Алгоритмы и структуры данных – Практическая работа №4
[Задачи 45-49]

45. Напишите программу конвертации количества лет из Арабских чисел в Римские. Количество лет в диапазоне 1 < n < 10000.

46. Напишите программу конвертации количества лет из Римских чисел в Арабские. Количество лет в диапазоне 1 < n < 10000.

---

47. На вход подается число N - количество чисел для случайной генерации. Полученный массив чисел необходимо отсортировать в формате Змейки.

Пр. Исходный массив
```
array = [[1,2,3],
[4,5,6],
[7,8,9]]
snail(array) #=> [1,2,3,6,9,8,7,4,5]
array = [[1,2,3],
[8,9,4],
[7,6,5]]
snail(array) #=> [1,2,3,4,5,6,7,8,9]
```

48. u(0) = 1. На вход подается x. y = 2 _ x + 1, z = 3 _ x + 1. Следовательно u = [1, 3, 4, 7, 9, 10, 13, 15, 19, 21, 22, 27, ...]. Напишите программу которая создаст ряд чисел u без дубликатов.

49. Задания 19 - Часть 2. Задана система уравнений:

```java
1) fib(0) = 1
2) fib(1) = 1
3) fib(n + 2) = fib(n) + fib(n + 1), if n + 2 > 1
```

Итак, первый шаг — попытаться найти определение хвостовой рекурсии. Для этого мы пытаемся записать обе части уравнения 3) в одной и той же форме. В настоящее время левая часть уравнения содержит один член, тогда как правая часть представляет собой сумму двух членов. Первая

попытка состоит в том, чтобы добавить `fib(n + 1)` к обеим частям уравнения: 3) -> `fib(n + 1) + fib(n + 2) = fib(n) + 2 * fib(n + 1)`
Две части уравнения выглядят гораздо более похожими, но все же есть существенное различие, которое представляет собой коэффициент при втором члене каждой части. В левой части уравнения это 1, а в правой — 2. Чтобы исправить это, мы можем ввести переменную b: 3) -> `fib(n + 1) + b * fib(n + 2) = b * fib(n) + (b + 1) * fib(n + 1)`
Заметим, что коэффициенты первого слагаемого не совпадают (с 1 которая слева и b - которая справа), поэтому введем переменную a: 3) -> `a * fib(n + 1) + b * fib(n + 2) = b * fib(n) + (a + b) * fib(n + 1)`
Теперь у нас сформированы две одинаковые части которые можно привести к уравнению F: `F(a,b,n) = a * fib(n) + b * fib(n + 1)`
Соответственно уравнение №3 теперь имеет вид: `F(a, b, n + 1) = F(b, a + b, n)`
Теперь возможно сформировать, по определению F и fib: 4) -> `F(a, b, 0) = a * fib(0) + b * fib(1) = a + b`
Соответственно 5) -> `fib(n) = F(1, 0, n)`
Для простоты понимания также опишу функцию на языке Python:

```python
def fib(n):

   def F(a, b, n):
      if n == 0: return a + b    # уравнение 4
      return F(b, a + b, n - 1)  # уравнение 3

   return F(1, 0, n)            # уравнение 5
```

Поскольку рекурсия больших чисел займет слишком много времени, переформируем функцию fib из рекурсии в цикл:

```python
def fib(n):
   a, b = 1, 0                  # уравнение 5
   while n > 0:
      a, b, n = b, a + b, n - 1  # уравнение 3
   return a + b .               # уравнение 4
```

Функция готова к применению. Теперь необходимо реализовать такой же метод решения только для системы уравнений:

1. $fusc(0) = 0$
2. $fusc(1) = 1$
3. $fusc(2n) = fusc(n)$
4. $fusc(2n + 1) = fusc(n) + fusc(n + 1)$