```python
__author__ = 'Eugene Kolivoshko'

class Angle(object): #Клас кутів
    def __init__(self, a): # Функція ініціалізації об’єкта
        self.__list = [[1, 0], [0, 1], [-1, 0], [0, -1]] # [cos(a), sin(a)]
        self.__index = 0
        self.angle = [1, 0]
        for i in [0, 1, 2, 3]:
            if a == self.__list[i]:
                self.__index = i
                self.angle = self.__list[i]

    def prev(self): #поворот на G-
        if self.__index == 0:
            self.__index = 3
        else:
            self.__index -= 1

        self.angle = self.__list[self.__index]

    def next(self): #поворот на G+
        if self.__index == 3:
            self.__index = 0
        else:
            self.__index += 1

        self.angle = self.__list[self.__index]


class Section(object): #Клас ланок
    def __init__(self, n, p, a): # Функція ініціалізації об’єкта
        self.number = n # Номер ланки
        self.point = p  # Координата
        self.angle = a  # Кут у вигляді [cos(a), sin(a)]

    def __str__(self):
        return "%d %s %d" % (self.number, self.point, self.angle)


class Chain(object): #Клас ланцюжків
    def __init__(self, maxLen): # Функція ініціалізації об’єкта
        self.maxLen = maxLen
        self.chain = []
        self.__count = 1
        self.chain.append(Section(self.__count, [0, 0], [0, 1]))
        self._endPoint = [0, 1]
        self.__thisSection = self.chain[-1]
        self.allow = True
        self.__thisPoint = [0, 1]
        self.length = 1.0

    def genChain(self, style, list=None): # Функція побудови ланцюжка
        if style == 'random': import random
        i = 0
        if style == 'random': rotate = random.randrange(-1, 2) # Випадкового
        if style == 'range': rotate = 0 # Прямого
        if style == 'list': rotate = list[i] # За заданим листом ротацій

        while i < self.maxLen - 1: # Власне побудова
            self.__next(rotate) # Виклик функції побудови ланки

            if self.allow == False:
                break
            i += 1

    def __next(self, rotate): # Функція побудови ланки
        self.__thisSection = self.chain[-1]
        angle = Angle(self.__thisSection.angle)
        nextSection = Section(self.__count, [0, 0], rotate)

        if rotate == -1:
            angle.prev()

        if rotate == 1:
            angle.next()
```

```
76
77              nextSection.point[0] = self.__thisSection.point[0] + self.__thisSection.angle[0]
78              nextSection.point[1] = self.__thisSection.point[1] + self.__thisSection.angle[1]
79              nextSection.angle = angle.angle
80
81              if self._test(nextSection): # Виклик перевірки на перетинність
82                  self.__count += 1
83                  nextSection.number = self.__count
84                  self.chain.append(nextSection) # Добавлення ланки
85
86      def _test(self, section): # Функція перевірки на перетинність
87          self._getEndPoint(section)
88          i = 0
89          while i < self.__count:
90              if self.chain[i].point == section.point or self.chain[i].point == self._endPoint:
91                  self.allow = False
92                  return False
93                  break
94              i += 1
95          return True
96
97      def _getEndPoint(self, section):
98          self._endPoint[0] = section.point[0] + section.angle[0]
99          self._endPoint[1] = section.point[1] + section.angle[1]
100
101     def getLength(self):
102         import numpy as np
103         self.length = np.sqrt(self.endPoint[0] ** 2 + self.endPoint[1] ** 2)
104
105
106 class ListConfigurations(object): # Клас листа конформацій
107     def __init__(self, length): # Функція ініціалізації об’єкта
108         self.maxChainLength = length
109         self.gapConfigurations = 0
110         self.allowConfigurations = 0
111         self.lengthArr
112         self._genConfigurations()
113
114     def _genConfigurations(self): # Функція побудови і сортування конформацій
115         import itertools
116
117         #Отримати лист всіх можливих ротацій
118         for list in itertools.product(range(-1, 2), repeat=(self.maxChainLength - 1)):
119             chain = Chain(self.maxChainLength) # створення ланцюжка
120             chain.genChain('list', list)       # побудова за листом ротацій ланцюжка
121             chain.getLength()                  # розрахунок довжини
122
123             if chain.allow:
124                 self.allowConfigurations += 1
125                 self.length += chain.length
126             else:
127                 self.gapConfigurations += 1
128
129             #виводимо отримані результати
130             print("Average length: %f" % (float(self.length)/len(self.allowConfigurations)))
131
132 obj2 = ListConfigurations(11)
133 #виводимо отримані результати
134 print(obj2.allowConfigurations, obj2.gapConfigurations)
```