *steer`, который является управляющим сигналом для руля. Определение функции:

```
int sngl_line (int sens[], int *steer);
```

Если `int sens[] = {0, 0, 1, 0, 0}`, это означает, что машина находится точно над линией разметки и должна ехать по прямой. В этом случае машина не должна изменять направления. В переменную, переданную по указателю `int *steer`, должен быть записан ноль.

В случае, когда в `int sens[] = {0, 1, 0, 0, 0}`, машина должна повернуть «влево» для этого в `int *steer` необходимо записать -1.

Для поворота «вправо» при `int sens [] = {0, 0, 0, 1,0}` запишите в `int *steer` значение 1.

В перечисленных выше случаях функция `int sngl_line()` должна возвращать код `OK`. Во всех остальных случаях функция `int sngl_line()` должна возвращать код ошибки `ERROR` и не изменять значение `int *steer`.

Пример ввода:

```
7
0000000100000000
0000001000000000
0000000100000000
0000000100000000
0000000100000000
0000000100000000
0000001000000000
0000001000000000
0000001000000000
0000010000000000
0001000000000000
0001000000000000
0010000000000000
0001000000000000
0001000000000000
0010000000000000
0001000000000000
0000100000000000
0001000000000000
```

Пример вывода:

```
0 1 -1 1 1 1 0 1 0 1 0 1 -1 1 0 1 0 1 -1 1 -1 1 -1 1 0 1 -1 1 1 1 0
1 -1 1 1 1 1 1 -1 1
```

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import random as rd

def settest (type = 'normal', width = 15):
    track = ''
    if type == 'normal':
        car = rd.randint(2, width - 3)
        track += "".join(str(car))
        track += '\n'
        line = car
        for i in range(20):
            track_part = [0] * width
            for i in [car]:
                track_part[i] = 1;
            for i in [car]:
                track += ''.join(str(i) for i in track_part)
            track += '\n'

            if car == 2:
                car += rd.randint(0,1)
            elif car == width - 3:
                car -= rd.randint(0,1)
            else:
                car += rd.randint(-1,1)

    elif type == 'outrange':
        car = rd.randint(2, width - 3)
        track += "".join(str(car))
        track += '\n'
        line = car
        for i in range(20):
            track_part = [0] * width
            for i in [car]:
                track_part[i] = 1;
            for i in [car]:
                track += ''.join(str(i) for i in track_part)
```

```
        track += '\n'

        if car == 0:
            car += rd.randint(0,1)
        elif car == width-1:
            car -= rd.randint(0,1)
        else:
            car += rd.randint(-1,1)

elif type == 'blocked':
    car = rd.randint(2, width - 3)
    track += "".join(str(car))
    track += '\n'
    line = car
    for i in range(20):
        track_part = [0] * width
        if rd.random() > 0.9:
            for i in [rd.randint(2, width-3) for i in range(width)]:
                track_part[i] = 1;
        else:
            track_part[car] = 1;
        for i in [line]:
            track += ''.join(str(i) for i in track_part)
        track += '\n'

        if car == 2:
            car += rd.randint(0,1)
        elif car == width-3:
            car -= rd.randint(0,1)
        else:
            car += rd.randint(-1,1)

elif type == 'double_blocked':
    car = rd.randint(2, width - 3)
    track += "".join(str(car))
    track += '\n'
    line = car
    for i in range(20):
        track_part = [0] * width
        if rd.random() > 0.9:
            for i in [rd.randint(3, width-4) for i in range(width)]:
```

```
                    track_part[i] = 1;
            else:
                track_part[car-3] = 1;
                track_part[car+3] = 1;
            for i in [line]:
                track += ''.join(str(i) for i in track_part)
            track += '\n'

            if car == 3:
                car += rd.randint(0,1)
            elif car == width-4:
                car -= rd.randint(0,1)
            else:
                car += rd.randint(-1,1)

    elif type == 'double':
        car = rd.randint(3, width - 4)
        track += "".join(str(car))
        track += '\n'
        line = car
        for i in range(20):
            track_part = [0] * width
            for i in [line]:
                track_part[car-3] = 1;
                track_part[car+3] = 1;
            for i in [line]:
                track += ''.join(str(i) for i in track_part)
            track += '\n'

            if car == 3:
                car += rd.randint(0,1)
            elif car == width - 4:
                car -= rd.randint(0,1)
            else:
                car += rd.randint(-1,1)

    return track

def generate():
    num_tests = 10
    tests = []
```

```
    for test in range(3):
        tests.append(settest())
        tests.append(settest(type = 'outrange'))
        tests.append(settest(type = 'blocked'))
    return tests

def check(reply, clue):
    r = reply.split()
    c = clue.split()

    if c[-1] == r[-1] == '2':
        return True

    for idx, i in enumerate(r):
        if int(i) != int(c[idx]):
            return False

    return True
```

РЕШЕНИЕ:

Программа на языке C++, решающая данную задачу:

```
01. int sngl_line (int sens[], int *steer)
02. {
02.    int sum = 0;
03.    for (int i = 0; i < 5; i++) sum += sens[i];
04.    if (sum > 1) return ERROR;
05.    if (sens[2] == 1)
06.    {
07.        *steer = 0;
08.        return OK;
09.    }
10.    if (sens[1] == 1)
11.    {
12.        *steer = -1;
13.        return OK;
14.    }
15.    if (sens[3] == 1)
16.    {
17.        *steer = 1;
18.        return OK;
19.    }
20.    return ERROR;
```

```
21. }
```

Задача 2.3.3 (1 балл)

Напишите функцию, которая будет удерживать машину на дорожной разметке между двух линий, между которыми идет расстояние в пять «полей» (один сенсор считывает участок шириной в одно «поле»). Функция очень похожа на функцию из предыдущего примера.

В данном задании необходимо написать определение функции `int dbl_line()`, которая принимает в качестве аргументов набор значений, полученный от сенсоров в виде массива `int sens[5]`, и аргумент, переданный по указателю — `int *steer`, который является управляющим сигналом для руля. Определение функции:

```
int dbl_line (int sens[], int *steer);
```

Если `int sens[] = {0, 0, 0, 0, 0}`, это означает, что машина находится меджду линиями разметки и должна ехать по прямой. В этом случае машина не нужно изменять направление. В переменную, переданную по указателю `int *steer`, должен быть записан ноль.

В случае, когда в `int sens[] = {0, 0, 0, 0, 1}`, машина должна повернуть «влево» для этого в `int *steer` необходимо записать -1.

Для поворота «вправо», `int sens[] = {1, 0, 0, 0, 0}` запишите в `int *steer` значение 1.

В перечисленных выше случаях функция `int dbl_line()` должна возвращать код `OK`. Во всех остальных случаях функция `int dbl_line(2)` должна возвращать код ошибки `ERROR` и не изменять значение `int *steer`.

Пример ввода:

```
7
000010000010000
000100000100000
000100000100000
000100000100000
000100000100000
001000001000000
001000001000000
000100000100000
001000001000000
000100000100000
```

```
000100000100000
000100000100000
000100000100000
000010000010000
000010000010000
000100000100000
001000001000000
001000001000000
001000001000000
010000010000000
```

Пример вывода:

```
0 1 -1 1 0 1 0 1 0 1 -1 1 0 1 1 1 -1 1 1 1 0 1 0 1 0 1 1 1 0 1 -1 1
-1 1 0 1 0 1 -1 1
```

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import random as rd
def settest (type = 'normal', width = 15):
    track = ''
    if type == 'normal':
        car = rd.randint(2, width - 3)
        track += "".join(str(car))
        track += '\n'
        line = car
        for i in range(20):
            track_part = [0] * width
            for i in [car]:
                track_part[i] = 1;
            for i in [car]:
                track += ''.join(str(i) for i in track_part)
            track += '\n'

            if car == 2:
                car += rd.randint(0,1)
            elif car == width - 3:
                car -= rd.randint(0,1)
            else:
                car += rd.randint(-1,1)

    elif type == 'outrange':
```

```
    car = rd.randint(2, width - 3)
    track += "".join(str(car))
    track += '\n'
    line = car
    for i in range(20):
        track_part = [0] * width
        for i in [car]:
            track_part[i] = 1;
        for i in [car]:
            track += ''.join(str(i) for i in track_part)
        track += '\n'

        if car == 0:
            car += rd.randint(0,1)
        elif car == width-1:
            car -= rd.randint(0,1)
        else:
            car += rd.randint(-1,1)

elif type == 'blocked':
    car = rd.randint(2, width - 3)
    track += "".join(str(car))
    track += '\n'
    line = car
    for i in range(20):
        track_part = [0] * width
        if rd.random() > 0.9:
            for i in [rd.randint(2, width-3) for i in range(width)]:
                track_part[i] = 1;
        else:
            track_part[car] = 1;
        for i in [line]:
            track += ''.join(str(i) for i in track_part)
        track += '\n'

        if car == 2:
            car += rd.randint(0,1)
        elif car == width-3:
            car -= rd.randint(0,1)
        else:
            car += rd.randint(-1,1)
```

```
elif type == 'double_blocked':
    car = rd.randint(2, width - 3)
    track += "".join(str(car))
    track += '\n'
    line = car
    for i in range(20):
        track_part = [0] * width
        if rd.random() > 0.9:
            for i in [rd.randint(3, width-4) for i in range(width)]:
                track_part[i] = 1;
        else:
            track_part[car-3] = 1;
            track_part[car+3] = 1;
        for i in [line]:
            track += ''.join(str(i) for i in track_part)
        track += '\n'

        if car == 3:
            car += rd.randint(0,1)
        elif car == width-4:
            car -= rd.randint(0,1)
        else:
            car += rd.randint(-1,1)

elif type == 'double':
    car = rd.randint(3, width - 4)
    track += "".join(str(car))
    track += '\n'
    line = car
    for i in range(20):
        track_part = [0] * width
        for i in [line]:
            track_part[car-3] = 1;
            track_part[car+3] = 1;
        for i in [line]:
            track += ''.join(str(i) for i in track_part)
        track += '\n'

        if car == 3:
            car += rd.randint(0,1)
```

```
            elif car == width - 4:
                car -= rd.randint(0,1)
            else:
                car += rd.randint(-1,1)

    return track

def generate():
    num_tests = 10
    tests = []
    for test in range(3):
        #tests.append(settest())
        tests.append(settest(type = 'double'))
        tests.append(settest(type = 'double_blocked'))
    return tests

def check(reply, clue):
    r = reply.split()
    c = clue.split()

    if c[-1] == r[-1] == '2':
        return True

    for idx, i in enumerate(r):
        if int(i) != int(c[idx]):
            return False

    return True
```

РЕШЕНИЕ:

Программа на языке C++, решающая данную задачу:

```
01. int dbl_line (int sens[], int *steer)
02. {
03.   int sum = 0;
04.   for (int i = 0; i < 5; i++) sum += sens[i];
05.   if (sum > 1) return ERROR;
06.   if (sum == 0)
07.   {
08.     *steer = 0;
09.     return OK;
10.   }
11.   if (sens[4] == 1)
```

```
12.   {
13.     *steer = -1;
14.     return OK;
15.   }
16.   if (sens[0] == 1)
17.   {
18.     *steer = 1;
19.     return OK;
20.   }
21.   return ERROR;
22. }
```

Задача 2.3.4 (1 балл)

Напишите программу, которая получает на стандартный ввод начальное положение машины (в виде символов цифр типа `char` от '0' до '9', число может быть одно- или двузначным) и данные в виде набора символов '0' и '1', описывающие трек, на который нанесена разметка в виде одной или двух линий. В стандартный вывод программа должна выводить набор команд для системы управления: 1 для поворота «вправо», -1 для поворота влево (подобно предыдущим шагам) и 0 для движения без изменения направления, разделенных символом табуляции (требование платформы проверки заданий). Для первой строки ваша программа должна вывести в стандартный вывод '0'. Вы можете реализовать вывод в стиле C++ или C, например,

```
printf("%d\t", steer);
```

Вы можете воспользоваться возможностями стандартных библиотек C и C++, подключив необходимые заголовки, а так же воспользоваться функциями `int sensors()`, `int sngl_line()`, `int dbl_line()` из предыдущих шагов. Эти функции уже реализованы в программе, не переопределяйте их.

Пример ввода №1:

```
4
000010000000000
000001000000000
000010000000000
000010000000000
000001000000000
000000100000000
000001000000000
000000100000000
```

```
000001000000000
000000100000000
```

Пример вывода №1:

```
0    1     -1    0     1     1     -1    1     -1    1
```

Пример ввода №2:

```
10
0000000100000010
0000000010000001
0000000100000010
0000000100000010
0000000010000001
0000000100000010
0000000010000001
0000000010000001
0000000100000010
0000000010000001
```

Пример вывода №2:

```
0    1     -1    0     1     -1    1     0     -1    1
```

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import sys
import random as rd
def settest (type = 'normal', width = 15):
    track = ''
    if type == 'normal':
        car = rd.randint(2, width - 3)
        track += "".join(str(car))
        track += '\n'
        line = car
        for i in range(10):
            track_part = [0] * width
            for i in [car]:
                track_part[i] = 1;
            for i in [car]:
                track += ''.join(str(i) for i in track_part)
            track += '\n'
```

```
        if car == 2:
            car += rd.randint(0,1)
        elif car == width - 3:
            car -= rd.randint(0,1)
        else:
            car += rd.randint(-1,1)

elif type == 'outrange':
    car = rd.randint(2, width - 3)
    track += "".join(str(car))
    track += '\n'
    line = car
    for i in range(20):
        track_part = [0] * width
        for i in [car]:
            track_part[i] = 1;
        for i in [car]:
            track += ''.join(str(i) for i in track_part)
        track += '\n'

        if car == 0:
            car += rd.randint(0,1)
        elif car == width-1:
            car -= rd.randint(0,1)
        else:
            car += rd.randint(-1,1)

elif type == 'blocked':
    car = rd.randint(2, width - 3)
    track += "".join(str(car))
    track += '\n'
    line = car
    for i in range(20):
        track_part = [0] * width
        if rd.random() > 0.9:
            for i in [rd.randint(2, width-3) for i in range(width)]:
                track_part[i] = 1;
        else:
            track_part[car] = 1;
        for i in [line]:
```

```
            track += ''.join(str(i) for i in track_part)
        track += '\n'

        if car == 2:
            car += rd.randint(0,1)
        elif car == width-3:
            car -= rd.randint(0,1)
        else:
            car += rd.randint(-1,1)

elif type == 'double_blocked':
    car = rd.randint(3, width - 4)
    track += "".join(str(car))
    track += '\n'
    line = car
    for i in range(10):
        track_part = [0] * width
        if rd.random() > 0.9:
            for i in [rd.randint(3, width-4) for i in range(width)]:
                track_part[i] = 1;
        else:
            track_part[car-3] = 1;
            track_part[car+3] = 1;
        for i in [line]:
            track += ''.join(str(i) for i in track_part)
        track += '\n'

        if car == 3:
            car += rd.randint(0,1)
        elif car == width-4:
            car -= rd.randint(0,1)
        else:
            car += rd.randint(-1,1)

elif type == 'double':
    car = rd.randint(3, width - 4)
    track += "".join(str(car))
    track += '\n'
    line = car
    for i in range(10):
        track_part = [0] * width
```

```
            for i in [line]:
                track_part[car-3] = 1;
                track_part[car+3] = 1;
            for i in [line]:
                track += ''.join(str(i) for i in track_part)
            track += '\n'

            if car == 3:
                car += rd.randint(0,1)
            elif car == width - 4:
                car -= rd.randint(0,1)
            else:
                car += rd.randint(-1,1)

    return track

def generate():
    num_tests = 10
    tests = []
    for test in range(10):
        if test % 2 == 0:
            tests.append(settest('normal'))
        else:
            tests.append(settest('double'))

    return tests

def check(reply, clue):
    return reply == clue
```

РЕШЕНИЕ:

Программа на языке C++, решающая данную задачу:

```
1.   #include <string.h>
2.   #include <iostream>
3.   #define ERROR -1
4.   #define OF 1
5.   int getsens (char track[], int car[], int sens[]);
6.   int sngl_line (int sens[], int *steer);
7.   int dbl_line (int sens[], int *steer);
8.   using namespace std;
9.   int main() {
10.   int pos;
```

```
11.   cin >> pos;
12.   cerr << pos << endl;
13.   string line;
14.   cin >> line;
15.   int j = 0;
16.   for (int i=0; i<line.size(); i++) {
17.         if (like[i] == '1')
18.               j++;
19.   }
20.   int (*func)(int *, int *) = 0;
21.   switch(j){
22.         case 1:
23.               func = sngl_line;
24.               break;
25.         case 2:
26.               func = dbl_line;
27.               break;
28.   }
29.   if (func == 0) {
30.         cerr << "Error\n";
31.         return 1;
32.   }
33.   int sens[5] = {};
34.   int d = 0;
35.   do {
36.         cerr << line << endl;
37.         cerr << getsens((char *) line.c_str(), pos, sens) << endl;
38.         func(sens,&d);
39.         count << d << "\t";
40.         pos += d;
41.   } while (cin >> line);
42.   cout << endl;
43.   return 0;
44. }
```

2.4 Беспилотные летательные аппараты

Задача 2.4.1 (тренировочная, не приносит баллов)

Рассчитайте, сможет ли взлететь следующий квадрокоптер. Считайте все двигатели одинаковыми, трение воздуха учитывать не нужно. Потерями тока, проходящего через

регуляторы мощности можно пренебречь.

- **Рама.** Масса (вместе со всеми проводами и пропеллерами): 180 грамм
- **Электродвигатель.** Характеристики двигателя приведены в таблице далее, используемые пропеллеры 5030. Масса: 12 грамм

NO LOAD		ON LOAD				LOAD TYPE
VOLTAGE	CURRENT	SPEED	CURRENT	Pull	Power	Battery/prop
V	A	rpm	A	g	W	
7.4	0.3	17100	0.6	40	4.4	LiPox2/5030
			1.6	80	11.8	
			1.9	100	14.1	
			2.1	110	15.5	
			2.5	130	18.5	
11.1	0.3	22800	0.9	70	10.0	LiPox3/5030
			2.0	100	22.2	
			2.1	110	23.3	
			2.3	150	25.5	
			4.3	220	47.7	
7.4	0.3	17100	0.7	45	5.2	LiPox2/6020
			1.6	80	11.8	
			2.3	100	17.0	
			2.8	110	20.7	
			3.6	140	26.6	
11.1	0.3	22800	1.1	70	12.2	LiPox3/6020
			2.6	100	28,9	
			2.8	110	31.1	
			3	150	33.3	
			5.7	220	63.3	

- **Полётный контроллер**. Масса: 15 грамм. Потребляемый ток: 0,2 А
- **Регуляторы мощности.** Масса: 8 грамм
- **Аккумулятор** LiPo. Масса: 125 грамм. Пиковый ток разряда: 5С. Емкость: 1500 мАч. Напряжение: 3S (11,1 В)

Взлетит ли коптер? Да/нет.

РЕШЕНИЕ:

1. Вычислим общую массу квадрокоптера, она складывается из массы рамы, 4-х электродвигателей, полётного контроллера, 4-х регуляторов мощности и аккумулятора. $180 + 4 \cdot 12 + 15 + 8 \cdot 4 + 125 = 400$ г.
2. Рассчитаем максимальную тягу 4-х электродвигателей: $4 \cdot 220$ г $= 880$ г. Мощности

моторов достаточно для того, чтобы квадрокоптер поднялся в воздух.

3. Вычислим потребляемый для взлета ток. У нас есть 2 потребителя тока: электродвигатели, полётный контроллер, их минимальный общий ток, необходимый для взлёта равен: $I_{общ} = 4 \cdot I_1 + I_2 = 4 \cdot 2A + 0,2\ A = 8,2\ A$
4. Рассчитаем токоотдачу аккумулятора (С), необходимую для висения. $C = I_{общ} / E_{аккумулятора} = 8,2\ A / 1,5\ A \cdot ч = 5,5\ C$. Имеющегося у нас аккумулятора с 5С недостаточно. Квадрокоптер не сможет взлететь.

Задача 2.4.2 (1 балл)

Для квадрокоптера из задания 4.1 рассчитайте максимальную длительность висения в безветренную погоду на аккумуляторе емкостью 2500 мАч, весом 165 г, с максимальным током разряда 30С. Потери учитывать не нужно.

Ответ дайте в минутах.

РЕШЕНИЕ:

1. Вычислим общую массу квадрокоптера, она складывается из массы рамы, 4-х электродвигателей, полётного контроллера, 4-х регуляторов мощности и аккумулятора. $180 + 4 \cdot 12 + 15 + 8 \cdot 4 + 165 = 440$ г.
2. Чтобы квадрокоптер находился в полёте как можно дольше, нужно использовать минимально возможную тягу, т.к. тогда ток будет минимальным, а значит заряда аккумулятора хватит на большее количество времени. Наш квадрокоптер весит 440 грамм, соответственно чтобы он висел в воздухе каждый из электромоторов должен создавать тягу в 110 грамм. По таблице электромотора выбираем соответствующее значение тока 2,1 А.
3. Посчитаем минимальное общее потребление: $I_{общ} = 4 \cdot I_1 + I_2 = 4 \cdot 2.1A + 0,2\ A = 8,6\ A$
4. Исходя из этого посчитаем максимальную продолжительность полёта: $t = A \cdot ч / 8,6A = 0,291$ ч. Переведём ответ в минуты. $t = 0,291 \cdot 60 = 17,46$ минут.
5. Правильный ответ лежит в интервале от 17,0 до 17,46 т. к. возможен вариант расчета с большей мощностью в начале, чтобы квадрокоптер взлетел.

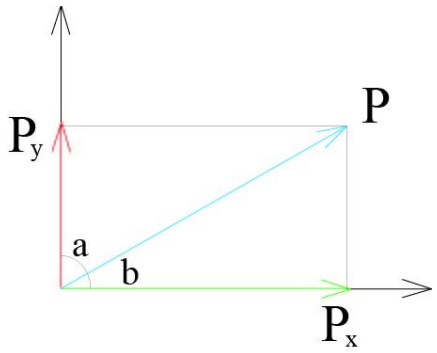
Задача 2.4.3 (2 балла)

Для квадрокоптера из задания 3 оцените максимальную горизонтальную скорость полёта на постоянной высоте. Используемые пропеллеры 5030 (длинна пропеллера 5 дюймов, шаг 3 дюйма), максимальная частота вращения электродвигателя 22800 об/мин. Сопротивлением воздуха пренебречь.

Ответ дайте в м/с.

РЕШЕНИЕ:

1. Горизонтальная скорость коптера может быть оценена сверху горизонтальной составляющей скорости воздуха, который он выталкивает своими винтами. Скорость воздуха оценивается сверху через расстояние, которое винт проходит за один оборот в вязкой среде (шаг), делённое на время одного оборота винта, которое мы можем вычислить из его частоты вращения. *Скорость*=шаг/(1/частота) = 3 дюйма / (1/22800 об/мин). Приведем все числа к системе СИ. 3 дюйма=3 · 2,54 см = 7,62 см = 0,0762 м. 22800 об/мин = 380 об/сек, тогда *Скорость*=0,0762 м/(1с/380) = 28,96 м/с.
2. Для уравновешивания силы тяжести квадрокоптера нужна тяга 440 грамм, а общая тяга 880 грамм. Соответственно проекция общей тяги на ось Y будет 440 грамм. Вычислим угол a. Проекция вектора Py = P · cos a, отсюда cos a = Py/P=440/880=1/2. Следовательно, угол a равен 60^o.
3. Соответственно угол b равен 90^o - 60^o = 30^o, посчитаем проекцию на ось x.
4. Скорость воздуха направлена под углом 30^o к оси x, следовательно проекция на ось x будет равна vx = v cos30^o = 28,96 · (√3/2) = 25,08 м/с



Задача 2.4.4 (2 балла)

На вашем квадрокоптере стоит сонар, который направлен вперёд и получает данные о расстоянии до препятствий, которые видит, но показания не идеальны — они имеют определенную погрешность и порой случаются выбросы. Частота обновления 10 Гц, программа обрабатывает значения, полученные каждую секунду.

Пример показаний (в сантиметрах):

250; 232; 218; 211; 10; 210; 212; 212; 220; 218.

Значение 10 — выброс, который может привести к тому, что квадрокоптер попытается отлететь от несуществующего препятствия. Напишите определение функции на языке C++, обрабатывающей массив входных данных описаным ниже образом.

Из массива удаляются выбросы. В данном случае под выбросами подразумеваются значения, лежащие вне интервала

[p25 − 1,5 · (p75 − p25), p75 + 1.5 · (p75 − p25)],

где p25, p75 - нижний и верхний квартили выборки. По отфильтрованным значениям должны быть вычислены среднее арифметическое и среднеквадратичное отклонение. Данные величины должны быть записаны в переменные типа `float`, переданные функции по указателю.

Объявление функции, которой необходимо дать определение, выглядит так:

`void distance (float dists[], float *mean, float *mse ),`

где `float dists[]` - массив значений, преданный от сонара, `float *mean` - указатель на переменную, куда будет записано среднего арифметического, и `float *mse` - указатель на переменную, куда будет записано среднеквадратичное отклонение.

При проверке решения допускается отклонение в 0,1% от абсолютного значения для среднего арифметического и среднеквадратичного отклонения.

Комментарий: Вы можете подключать стандартные бибилиотеки языка C/C++, добавляя их при помощи макроса #include.

Пример ввода:

```
495.151989742  490.515255103  490.97001079  489.813852978
487.181192706  477.62829801  488.322608191  495.251663444
477.757448578 474.005367207
```

Пример вывода:

```
 486.659768675 7.16463010295
```

СПОСОБ ОЦЕНКИ РАБОТЫ:

Для генерации тестов и проверки результата используется следующий код на языке Python:

```
import math
import sys
import numpy as np
import random as rd

def isclose(a, b, rel_tol=1e-09, abs_tol=0.0):
    return abs(a-b) <= max(rel_tol * max(abs(a), abs(b)), abs_tol)

class Sample():
    def __init__ (self, speed = rd.randint(0, 10), rng = 10, start =
```

```
rd.randint(250, 500)):
        speed *= rd.randint(-1, 1)
        self.speed = speed
        self.rng = rng
        self.start = start

    def skew (self):
        if self.speed * self.rng > self.start:
            rng = int (self.start / self.speed)
        else:
            rng = self.rng
        self.track = [self.start - i*self.speed for i in range(rng)]

    def shift(self):
        try:
            mn = float(sum(self.track) / len(self.track))
            for idx, i in enumerate (self.track):
                self.track[idx] += (rd.random() - 0.5) * mn/20.
        except ValueError:#NameError:
      #      print ("Track is not exist")
            pass
    def blowout(self):
        try:
            mn = float(sum(self.track) / len(self.track))
            for idx, i in enumerate (self.track):
                self.track[idx] += (np.random.normal(0, 2*mn, 1)[0] *
((rd.random() + 0.05) // 1))

                self.track[idx] = abs(self.track[idx])
        except NameError:
            print ("Track is not exist")

def generate():
    num_tests = 20
    tests = []
    for test in range(num_tests):
        S = Sample()
        S.skew()
        S.shift()
        S.blowout()
        text = ' '.join(str(i) for i in S.track) + '\n'
```

```
        tests.append(text)

    return tests

def solve(dataset):
    sample = [float(i) for i in dataset.split()]



    p25 = np.percentile(sample, 25)
    p75 = np.percentile(sample, 75)

    buf = []
    for i in sample:
        if (p25 - 1.5*(p75-p25)) < i < (p75 + 1.5*(p75-p25)):
            buf.append(i)

    mean = np.mean(buf)
    mse = np.std(buf)

    return ' '.join(str(i) for i in (mean, mse))

def check(reply, clue):
    rep = [float(i) for i in reply.split()]
    clu = [float(i) for i in clue.split()]

    if isclose(rep[1], clu[1], rel_tol = 0.0001) and isclose(rep[0], clu[0],
rel_tol = 0.0001):
        return True



    return False
```

РЕШЕНИЕ:

Программа на языке C++, решающая данную задачу:

```
1. include <iostream>
2. #include <alorithm>
3. #include <vector>
4. using namespace std;
5. void distance (float dists[], float *mean, float *mrse)
6. {
7.     sort(dists, dists+10);
8.     float p25 = dists[2];
```

```
 9.     float p75 = dists[7];
10.     float m_low = p25 - 1.5f*(p75-p25);
11.     float m_hig = p75 - 1.5f*(p75-p25);
12.     vector<float> data;
13.     float sum = 0;
14.     for (int j = 0; j < 10; ++j) {
15.           if (dists[j] >= m_low && dists[j] <= m_hig) {
16.                 data.push_back(dists[j]);
17.                 sum += dists[j];
18.           }
19.     }
20.     float my_mean = sum/(float)data.size();
21.     *mean = my_mean;
22.     float sqr_err_sum = 0;
23.     for (vector<float>::iterator iter = data.begin(); iter !=
data.end(); ++iter) {
24.           sqr_err_sum += pow(*iter - my_mean,2);
25.     }
26.     *mrse = pow(sqr_err_sum/(float)data.size(),1.0f/2.0f);
27. }
```

2.5. Критерии определения победителей и призеров

Количество баллов, набранных при решение всех задач суммируется. Призерам второго отборочного этапа было необходимо набрать 4 балла из 25 возможных. Победители второго отборочного этапа должны были набрать 20 баллов из 25.