

Task1

Круговой массив - массив из элементов, в котором по достижению конца массива следующим элементом будет снова первый. Массив задается числом n , то есть представляет собой числа от 1 до n .

Пример кругового массива для $n=3$:

1 ⁰	2 ¹	3 ²	1 ³	2 ⁴	3 ⁵	1 ⁶
----------------	----------------	----------------	----------------	----------------	----------------	----------------

Напишите программу, которая выводит путь, по которому, двигаясь интервалом длины m по заданному массиву, концом будет являться первый элемент.

Началом одного интервала является конец предыдущего. Путь - массив из начальных элементов полученных интервалов.

Пример 1:

$n = 4$, $m = 3$

Решение:

Круговой массив: 1234. При длине обхода 3 получаем интервалы: 123, 341. Полученный путь: 13.

Пример 2:

$n = 5$, $m = 4$

Решение:

Круговой массив: 123456. При длине обхода 4 получаем интервалы: 1234, 4512, 2345, 5123, 3451. Полученный путь: 14253.

Параметры передаются в качестве аргументов командной строки (типа `java -jar task1.jar 5 4` или `task1.py 5 4`).

Например, для последнего примера на вход подаются аргументы: 5 4

Ожидаемый вывод в консоль: 14253

Task2

Напишите программу, которая рассчитывает положение точки относительно окружности.
Координаты центра окружности и его радиус считываются из файла1.

Пример:

```
1 1
5
```

Координаты точек считываются из файла2.

Пример:

```
0 0
1 6
6 6
```

Вывод: 1 0 2

Файлы передаются программе в качестве аргументов (типа `java -jar task2.jar test1.txt test2.txt` или `task2.py test1.txt test2.txt`). Файл с координатами и радиусом окружности - 1 аргумент, файл с координатами точек - 2 аргумент.

Координаты в диапазоне float. Количество точек от 1 до 100. Вывод каждого положения точки заканчивается символом новой строки.

Соответствия ответов:

0 - точка лежит на окружности

1 - точка внутри

2 - точка снаружи

Task3

На вход в качестве аргументов программы поступают два файла: tests.json и values.json (в приложении к заданию находятся примеры этих файлов)

values.json содержит результаты прохождения тестов с уникальными id.

tests.json содержит структуру для построения отчёта на основе прошедших тестов (вложенность может быть большей, чем в примере)

Напишите программу, которая формирует файл report.json с заполненными полями value для структуры tests.json на основании values.json.

Пример:

Часть структуры tests.json:

```
{
  "id": 122, "title": "Security test", "value": "", "values": [
    {
      "id": 5321, "title": "Confidentiality", "value": ""
    },
    {
      "id": 5322, "title": "Integrity", "value": ""
    }
  ]
}
```

После заполнения в соответствии с values.json:

```
{
  "values": [
    {
      "id": 122, "value": "failed"
    },
    {
      "id": 5321, "value": "passed"
    },
    {
      "id": 5322, "value": "failed"
    }
  ]
}
```

Будет иметь следующий вид в файле report.json:

```
{
  "id": 122, "title": "Security test", "value": "failed", "values": [
    {
      "id": 5321, "title": "Confidentiality", "value": "passed"
    },
    {
      "id": 5322, "title": "Integrity", "value": "failed"
    }
  ]
}
```

Task4

Дан массив целых чисел `nums`. Напишите программу, выводящую минимальное количество ходов, требуемых для приведения всех элементов к одному числу. За один ход можно уменьшить или увеличить число массива на 1.

Пример:

`nums = [1, 2, 3]`

Решение: `[1, 2, 3] => [2, 2, 3] => [2, 2, 2]`

Минимальное количество ходов: 2

Элементы массива читаются из файла, переданного в качестве аргумента командной строки.

Пример:

На вход подаётся файл с содержимым:

```
1
10
2
9
```

Вывод в консоль: 16