Python 4 семестр

СПРАВКА

ЗАДАНИЕ №5.

Для того, чтобы найти номер, пишите первое число. Пример:

$$f(\vec{x},\vec{y}) = \sum_{i=1}^{n} 22 \Big($$

22

$$f(\vec{y},\vec{x},\vec{z}) = \sum_{i=1}^{n} \arcsin^5\left(z_{n+1-\lceil i/4\rceil} - 10y_{n+1-i}^3 - x_i^2\right)$$

10

Задача №5. 96

Задача №5

Реализовать функцию, оперирующую векторами длины n:

$$f(\vec{x}, \vec{z}, \vec{y}) = 96 \sum_{i=1}^{n} 98 \left(z_{\lceil i/2 \rceil}^2 + 31 y_{n+1-\lceil i/4 \rceil}^3 + 51 x_{\lceil i/3 \rceil} \right)^5$$

Примеры результатов вычислений:

```
f([-0.83, -0.26, 0.78, -0.41, -0.49, -0.17],
[0.49, -0.18, 0.48, 0.83, -0.22, -0.6],
[0.12, -0.01, 0.57, -0.98, -0.91, 0.06]) = -4.96e+12
f([-0.52, -0.36, 0.09, 0.8, 0.62, -0.38],
[0.06, -0.81, -0.08, -0.11, 0.26, 0.44],
[0.2, 0.4, -0.23, -0.92, -0.65, -0.18]) = -6.49e+11
f([0.54, 0.63, -0.68, 0.34, 0.09, -0.01],
[0.99, -0.82, -0.07, -0.48, 0.9, 0.91],
[0.13, -0.43, -0.74, -0.89, 0.62, 0.04]) = 2.70e+12
f([0.31, 0.23, -0.62, -0.4, 0.18, -0.02],
[-0.89, -0.8, 0.4, -0.96, -0.34, -0.86],
[0.16, -0.73, 0.81, -0.32, 0.63, -0.34]) = 8.06e+10
f([0.06, 0.36, -0.07, 0.25, -0.43, -0.39],
[0.25, -0.21, -0.17, -0.56, -0.37, -0.9],
[0.75, 0.81, 0.2, 0.77, -0.16, 0.97]) = 2.99e+12
```

Решение.

```
import math


def main(x, z, y):
    ans = 0
    n = len(x)
    for i in range(1, n + 1):
        ans += 98 * (z[math.ceil(i / 2) - 1] ** 2 + 31 * (
                    y[n - math.ceil(i / 4)] ** 3) + 51 * x[
                        math.ceil(i / 3) - 1]) ** 5
    return 96 * ans
```

Задача №5. 61

Задача №5

Реализовать функцию, оперирующую векторами длины n:

$$f(\vec{y}, \vec{z}) = 61 \sum_{i=1}^{n} \cos^5 \left(1 - 47 z_{n+1-i} - 62 y_i^3 \right)$$

Примеры результатов вычислений:

```
f([0.97, 0.41, -0.63, -0.89, 0.67],
[-0.85, 0.53, -0.52, -0.12, -0.82]) = -1.14e+02
f([-0.09, -0.14, -0.71, -0.13, -0.83],
[0.74, -0.29, -0.52, 0.58, -0.01]) = -3.16e+01
f([0.48, 0.07, -0.53, -0.56, 0.23],
[-0.41, 0.08, 0.69, 0.56, -0.92]) = 6.56e+01
f([0.29, 0.07, -0.28, 0.54, -0.12],
[-0.87, -0.83, -0.68, 0.77, 0.3]) = -7.35e+01
f([0.68, 0.09, 0.82, 0.49, 0.54],
[-0.76, -0.44, 0.87, -0.61, -0.25]) = 3.50e+01
```

Решение.

```
import math


def main(y, z):
    n = len(z)
    s = 0
    for i in range(1, n + 1):
        s += (math.cos(1-47*(z[n-i])-62*((y[i-1])**3)))**5
    return 61*s
```

Задача №5. 63

Задача №5

Реализовать функцию, оперирующую векторами длины n:

$$f(\vec{x}, \vec{z}, \vec{y}) = \sum_{i=1}^{n} 63\left(x_i^2 - 23z_{n+1-\lceil i/4 \rceil} - y_i^3\right)$$

Примеры результатов вычислений:

```
f([-0.83, 0.08, -0.96, -0.95, 0.11, 0.73, 0.43],
[0.86, 0.4, -0.31, -0.98, -0.42, 0.92, -0.32],
[-0.71, 0.07, 0.94, -0.23, -0.31, -0.91, -0.9]) = -1.87e+03
f([-0.31, 0.98, -0.39, -0.03, 0.89, 0.67, 0.08],
[0.87, -0.32, 0.69, 0.62, -0.41, 0.31, 0.71],
[0.82, 0.23, 0.22, -0.98, -0.02, 0.47, 0.2]) = -5.29e+03
f([-0.65, 0.17, -0.48, 0.8, 0.38, -0.01, 0.07],
[-0.01, -0.76, 0.92, -0.48, -0.79, 0.6, 0.43],
[0.05, 0.85, 0.22, -0.87, -0.69, 0.75, -0.34]) = -5.01e+03
f([-0.18, -0.27, -0.8, 0.47, -0.93, -0.2, -0.63],
[0.81, -0.72, 0.7, -0.9, 0.78, -0.8, 0.25],
[0.73, 0.25, -0.59, 0.45, 0.17, -0.16, 0.56]) = 2.14e+03
f([0.29, -0.74, -0.41, -0.71, 0.38, 0.46, -0.56],
[0.33, 0.33, 0.73, 0.36, -0.89, 0.34, 0.81],
[0.65, -0.42, 0.78, -0.45, 0.68, -0.79, -0.86]) = -6.03e+03
```

Решение.

```
from math import ceil


def main(x, z, y):
    n = 5
    su1 = 0
    for i in range(len(x)):
        su1 += 63 * (x[i] ** 2 - 23 * z[n + 1 - ceil(i // 4)] - y[i] ** 3)
    return su1
```

Задача №5. 16

Задача №5

Реализовать функцию, оперирующую векторами длины n:

$$f(\vec{z}, \vec{y}) = \sum_{i=1}^{n} \left(16z_{n+1-\lceil i/2 \rceil}^3 - 38 - 58y_i\right)^4$$

Примеры результатов вычислений:

```
f([-0.64, 0.03, 0.83, 0.58, 0.71, -0.37, -0.3],
[0.54, 0.09, -0.74, 0.95, 0.57, 0.51, -0.68]) = 1.38e+08
f([0.48, 0.73, 0.38, 0.84, -0.6, 0.26, 0.43],
[-0.14, -0.53, -0.11, 0.39, 0.59, -0.94, -0.94]) = 4.82e+07
f([0.44, -0.8, -0.65, -0.13, -0.1, -0.28, 0.49],
[0.02, 0.23, -0.95, -0.67, -0.56, 0.46, 0.95]) = 1.01e+08
f([-0.56, 0.35, -0.42, 0.03, 0.2, -0.02, -0.18],
[-0.35, 0.54, -0.56, 0.87, 0.93, 0.27, -0.84]) = 1.64e+08
f([0.1, 0.63, -0.09, -0.08, 0.47, -0.61, 0.26],
[-0.73, 0.42, 0.05, 0.13, -0.72, -0.13, 0.64]) = 5.72e+07
```

Решение.

```
import math


def main(z, y):
    sum = 0
    for i in range(1, len(z)+1):
        sum = sum + (16*z[len(z)-math.ceil(i/2)]**3-38-58*y[i-1])**4
    return sum
```

Задача №5. 58

Задача №5

Реализовать функцию, оперирующую векторами длины n:

$$f(\vec{y}, \vec{x}) = 58 \sum_{i=1}^{n} \left(x_i^3 + \frac{y_i}{16} + 0.02 \right)^4$$

Примеры результатов вычислений:

```
f([-0.8, 0.03, 0.19, 0.6],
[-0.66, -0.96, 0.79, -0.75]) = 3.82e+01
f([0.35, -0.7, 0.41, -0.31],
[-0.29, -0.33, -0.21, -0.44]) = 3.81e-03
f([0.04, 0.16, 0.28, -0.49],
[0.65, -0.57, -0.52, 0.2]) = 4.92e-01
f([0.1, -0.26, 0.02, -0.99],
[-0.35, 0.37, -0.97, -0.85]) = 4.74e+01
f([-0.86, 0.23, -0.59, -0.54],
[-0.08, -0.98, -0.49, -0.93]) = 6.52e+01
```

Решение.

```
import math


def main(y, x):
    res = 0
    for i in range(len(y)):
        res += (x[i]**3 + y[i]/16 + 0.02)**4
    return res*58
```

Задание №5. 77

Задача №5

Реализовать функцию, оперирующую векторами длины n:

$$f(\vec{y}, \vec{z}) = 77 \sum_{i=1}^{n} \frac{\left(\lfloor y_i^2 + 1 + 15 z_{\lceil i/4 \rceil} \rfloor \right)^4}{26}$$

Примеры результатов вычислений:

```
f([0.83, 0.16, -0.05, -0.53, -0.97],
[0.25, 0.05, -0.79, 0.12, 0.68]) = 5.27e+03
f([0.92, -0.3, 0.6, -0.93, 0.46],
[-0.61, 0.36, 0.05, -0.7, -0.08]) = 5.97e+04
f([-0.33, 0.17, 0.72, 0.18, -0.38],
[-0.1, -0.31, -0.99, 0.9, 0.98]) = 7.67e+02
f([-0.2, 0.94, -0.49, -0.66, -0.02],
[-0.38, -1.0, 0.33, 0.66, -0.12]) = 1.20e+05
f([0.04, 0.97, 0.02, 0.81, -0.27],
[0.29, 0.82, -0.04, 0.76, -0.93]) = 9.60e+04
```

Решение.

```
from math import floor, ceil


def main(y, z):
    result = 0
    n = len(z)
    for i in range(1, n+1):
        result += (floor(y[i-1] ** 2 + 1 + 15 * z[ceil(i/4)-1]) ** 4) / 26
    return 77 * result
```

Задание №5. 46

Задача №5

Реализовать функцию, оперирующую векторами длины n:

$$f(\vec{x}) = \sum_{i=1}^{n} 46\left(14x_{n+1-i}^3 + x_{n+1-i}\right)$$

Примеры результатов вычислений:

```
f([0.36, 0.63, 0.97, 0.19, 0.11]) = 8.88e+02
f([0.16, -0.7, 0.3, 0.73, 0.17]) = 8.32e+01
f([-0.49, -0.25, 0.57, -0.74, -0.19]) = -2.83e+02
f([0.44, -0.83, 0.83, -0.31, 0.78]) = 3.83e+02
f([0.31, -0.64, 0.56, 0.73, -0.94]) = -3.20e+02
```

Решение.

```
import math


def main(x):
    n = len(x)
    x.insert(0, 0)
    count = sum(14 * (x[n + 1 - i])**3 + x[n + 1 - i] for i in range(1, n+1))
    return count * 46
```

Задание №5. 47

Задача №5

Реализовать функцию, оперирующую векторами длины n:

$$f(\vec{x}, \vec{y}) = \sum_{i=1}^{n} 47\left(\frac{x_{n+1-\lceil i/4 \rceil}}{86} - \frac{y_{\lceil i/2 \rceil}^3}{17}\right)^3$$

Примеры результатов вычислений:

```
f([-0.52, 0.21, -0.86, -0.46],
[0.13, -0.51, 0.46, 0.77]) = -1.41e-05
f([0.02, -0.84, 0.78, -0.52],
[0.24, 0.19, 0.93, 0.99]) = -5.56e-05
f([-0.62, -0.08, -0.67, 0.18],
[-0.04, -0.67, -0.37, -0.22]) = 7.29e-04
f([0.72, 0.02, -0.93, -0.76],
[-0.97, 0.57, 0.28, -0.25]) = 7.76e-03
f([0.43, -0.86, 0.7, -0.55],
[0.58, -0.07, -0.54, -0.59]) = -5.61e-04
```

Решение.

```
from math import ceil


def main(x, y):
    result = 0
    n = len(y)
    y = [0] + y
    x = [0] + x
    for i in range(1, n + 1):
        a = x[n + 1 - ceil(i / 4)] / 86
        b = (y[ceil(i / 2)]) ** 3 / 17
        result += 47 * (a - b) ** 3
    return result
```

Задание №5. 66

Задача №5

Реализовать функцию, оперирующую векторами длины n:

$$f(\vec{z}) = \sum_{i=1}^{n} \left(z_{\lceil i/3 \rceil}^3 - \frac{z_{n+1-\lceil i/3 \rceil}}{66} \right)^5$$

Примеры результатов вычислений:

```
f([0.87, 0.34, -0.2, -0.63, 0.44, -0.83, 0.35, -0.57]) = 3.96e-01
f([0.24, 0.79, 0.91, -0.68, 0.06, -0.71, -0.03, -0.01]) = 6.10e-01
f([0.2, -0.08, 0.94, 0.19, 0.36, 0.56, 0.45, 0.14]) = 7.51e-01
f([-0.14, 0.04, -0.62, 0.34, -0.04, -0.24, 0.46, 0.51]) = -1.42e-03
f([-0.14, -0.3, -0.12, -0.38, 0.1, 0.59, -0.35, 0.45]) = -1.49e-08
```

Решение.

```
from math import ceil


def main(z):
    n = len(z)
    z.insert(0, 0)
    suma = 0
    for i in range(1, n+1):
        p = z[ceil(i/3)]
        d = z[n + 1 - ceil(i/3)]
        sum1 = (p**3 - d/66)**5
        suma = suma + sum1
    return suma
```

Задание №5. 22

Задача №5

Реализовать функцию, оперирующую векторами длины n:

$$f(\vec{x}, \vec{y}) = \sum_{i=1}^{n} 22 \left(17x_{\lceil i/2 \rceil}^2 - 97y_{n+1-\lceil i/2 \rceil}^3 - 2 \right)^7$$

Примеры результатов вычислений:

```
f([-0.45, -0.95, 0.89, 0.63, 0.15, 0.82],
[0.79, 0.02, 0.72, 0.52, 0.7, 0.19]) = -5.49e+10
f([0.24, 0.86, 0.72, 0.45, -0.62, 0.25],
[-0.05, 0.8, -0.75, -0.56, -0.8, 0.36]) = 1.27e+14
f([-0.09, 0.75, 0.67, 0.49, -0.77, -0.01],
[0.47, -0.29, -0.38, -0.9, -0.55, 0.45]) = 6.65e+14
f([-0.57, -0.06, -0.45, -0.65, -0.26, 0.27],
[-0.05, 0.91, 0.37, -0.94, 0.41, -0.69]) = 1.10e+15
f([-0.47, -0.85, 0.37, -0.65, 0.85, -0.03],
[0.13, -0.03, -0.5, -0.56, 0.05, -0.47]) = 2.29e+10
```

Решение.

```
from math import ceil


def main(x, y):
    n = len(x)
    res = 0
    for i in range(1, n+1):
        res += 22*(17*x[ceil(i/2)-1] ** 2 - 97 * (y[n-ceil(i/2)] ** 3) - 2)**7
    return res
```

Задание №5. 3

Задача №5

Реализовать функцию, оперирующую векторами длины n:

$$f(\vec{x}, \vec{y}) = \sum_{i=1}^{n} 3\left(45y_{\lceil i/3 \rceil} - 50y_{\lceil i/3 \rceil}^2 - 52x_{n+1-i}^3\right)$$

Примеры результатов вычислений:

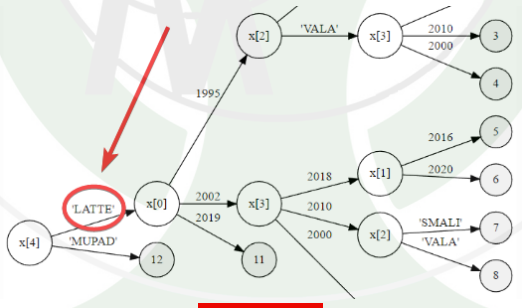
```
f([-0.82, -0.62, -0.58, 0.83, -0.19, -0.49, 0.93],
[0.29, -0.85, -0.97, -0.21, -0.52, 0.67, -0.38]) = -9.03e+02
f([-0.5, -0.57, 0.06, -0.16, -0.44, -0.06, -0.63],
[0.51, -0.29, -0.16, -0.62, -0.72, -0.34, 0.56]) = 1.01e+01
f([0.37, 0.27, 0.43, 0.93, 0.44, -0.75, 0.57],
[-0.98, -0.02, 0.84, 0.09, 0.21, 0.07, -0.56]) = -9.55e+02
f([-0.85, -0.99, -0.12, 0.97, 0.55, -0.35, -0.77],
[0.16, 0.34, 0.69, -0.84, 0.69, 0.67, -0.11]) = 3.18e+02
f([0.49, 0.9, 0.72, 0.56, -0.2, -0.96, -0.35],
[-0.82, 0.05, 0.07, 0.7, -0.04, -0.52, 0.01]) = -6.79e+02
```

Решение.

```
import math


def main(*lst):
    def f(y, x):
        return 3 * ((45 * y) - (50 * y ** 2) - (52 * x ** 3))

    c, z = lst
    res = 0
    n = len(c)
    for i in range(n):
        res += f(z[math.ceil(i // 3)], c[n - 1 - i])
    return res
```

ЗАДАНИЕ №6.

Для того, чтобы найти номер, пишите первое число или слово. Пример:

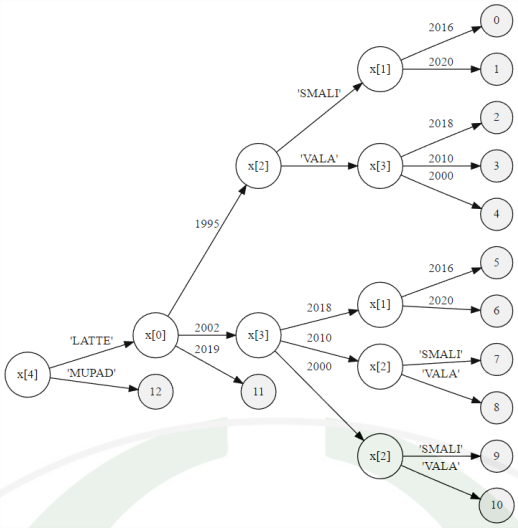


LATTE

Задание №6. LATTE

Задача №6

Реализовать функцию для вычисления дерева решений:



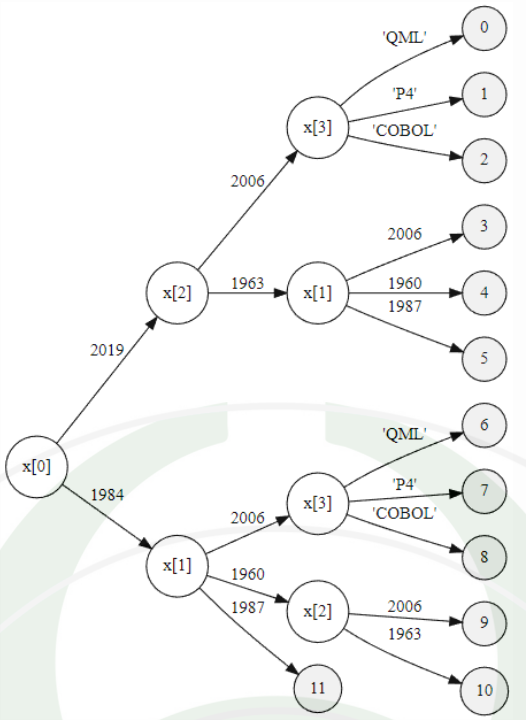
Решение.

```
def main(arr):
    default = [
        [1995, 2016, "SMALI", arr[3], "LATTE"],  # 0
        [1995, 2020, "SMALI", arr[3], "LATTE"],  # 1
        [1995, arr[1], "VALA", 2018, "LATTE"],  # 2
        [1995, arr[1], "VALA", 2010, "LATTE"],  # 3
        [1995, arr[1], "VALA", 2000, "LATTE"],  # 4
        [2002, 2016, arr[2], 2018, "LATTE"],  # 5
        [2002, 2020, arr[2], 2018, "LATTE"],  # 6
        [2002, arr[1], "SMALI", 2010, "LATTE"],  # 7
        [2002, arr[1], "VALA", 2010, "LATTE"],  # 8
        [2002, arr[1], "SMALI", 2000, "LATTE"],  # 9
        [2002, arr[1], "VALA", 2000, "LATTE"],  # 10
        [2019, arr[1], arr[2], arr[3], "LATTE"],  # 11
        [arr[0], arr[1], arr[2], arr[3], "MUPAD"]  # 1
    ]

    return default.index(arr)
```

Задание №6. 2019

Задача №6

Реализовать функцию для вычисления дерева решений:



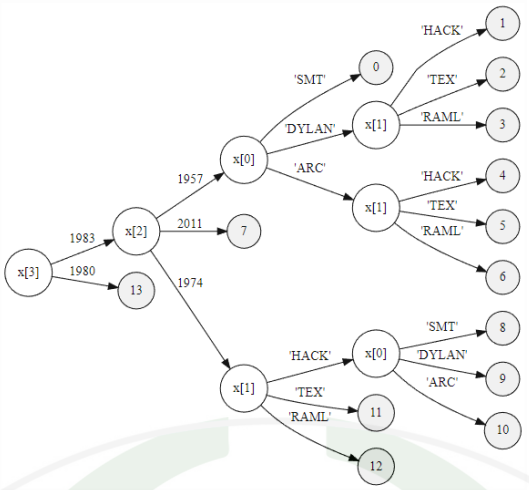
Решение.

```
def main(res):
    ans = {0: {1984: {1: {1987: 11, 1960: {2: {2006: 9, 1963: 10}},
                          2006: {
                              3: {'COBOL': 8, 'P4': 7, 'QML': 6}}}},
               2019: {2: {1963: {1: {1987: 5, 1960: 4, 2006: 3}},
                          2006: {
                              3: {'COBOL': 2, 'P4': 1, 'QML': 0}}}}}}
    while isinstance(ans, dict):
        Ind = int(*ans)
        ans = ans[Ind]
        Key = res[Ind]
        ans = ans[Key]
    return ans
```

Задание №6. 1983

Задача №6

Реализовать функцию для вычисления дерева решений:



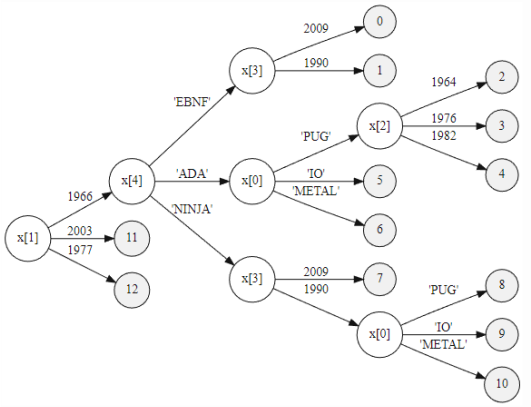
Решение.

```
def main(x):
    if x[3] == 1980:
        return 13
    elif x[3] == 1983:
        if x[2] == 2011:
            return 7
        elif x[2] == 1974:
            return f(x)
        elif x[2] == 1957:
            return s(x)


def f(x):
    if x[1] == 'RAML':
        return 12
    elif x[1] == 'TEX':
        return 11
    elif x[1] == 'HACK':
        if x[0] == 'ARC':
            return 10
        elif x[0] == 'DYLAN':
            return 9
        elif x[0] == 'SMT':
            return 8


def s(x):
    if x[0] == 'SMT':
        return 0
    elif x[0] == 'DYLAN':
        if x[1] == 'HACK':
            return 1
        elif x[1] == 'TEX':
            return 2
        elif x[1] == 'RAML':
            return 3
    elif x[0] == 'ARC':
        if x[1] == 'HACK':
            return 4
        elif x[1] == 'TEX':
            return 5
        elif x[1] == 'RAML':
            return 6
```

Задание №6. 1966

Задача №6

Реализовать функцию для вычисления дерева решений:

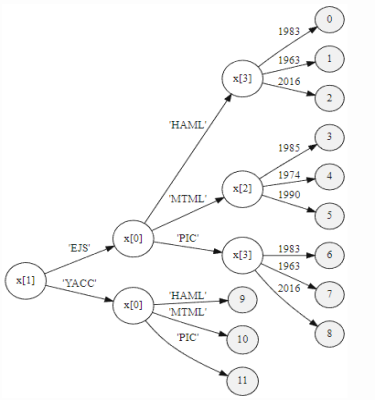


Решение.

```
s = ({1966, 'EBNF', 2009},
     {1966, 'EBNF', 1990},
     {1966, 'ADA', 'PUG', 1964},
     {1966, 'ADA', 'PUG', 1976},
     {1966, 'ADA', 'PUG', 1982},
     {1966, 'ADA', 'IO'},
     {1966, 'ADA', 'METAL'},
     {1966, 'NINJA', 2009},
     {1966, 'NINJA', 1990, 'PUG'},
     {1966, 'NINJA', 1990, 'IO'},
     {1966, 'NINJA', 1990, 'METAL'},
     {2003},
     {1977})


def main(r):
    s1 = set(r)
    s2 = [i for i in range(len(s)) if not(len(s[i] - s1))][0]
    return s2
```

Задание №6. EJS

Реализовать функцию для вычисления дерева решений:



Примеры результатов вычислений:

```
main(['PIC', 'YACC', 1990, 1963]) = 11
```

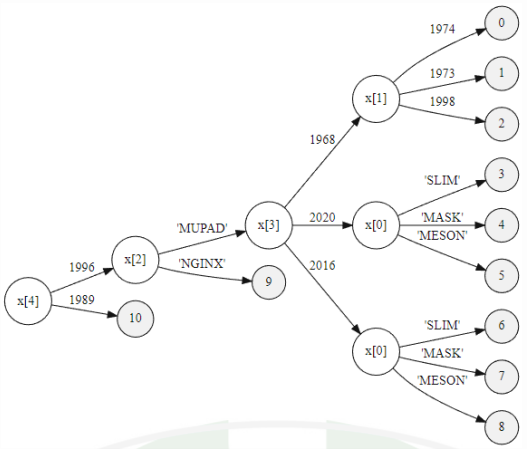
Решение.

```
def main(arr):
    default = [
        ["HAML", "EJS", arr[2], 1983],      # 0
        ["HAML", "EJS", arr[2], 1963],       # 1
        ["HAML", "EJS", arr[2], 2016],       # 2
        ["MTML", "EJS", 1985, arr[3]],      # 3
        ["MTML", "EJS", 1974, arr[3]],     # 4
        ["MTML", "EJS", 1990, arr[3]],       # 5
        ["PIC", "EJS", arr[2], 1983],      # 6
        ["PIC", "EJS", arr[2], 1963],       # 7
        ["PIC", "EJS", arr[2], 2016],       # 8
        ["HAML", "YACC", arr[2], arr[3]],     # 9
        ["MTML", "YACC", arr[2], arr[3]],      # 10
        ["PIC", "YACC", arr[2], arr[3]],      # 11
    ]

    return default.index(arr)
```

Задание №6. 1996 MUPAD

Задача №6

Реализовать функцию для вычисления дерева решений:



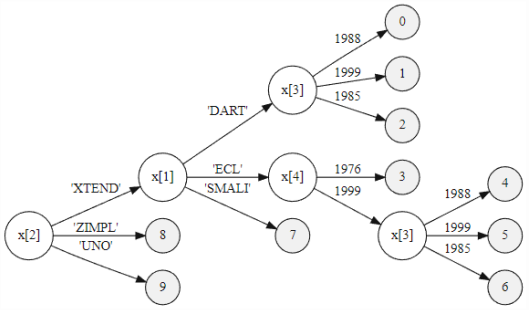
Решение.

```
x = {(1974, 1968): 0,
     (1973, 1968): 1,
     (1998, 1968): 2,
     ('SLIM', 2020): 3,
     ('MASK', 2020): 4,
     ('MESON', 2020): 5,
     ('SLIM', 2016): 6,
     ('MASK', 2016): 7,
     ('MESON', 2016): 8,
     ('NGINX', 1995): 9,
     ('XOJO', 1995): 10, }


def main(arr):
    if arr[4] == 1989:
        return 10
    elif arr[2] == 'NGINX':
        return 9
    else:
        arr.pop(4)
        arr.pop(2)
        if arr[2] == 1968:
            arr.pop(0)
        else:
            arr.pop(1)
        arr = tuple(arr)
        return x[arr]
```

Задание №6. XTEND

Задача №6

Реализовать функцию для вычисления дерева решений:



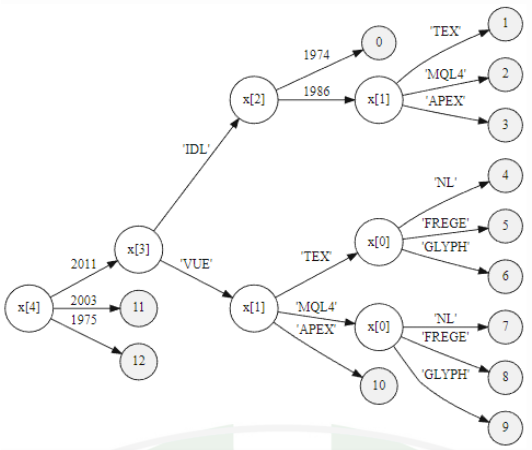
Решение.

```
def main(x):
    dict3 = {1988: 4, 1999: 5, 1985: 6}
    dict3_up = {1988: 0, 1999: 1, 1985: 2}
    dict4 = {1976: 3, 1999: dict3[x[3]]}
    dict1 = {'DART': dict3_up[x[3]], 'ECL': dict4[x[4]], 'SMALI': 7}
    dict2 = {'XTEND': dict1[x[1]], 'ZIMPL': 8, 'UNO': 9}
    return dict2[x[2]]
```

Задание №6. 2011

Задача №6

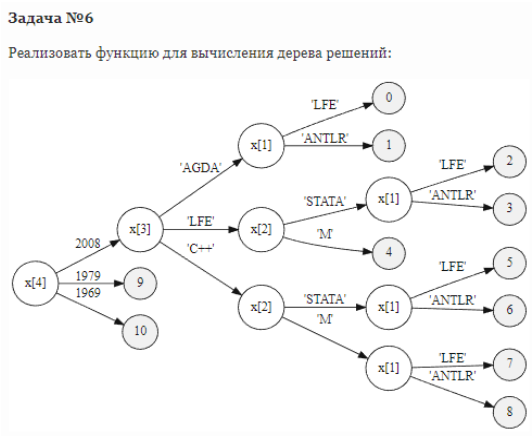
Реализовать функцию для вычисления дерева решений:



Решение.

```
def main(arr):
    default = [
        [arr[0], arr[1], 1974, "IDL", 2011],      # 0
        [arr[0], "TEX", 1986, "IDL", 2011],       # 1
        [arr[0], "MQL4", 1986, "IDL", 2011],       # 2
        [arr[0], "APEX", 1986, "IDL", 2011],      # 3
        ["NL", "TEX", arr[2], "VUE", 2011],     # 4
        ["FREGE", "TEX", arr[2], "VUE", 2011],       # 5
        ["GLYPH", "TEX", arr[2], "VUE", 2011],      # 6
        ["NL", "MQL4", arr[2], "VUE", 2011],       # 7
        ["FREGE", "MQL4", arr[2], "VUE", 2011],       # 8
        ["GLYPH", "MQL4", arr[2], "VUE", 2011],     # 9
        [arr[0], "APEX", arr[2], "VUE", 2011],     # 10
        [arr[0], arr[1], arr[2], arr[3], 2003],      # 11
        [arr[0], arr[1], arr[2], arr[3], 1975],     # 12
    ]

    return default.index(arr)
```

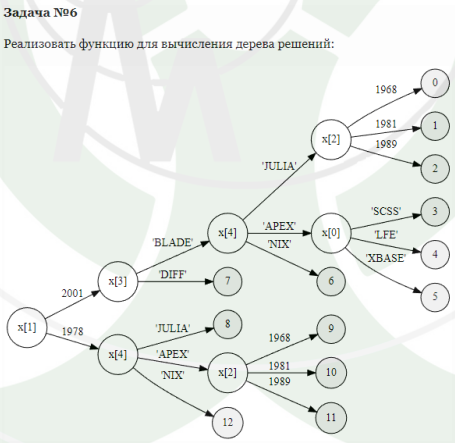
Задание №6. 2008



Решение.

```
def main(x):
    default = [
        [x[0], "LFE",   x[2], "AGDA", 2008],     # 0
        [x[0], "ANTLR", x[2], "AGDA", 2008],     # 1
        [x[0], "LFE", "STATA", "LFE", 2008],    # 2
        [x[0], "ANTLR", "STATA", "LFE", 2008],     # 3
        [x[0], x[1], "M", "LFE", 2008],     # 4
        [x[0], "LFE", "STATA", "C++", 2008],     # 5
        [x[0], "ANTLR", "STATA", "C++", 2008],     # 6
        [x[0], "LFE", "M", "C++", 2008],     # 7
        [x[0], "ANTLR", "M", "C++", 2008],     # 8
        [x[0], x[1], x[2], x[3], 1979],     # 9
        [x[0], x[1], x[2], x[3], 1969],     # 10
    ]

    return default.index(x)
```

Задание №6. 2001



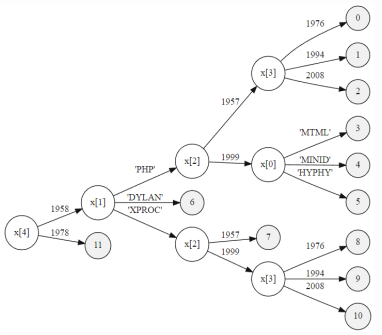
Решение.

```
def main(x):
    dict2 = {1968: 0, 1981: 1, 1989: 2}
    dict0 = {'SCSS': 3, 'LFE': 4, 'XBASE': 5}
    dict4 = {'JULIA': dict2[x[2]], 'APEX': dict0[x[0]], 'NIX': 6}
    dict2_up = {1968: 9, 1981: 10, 1989: 11}
    dict3 = {'BLADE': dict4[x[4]], 'DIFF': 7}
    dict4_up = {'JULIA': 8, 'APEX': dict2_up[x[2]], 'NIX': 12}
    dict1 = {2001: dict3[x[3]], 1978: dict4_up[x[4]]}
    return dict1[x[1]]
```

Задание №6. 1958

Задача №6

Реализовать функцию для вычисления дерева решений:



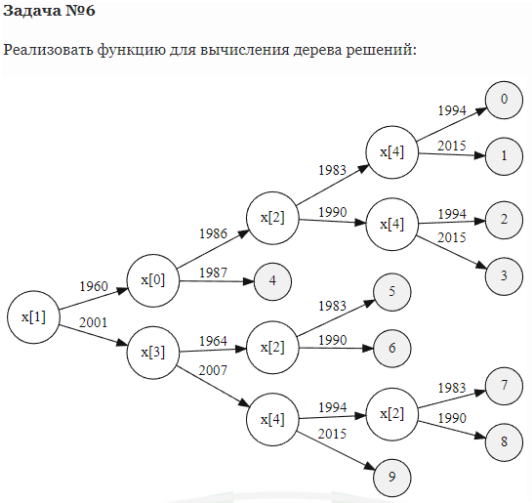
Примеры результатов вычислений:

```
main(['MINID', 'XPROC', 1999, 1994, 1958]) = 9
```

Решение.

```
def main(x):
    dict41 = {1976: 0, 1994: 1, 2008: 2}
    dict42 = {'MTML': 3, 'MINID': 4, 'HYPHY': 5}
    dict43 = {1976: 8, 1994: 9, 2008: 10}
    dict31 = {1957: dict41[x[3]], 1999: dict42[x[0]]}
    dict32 = {1957: 7, 1999: dict43[x[3]]}
    dict21 = {'PHP': dict31[x[2]], 'DYLAN': 6, 'XPROC': dict32[x[2]]}
    dict1 = {1958: dict21[x[1]], 1978: 11}
    return dict1[x[4]]
```

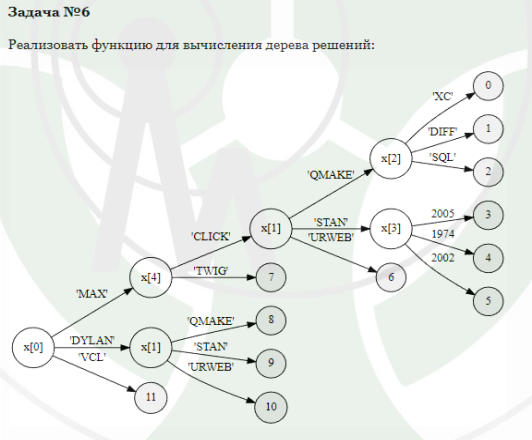
Задание №6. 1960



Решение.

```
def main(x):
    dict41 = {1994: 0, 2015: 1}
    dict42 = {1994: 2, 2015: 3}
    dict21 = {1983: dict41[x[4]], 1990: dict42[x[4]]}
    dict0 = {1986: dict21[x[2]], 1987: 4}
    dict22 = {1983: 7, 1990: 8}
    dict43 = {1994: dict22[x[2]], 2015: 9}
    dict23 = {1983: 5, 1990: 6}
    dict3 = {1964: dict23[x[2]], 2007: dict43[x[4]]}
    dict1 = {1960: dict0[x[0]], 2001: dict3[x[3]]}
    return dict1[x[1]]
```

Задание №6. MAX



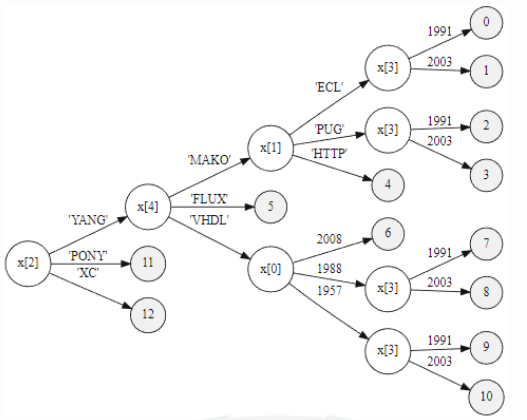
Решение.

```
def main(x):
    dir2 = {'XC': 0, 'DIFF': 1, 'SQL': 2}
    dir3 = {2005: 3, 1974: 4, 2002: 5}
    dir_up1 = {'QMAKE': dir2[x[2]], 'STAN': dir3[x[3]], 'URWEB': 6}
    dir4 = {'CLICK': dir_up1[x[1]], 'TWIG': 7}
    dir_d_1 = {'QMAKE': 8, 'STAN': 9, 'URWEB': 10}
    dir0 = {'MAX': dir4[x[4]], 'DYLAN': dir_d_1[x[1]], 'VCL': 11}
    return dir0[x[0]]
```

Задание №6. YANG

Задача №6

Реализовать функцию для вычисления дерева решений:



Решение.

```
s = ({'YANG', 'MAKO', 'ECL', 1991}, #Change list name on s
     {'YANG', 'MAKO', 'ECL', 2003},
     {'YANG', 'MAKO', 'PUG', 1991},
     {'YANG', 'MAKO', 'PUG', 2003},
     {'YANG', 'MAKO', 'HTTP'},
     {'YANG', 'FLUX'},
     {'YANG', 'VHDL', 2008},
     {'YANG', 'VHDL', 1988, 1991},
     {'YANG', 'VHDL', 1988, 2003},
     {'YANG', 'VHDL', 1957, 1991},
     {'YANG', 'VHDL', 1957, 2003},
     {'PONY'},
     {'XC'})


def main(list):
    s1 = set(list)
    return [i for i in range(len(s))
            if not(len(s[i] - s1))][0]
```

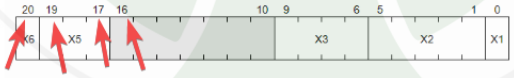
ЗАДАНИЕ №7.

Для того, чтобы найти номер, пишите первые четыре числа подряд. Пример:

Реализовать функцию для декодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции. Неиспользуемые поля результата должны содержать нулевые биты.

Входные данные:

Шестнадцатиричная строка.



20191716

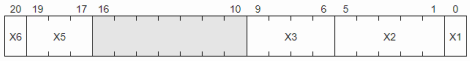
Задание №7. 20191710

Задача №7

Реализовать функцию для декодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции. Неиспользуемые поля результата должны содержать нулевые биты.

Входные данные:

Шестнадцатиричная строка.



Выходные данные:

Кортеж из битовых полей в порядке от младших бит к старшим. Значения битовых полей имеют тип: десятичная строка.

Тесты:

```
>>> main('0x86afe')
('0', '31', '11', '4', '0')
```

Решение.

```
def main(num):
    num = int(num, 16)

    x6 = str((num >> 20) & 0b1)
    x5 = str((num >> 17) & 0b111)
    x3 = str((num >> 6) & 0b1111)
    x2 = str((num >> 1) & 0b11111)
    x1 = str((num >> 0) & 0b1)

    return (x1, x2, x3, x5, x6)


if __name__ == "__main__":
    print(main('0x86afe'))
    print(main('0x18056a'))
    print(main('0x65f88'))
    print(main('0x7b946'))
```

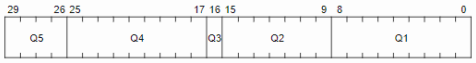
Задание №7. 29262517

Задача №7

Реализовать функцию для декодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции.

Входные данные:

Целое.



Выходные данные:

Кортеж из битовых полей в порядке от младших бит к старшим. Значения битовых полей имеют тип: целое.

Тесты:

```
>>> main(971354762)
(138, 89, 1, 242, 14)
```

Решение.

```
def main(n):
    n = int(n)
    A = ((int(n) >> 0) & int(
        (32 - (8 - 0 + 1)) * '0' + (8 - 0 + 1) * '1',
        2))
    B = ((int(n) >> 9) & int(
        (32 - (15 - 9 + 1)) * '0' + (15 - 9 + 1) * '1',
        2))  # << b[3]
    C = ((int(n) >> 16) & int(
        (32 - (16 - 16 + 1)) * '0' + (16 - 16 + 1) * '1',
        2))  # << b[3]
    D = ((int(n) >> 17) & int(
        (32 - (25 - 17 + 1)) * '0' + (25 - 17 + 1) * '1',
        2))
    E = ((int(n) >> 26) & int(
        (32 - (29 - 26 + 1)) * '0' + (29 - 26 + 1) * '1',
        2))
    # << a[3]
    # print(A,B,C,D,E)
    # ans = [('A1', A),('A2',B), ('A3', C), ('A4', D), ('A5', E)]
    ans = (A, B, C, D, E)
    # ans.append(bin(n)[0:11])

    return ans
```

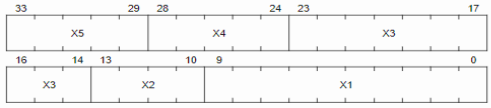
Задание №7.33292824

Задача №7

Реализовать функцию для транскодирования данных, содержащих битов
поля. В решении необходимо использовать побитовые операции.

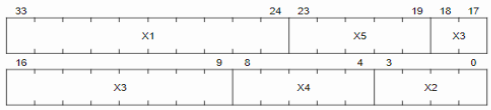
Входные данные:

Шестнадцатиричная строка.



Выходные данные:

Целое.



Тесты:

```
>>> main('0x2c845fb48')
14104538766
```

Решение.

```
def main(num):
    num = int(num, 16)
    x5 = (num >> 29) & 0b11111
    x4 = (num >> 24) & 0b11111
    x3 = (num >> 14) & 0b1111111111
    x2 = (num >> 10) & 0b1111
    x1 = (num >> 0) & 0b1111111111

    res = (x1 << 24) | (x5 << 19) | (x3 << 9) | (x4 << 4) | (x2 << 0)
    return int(res)
```

Задание №7. 21201918

Задача №7

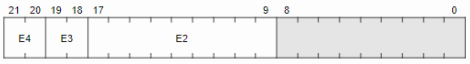
Реализовать функцию для кодирования данных, содержащих битовые поля.
В решении необходимо использовать побитовые операции.
Неиспользуемые поля результата должны содержать нулевые биты.

Входные данные:

Кортеж из битовых полей в порядке от младших бит к старшим. Значения битовых полей имеют тип: целое.

Выходные данные:

Шестнадцатиричная строка.



Тесты:

```
>>> main((393, 2, 1))
'0x1b1200'
```

Решение.

```
def main(x):
    x = [i for i in x]
    e1 = 0b0000_0000_0
    e2 = x[0] & 0b1111_1111_1
    e3 = x[1] & 0b11
    e4 = x[2] & 0b11
    result = e1 | (e2 << 9) | (e3 << 18) | (e4 << 20)
    return str(hex(result))
```

Задание №7. 33292819

Задача №7

Реализовать функцию для кодирования данных, содержащих битовые поля.
В решении необходимо использовать побитовые операции.
Неиспользуемые поля результата должны содержать нулевые биты.

Входные данные:

Список из битовых полей в виде пар имя-значение. Значения битовых полей имеют тип: десятичная строка.

Выходные данные:

Шестнадцатиричная строка.

Тесты:

```
>>> main([('O1', '89'), ('O2', '185'), ('O4', '27')])
'0x36002e459'
```

Решение.

```
def main(arr):
    r4 = int(arr[2][1]) & 0b111
    r3 = int(arr[1][1]) & 0b1
    r1 = int(arr[0][1]) & 0b1111

    res = (r4 << 7) | (r3 << 6) | (r1 << 0)

    return res
```

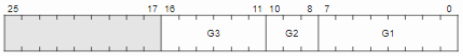
Задание №7. 25171611

Задача №7

Реализовать функцию для транскодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции. Неиспользуемые поля результата должны содержать нулевые биты.

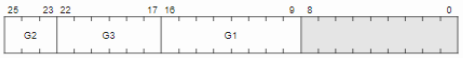
Входные данные:

Шестнадцатиричная строка.



Выходные данные:

Десятичная строка.



Тесты:

```
>>> main('0x5bf2')
```

Решение.

```
def main(number):
    num = int(number, 16)

    g1 = (num >> 0) & 0b11111111
    g2 = (num >> 8) & 0b111
    g3 = (num >> 11) & 0b111111

    result = (g2 << 23) | (g3 << 17) | (g1 << 9)

    return str(result)
```

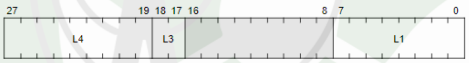
Задание №7. 27191817

Задача №7

Реализовать функцию для транскодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции. Неиспользуемые поля результата должны содержать нулевые биты.

Входные данные:

Десятичная строка.



Выходные данные:

Десятичная строка.

Тесты:

```
>>> main('40062127')
'39846319'
```

Решение.

```
def main(s):
    i = int(s)
    L1 = 0b11111111 & i
    L3 = 0b11 & (i >> 17)
    L4 = 0b111111111 & (i >> 19)
    result = (L4 << 19) | (L3 << 8) | L1
    return str(result)
```

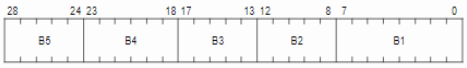
Задание №7. 28242318

Задача №7

Реализовать функцию для транскодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции.

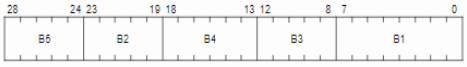
Входные данные:

Десятичная строка.



Выходные данные:

Десятичная строка.



Тесты:

```
>>> main('266654201')
```

Решение.

```
def main(number):
    num = int(number)

    b1 = (num >> 0) & 0b11111111
    b2 = (num >> 8) & 0b11111
    b3 = (num >> 13) & 0b11111
    b4 = (num >> 18) & 0b111111
    b5 = (num >> 24) & 0b11111

    result = (b5 << 24) | (b2 << 19) | (b4 << 13) | (b3 << 8) | b1
    return str(result)
```

Задание №7. 31232218

Задача №7

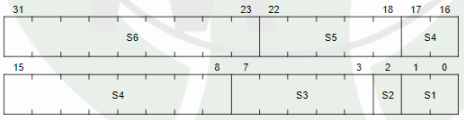
Реализовать функцию для кодирования данных, содержащих битовые поля.
В решении необходимо использовать побитовые операции.

Входные данные:

Словарь из битовых полей. Значения битовых полей имеют тип: десятичная строка.

Выходные данные:

Десятичная строка.



Тесты:

```
>>> main({'S1': '1', 'S2': '0', 'S3': '1', 'S4': '288', 'S5': '25', 'S
'1558519817'
```

Решение.

```
def main(n):
    s1 = int(n['S1'])
    s2 = int(n['S2'])
    s3 = int(n['S3'])
    s4 = int(n['S4'])
    s5 = int(n['S5'])
    s6 = int(n['S6'])
    r = (((s6 << 5) | s5) << 10) | s4
    r = (((r << 5) | s3) << 1) | s2
    return str((r << 2) | s1)
```

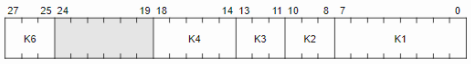
Задание №7. 27252419

Задача №7

Реализовать функцию для транскодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции. Неиспользуемые поля результата должны содержать нулевые биты.

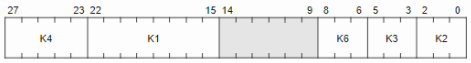
Входные данные:

Десятичная строка.



Выходные данные:

Десятичная строка.



Тесты:

```
>>> main('164577742')
'250020097'
```

Решение.

```
def main(x):
    x = int(x)
    k1 = x & 0b1111_1111
    k2 = (x >> 8) & 0b111
    k3 = (x >> 11) & 0b111
    k4 = (x >> 14) & 0b1111_1
    k5 = (x >> 19) & 0b0000_00
    k6 = (x >> 25) & 0b111
    result = k2 | (k3 << 3) | (k6 << 6) | (k5 << 9) | (k1 << 15) | (k4 << 23)
    return str(result)
```

Задание №7. 131276

Задача №7

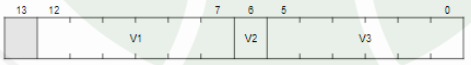
Реализовать функцию для транскодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции. Неиспользуемые поля результата должны содержать нулевые биты.

Входные данные:

Целое.

Выходные данные:

Десятичная строка.



Тесты:

```
>>> main(623)
```

Решение.

```
def main(num):
    v1 = (num >> 0) & 0b111111
    v2 = (num >> 6) & 0b1
    v3 = (num >> 7) & 0b111111

    result = int((v1 << 7) | (v2 << 6) | (v3 << 0))
    return str(result)
```

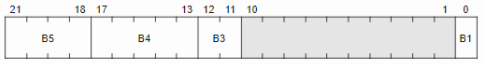
Задание №7. 21181713

Задача №7

Реализовать функцию для декодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции. Неиспользуемые поля результата должны содержать нулевые биты.

Входные данные:

Десятичная строка.



Выходные данные:

Кортеж из битовых полей в порядке от младших бит к старшим. Значения битовых полей имеют тип: целое.

Тесты:

```
>>> main('1826647')
(1, 3, 30, 6)
```

Решение.

```
def main(x):
    x = int(x)
    b1 = x & 0b1
    b3 = (x >> 11) & 0b11
    b4 = (x >> 13) & 0b1111_1
    b5 = (x >> 18) & 0b1111
    result = (b1, b3, b4, b5)
    return result


print(main('1826647'))
```

Задание №7. 15141110

Задача №7

Реализовать функцию для транскодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции.

Входные данные:

Шестнадцатиричная строка.

Выходные данные:

Целое.

Тесты:

```
>>> main('0xfd82')
45182
```

Решение.

```
def main(x):
    x = int(x, 16)
    l1 = x & 0b1
    l2 = (x >> 1) & 0b1111_1111_11
    l3 = (x >> 11) & 0b1111
    l4 = (x >> 15) & 0b1
    result = l1 | (l3 << 1) | (l4 << 5) | (l2 << 6)
    return result
```

Задание №7. 34333225

Задача №7

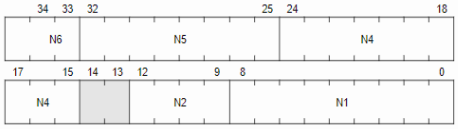
Реализовать функцию для кодирования данных, содержащих битовые поля.
В решении необходимо использовать побитовые операции.
Неиспользуемые поля результата должны содержать нулевые биты.

Входные данные:

Словарь из битовых полей. Значения битовых полей имеют тип: целое.

Выходные данные:

Десятичная строка.



Тесты:

```
>>> main({'N1': 402, 'N2': 12, 'N4': 758, 'N5': 128, 'N6': 3})
```

Решение.

```
def encode(fields):
    mask1 = 2**9 - 1
    mask2 = (2**5 - 1) << 9
    mask4 = (2**11 - 1) << 15
    mask5 = (2**8 - 1) << 25
    mask6 = (2**2 - 1) << 33
    n1 = fields.get('N1', 0) & mask1
    n2 = (fields.get('N2', 0) << 9) & mask2
    n4 = (fields.get('N4', 0) << 15) & mask4
    n5 = (fields.get('N5', 0) << 25) & mask5
    n6 = (fields.get('N6', 0) << 33) & mask6
    encoded = n1 | n2 | n4 | n5 | n6
    return str(encoded)


def main(fields):
    return encode(fields)
```

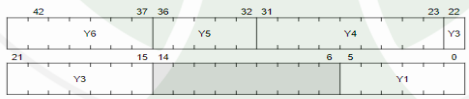
Задание №7. 42373632

Задача №7

Реализовать функцию для транскодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции.
Неиспользуемые поля результата должны содержать нулевые биты.

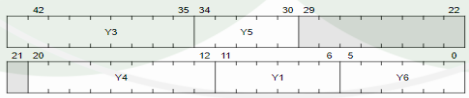
Входные данные:

Целое.



Выходные данные:

Шестнадцатиричная строка.



Тесты:

```
>>> main(1637805072997)
'0x4a7400a994b'
```

Решение.

```
def main(num):
    y6 = (num >> 37) & 0b1111111
    y5 = (num >> 32) & 0b11111
    y4 = (num >> 23) & 0b111111111
    y3 = (num >> 15) & 0b11111111
    y1 = (num >> 0) & 0b111111

    res = (y3 << 35) | (y5 << 30) | (y4 << 12) | (y1 << 6) | (y6 << 0)
    return(hex(res))
```

Задание №7. 25201918

Задача №7

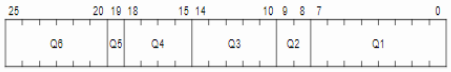
Реализовать функцию для кодирования данных, содержащих битовые поля. В решении необходимо использовать побитовые операции.

Входные данные:

Словарь из битовых полей. Значения битовых полей имеют тип: десятичная строка.

Выходные данные:

Шестнадцатиричная строка.



Тесты:

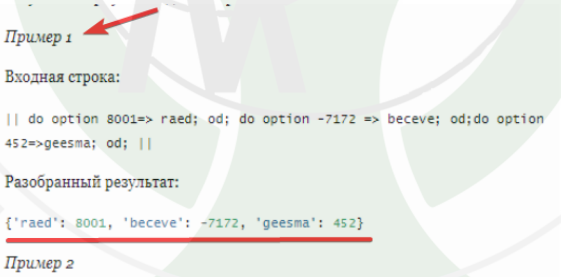
```
>>> main({'Q1': '212', 'Q2': '0', 'Q3': '31', 'Q4': '1', 'Q5': '0', 'Q
'0x230fcd4'
```

Решение.

```
def main(d):
    d1 = bin(int(d['Q1']))[2:].zfill(8)
    d2 = bin(int(d['Q2']))[2:].zfill(2)
    d3 = bin(int(d['Q3']))[2:].zfill(5)
    d4 = bin(int(d['Q4']))[2:].zfill(4)
    d5 = bin(int(d['Q5']))[2:].zfill(1)
    d6 = bin(int(d['Q6']))[2:].zfill(6)
    return(hex(int(d6 + d5 + d4 + d3 + d2 + d1, 2)))
```

ЗАДАНИЕ №8.

Для того, чтобы найти номер, пишите первые четыре числа подряд. Пример:



Пример 1

Входная строка:

```
|| do option 8001=> raed; od; do option -7172 => beceve; od;do option
452=>geesma; od; ||
```

Разобранный результат:

```
{'raed': 8001, 'beceve': -7172, 'geesma': 452}
```

Пример 2

{ `raed`: 801, `beceve`: -7172, `geesma`: 452}

Задание №8. [(`beza`, [7356, 6390, -6108]),

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде списка пар.

Пример 1

Входная строка:

```
do .begin option "beza" ::= #(#7356 . #6390 . #-6108); .end .begin
option "beorin" ::= #( #-151 . #9442); .end .begin option "tive" ::=
#( #-5399 .#1343 . #-9982 .#-3722 ); .end end
```

Разобранный результат:

```
[('beza', [7356, 6390, -6108]),
 ('beorin', [-151, 9442]),
 ('tive', [-5399, 1343, -9982, -3722])]
```

Пример 2

Входная строка:

```
do .begin option"orinle"::= #( #7953 . #5584 . #-7383); .end .begin
option "cequri" ::= #( #4724 . #8727 ); .end .begin option "atxela"::=
#( #1060 . #3617); .end .begin option"rezave"::= #( #-1047 . #-2657 .
#8110); .end end
```

Разобранный результат:

```
[('orinle', [7953, 5584, -7383]),
```

Решение.

```
import re


def main(input_string):
    pattern = r'option\s?\"(\w+)\"\s?::=\s?#\(([^)]+)\);'
    matches = re.findall(pattern, input_string.replace("\n", " "))

    result = []
    for match in matches:
        name = match[0]
        values = [int(num.replace("#", "")) for num in match[1].split('.')]
        result.append((name, values))

    return result
```

Задание №8. {`onma`: `mation`, `retebi`: `lama_633`, `xeenor_382`: `tianes_997`}

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде словаря.

Пример 1

Входная строка:

```
<sect><< var 'onma': mation. >>; <<var'retebi' : lama_633.>>; <<var 'xeenor_382' :tianes_997.>>; </sect>
```

Разобранный результат:

```
{'onma': 'mation', 'retebi': 'lama_633', 'xeenor_382': 'tianes_997'}
```

Пример 2

Входная строка:

```
<sect><<var'ined' : isa. >>; <<var 'diage' :raen. >>; <<var 'isqu' : begece_658.>>;<<var 'veedar_372' : usri. >>; </sect>
```

Разобранный результат:

```
{'ined': 'isa',
```

Решение.

```
import re


def main(s):
    ans = {}
    s = s.replace("\n", " ")
    d = re.findall(r'var(.*?)[.]', s)
    # print(d)
    for i in d:
        key, value = i.split(':')
        key = key.replace(' ', '')
        key = key.replace('@', '')
        key = key.replace('\"', '')
        key = key.replace('\'', '')
        value = value.replace('#', '')
        value = value.replace('(', '')
        value = value.replace(')', '')
        value = value.replace(' ', '')
        value = value.replace('@', '')
        value = value.replace('\"', '')
        value = value.replace('\'', '')
        ans[key] = value
        # ans.append((key, value))
    return ans
```

Задание №8. [(`geen_431`, [9449, 598, -6994]),

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде списка пар.

Пример 1

Входная строка:

```
<section> .begin data #( 9449 , 598,-6994) ==>q(geen_431) .end. .begin data #( -5959, -8674, 3216 )==>q(onvela) .end. .begin data #( -2884, -6734 , -2601 ,-7938 ) ==> q(raer_345).end. </section>
```

Разобранный результат:

```
[('geen_431', [9449, 598, -6994]),
 ('onvela', [-5959, -8674, 3216]),
 ('raer_345', [-2884, -6734, -2601, -7938])]
```

Решение.

```
import re


def main(text):
    value_pattern = r"#\((.*?)\)"
    key_pattern = r"q\((.*?)\)"

    value_matches = re.findall(value_pattern, text, re.DOTALL)
    values = [[int(i) for i in match.split(',')] for match in value_matches]

    key_matches = re.findall(key_pattern, text)
    keys = [match for match in key_matches]

    result = list(zip(keys, values))
    return result
```

Задание №8. [(‘ates_350`, [`beti`, `qulete`, `lequed_850`]),

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде списка пар.

Пример 1

Входная строка:

```
.do <: variable ates_350 <- list(beti ;qulete ; lequed_850 ); :> <: variable bitere_771 <- list( maedza_317; biti_236); :> .end
```

Разобранный результат:

```
[('ates_350', ['beti', 'qulete', 'lequed_850']),
 ('bitere_771', ['maedza_317', 'biti_236'])]
```

Пример 2

Входная строка:

```
.do<:variable xeandi_495<- list( esgean_193 ; teabe_276 ; edza_308 ; xexeti ); :><: variable vece_990 <- list( orer_773 ; inoran ; ceor ; soge ); :> .end
```

Разобранный результат:

```
[('xeandi_495', ['esgean_193', 'teabe_276', 'edza_308', 'xexeti']),
 ('vece_990', ['orer_773', 'inoran', 'ceor', 'soge'])]
```

Решение.

```
def main(x):
    x = x.replace('.do', '')\
        .replace('.end', '')\
        .replace('<:', '')\
        .replace('variable', '')\
        .replace('list', '')\
        .replace('(', '')\
        .replace(')', '')\
        .replace('\n', '')\
        .replace(' ', '')
    x_parts = x.split(';:>')
    x_parts.pop(-1)
    x_parts1 = [i.split('<-') for i in x_parts]
    result = []
    for i in x_parts1:
        result.append((i[0], i[1].split(';')))
    return result
```

Задание №8. {`sote_540`: `xebeor`,

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде словаря.

Пример 1

Входная строка:

```
( var sote_540 ::=xebeor; ),( var uszace_14 ::= marion; ), ( var erte
::=eson_46; ), ( var rizaan ::= diqu; ),
```

Разобранный результат:

```
{'sote_540': 'xebeor',
 'uszace_14': 'marion',
 'erte': 'eson_46',
 'rizaan': 'diqu'}
```

Пример 2

Входная строка:

```
(var beer_827 ::= isso_180; ), ( var tion_500 ::= reti;), ( var
biinat ::=inxera_931; ), ( var ina_819 ::=mare;),
```

Разобранный результат:

```
{'beer_827': 'isso_180',
 'tion_500': 'reti',
 'biinat': 'inxera_931',
 'ina_819': 'mare'}
```

Решение.

```
def main(x):
    x = x.replace('(', '')\
        .replace(')', '')\
        .replace('var', '')\
        .replace('\n', '')\
        .replace(' ', '')\
        .replace('|', '')
    x_parts = x.split(';,')
    x_parts.pop(-1)
    x_parts1 = [i.split('::=') for i in x_parts]
    result = {}
    for i in x_parts1:
        result[i[0]] = i[1]
    return result
```

Задание №8. {`edla`: -9836, `dite`:1612, `orbees`: 5664, `uson_927`: 6333}

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде словаря.

Пример 1

Входная строка:

```
{<data> variable edla<= -9836;</data>,<data> variable dite<= 1612;
</data>, <data> variable orbees <= 5664; </data>, <data> variable
uson_927 <= 6333;</data>,}
```

Разобранный результат:

```
{'edla': -9836, 'dite': 1612, 'orbees': 5664, 'uson_927': 6333}
```

Пример 2

Входная строка:

```
{ <data> variable dicebe_689 <= 6871; </data>, <data> variable
isxe_552 <= 6759;</data>,}
```

Разобранный результат:

```
{'dicebe_689': 6871, 'isxe_552': 6759}
```

Решение.

```
import re


def main(data_string):
    p = r'<data>\s*variable\s+(?P<var>\w+)\s*<=\s*(?P<val>-?\d+)\s*;\s*</data>'
    matches = re.findall(p, data_string)
    result = {var: int(val) for var, val in matches}
    return result
```

Задание №8. { `raed`: 801, `beceve`: -7172, `geesma`: 452}

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде словаря.

Пример 1

Входная строка:

```
|| do option 8001=> raed; od; do option -7172 => beceve; od;do option
452=>geesma; od; ||
```

Разобранный результат:

```
{'raed': 8001, 'beceve': -7172, 'geesma': 452}
```

Пример 2

Входная строка:

```
||do option -2096 => geriis_736; od;do option 8785 => zaes; od; do
option 3345 => aused;od; ||
```

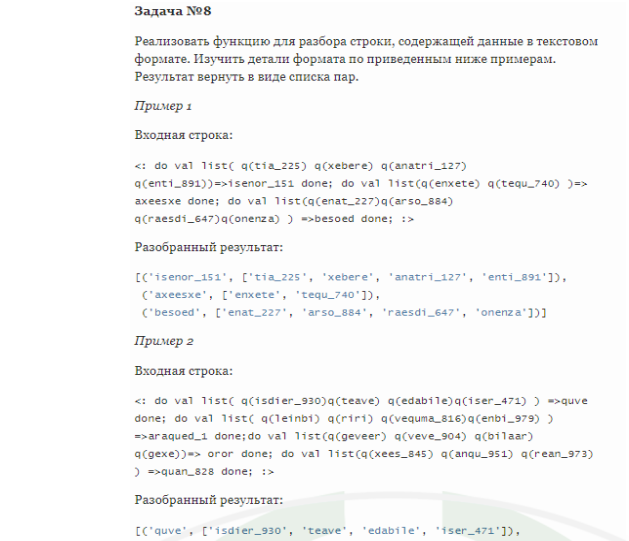
Разобранный результат:

```
{'geriis_736': -2096, 'zaes': 8785, 'aused': 3345}
```

Решение.

```
import re


def main(input_string):
    pattern = r'do\s*option\s*(-?\d+)\s*=>\s*([\w_]+);\s*od;'
    matches = re.findall(pattern, input_string)
    result = {key: int(value) for value, key in matches}
    return result
```

Задание №8. [(`isenor_151`, [`tia_225`, `xebere`, `anatri_127`, `enti_891`]),



Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде списка пар.

Пример 1

Входная строка:

```
<: do val list( q(tia_225) q(xebere) q(anatri_127)
q(enti_891))=>isenor_151 done; do val list(q(enxete) q(tequ_740) )=>
axeesxe done; do val list(q(enat_227)q(arso_884)
q(raesdi_647)q(onenza) ) =>besoed done; :>
```

Разобранный результат:

```
[('isenor_151', ['tia_225', 'xebere', 'anatri_127', 'enti_891']),
 ('axeesxe', ['enxete', 'tequ_740']),
 ('besoed', ['enat_227', 'arso_884', 'raesdi_647', 'onenza'])]
```

Пример 2

Входная строка:

```
<: do val list( q(isdier_930)q(teave) q(edabile)q(iser_471) ) =>quve
done; do val list( q(leinbi) q(riri) q(vequma_816)q(enbi_979) )
=>araqued_1 done;do val list(q(geveer) q(veve_904) q(bilaar)
q(gexe))=> oror done; do val list(q(xees_845) q(anqu_951) q(rean_973)
) =>quan_828 done; :>
```

Разобранный результат:

```
[('quve', ['isdier_930', 'teave', 'edabile', 'iser_471']),
```

Решение.

```
import re


def main(input_string):
    result = []
    input_string = input_string.replace("\n", " ")
    pattern = r"do val list\((.*?)\)\s*=>\s?(.*?)\s*done"

    matches = re.findall(pattern, input_string, re.DOTALL)
    for match in matches:

        value = match[1]
        items = re.findall(r"q\((.*?)\)", match[0])
        result.append((value, items))
    return result
```

Задание №8. {`zabidi_86`: `qulaer_887`, `maxe`: `bile`, `sotiis`: `esedle_506`}

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде словаря.

Пример 1

Входная строка:

```
(( <block> define"zabidi_86" :"qulaer_887"; </block>, <block> define
"maxe": "bile";</block>,<block> define "sotiis": "esedle_506";
</block>,))
```

Разобранный результат:

```
{'zabidi_86': 'qulaer_887', 'maxe': 'bile', 'sotiis': 'esedle_506'}
```

Пример 2

Входная строка:

```
((<block>define "enaor_319":"onbile_123"; </block>,<block> define
"edlexe_394" :"ati"; </block>,<block> define"onared" :
"onan";</block>, ))
```

Разобранный результат:

```
{'enaor_319': 'onbile_123', 'edlexe_394': 'ati', 'onared': 'onan'}
```

Решение.

```
import re


def main(s):
    ms = re.findall(r'"(.*?)"\s?:\s?"(.*?)"', s, re.DOTALL)
    d = {}
    for m in ms:
        d[m[0]] = m[1]
    return d
```

Задание №8. [(`erra`, -4815),

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде списка пар.

Пример 1

Входная строка:

```
<block>||auto -4815 to erra || ||auto 1668 to isteza_411 |||| auto
4772 to geis_881 || || auto 308 to soes_69 ||</block>
```

Разобранный результат:

```
[('erra', -4815),
 ('isteza_411', 1668),
 ('geis_881', 4772),
 ('soes_69', 308)]
```

Пример 2

Входная строка:

```
<block> || auto 8639 to enri_384 || || auto 2652 to geleor||</block>
```

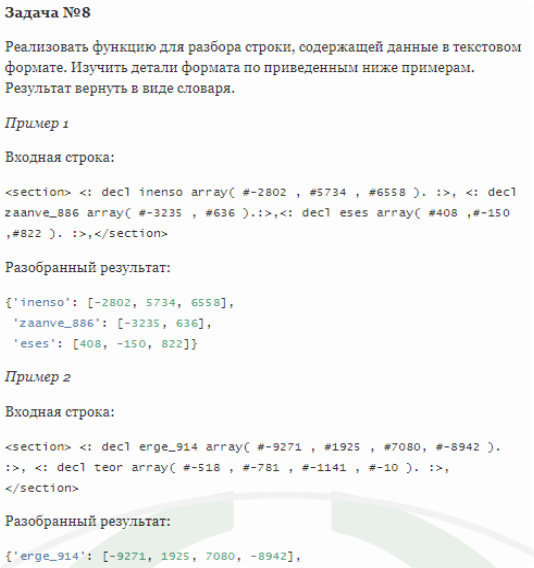
Разобранный результат:

```
[('enri_384', 8639), ('geleor', 2652)]
```

Решение.

```
def main(x):
    x = x.replace('<block>', '')\
        .replace('', '')\
        .replace('</block>', '')\
        .replace('\n', '')\
        .replace(' ', '')\
        .replace('|', '')
    x_parts = x.split('auto')
    x_parts.pop(0)
    x_parts1 = [i.split('to') for i in x_parts]
    result = []
    for i in x_parts1:
        result.append((i[1], int(i[0])))
    return result
```

Задание №8. {`inenso`: [-2802, 5734, 6558],



Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде словаря.

Пример 1

Входная строка:

```
<section> <: decl inenso array( #-2802 , #5734 , #6558 ). :>, <: decl
zaanve_886 array( #-3235 , #636 ).:>,<: decl eses array( #408 ,#-150
,#822 ). :>,</section>
```

Разобранный результат:

```
{'inenso': [-2802, 5734, 6558],
 'zaanve_886': [-3235, 636],
 'eses': [408, -150, 822]}
```

Пример 2

Входная строка:

```
<section> <: decl erge_914 array( #-9271 , #1925 , #7080, #-8942 ).
:>, <: decl teor array( #-518 , #-781 , #-1141 , #-10 ). :>,
</section>
```

Разобранный результат:

```
{'erge_914': [-9271, 1925, 7080, -8942],
```

Решение.

```
def main(input_string):
    result = {}
    # input_string = input_string.replace("\n", " ")
    # print(input_string)
    pattern = r'<:\s*decl\s*(\w+)\s*array\(([-\d\s,#]*)\)'

    matches = re.findall(pattern, input_string)
    for match in matches:
        key = match[0]
        values = [int(num.replace("#", '')) for num in match[1].split(',')]
        result[key] = values

    return result
```

Задание №8. [(`erceon_474`, 507),

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде списка пар.

Пример 1

Входная строка:

```
\begin .begin local #507==>erceon_474 .end .begin local #-7728 ==>
lain_767 .end .begin local #1748==> diza_84 .end .begin local #-8989
==> veama .end \end
```

Разобранный результат:

```
[('erceon_474', 507),
 ('lain_767', -7728),
 ('diza_84', 1748),
 ('veama', -8989)]
```

Пример 2

Входная строка:

```
\begin .begin local #4171 ==> quer_220 .end .begin local #9939
==>laaadi_681 .end .begin local #8886 ==> anis .end \end
```

Разобранный результат:

```
[('quer_220', 4171), ('laaadi_681', 9939), ('anis', 8886)]
```

Решение.

```
def main(x):
    x = x.replace('\\', '') \
        .replace('end', '') \
        .replace('.begin', '') \
        .replace('begin', '') \
        .replace('local', '') \
        .replace('=', '') \
        .replace('#', '') \
        .replace('\n', '') \
        .replace(' ', '')
    x_parts = x.split('.')
    x_parts.pop(-1)
    x_parts1 = [i.split('>') for i in x_parts]
    result = []
    for i in x_parts1:
        result.append((i[1], int(i[0])))
    return result
```

Задание №8. [(`biedan`, [`rasola`, `diza`, `abibele_416`, `isri_540`]),

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде списка пар.

Пример 1

Входная строка:

```
<: .begin set list( q(rasola); q(diza); q(abibele_416) ; q(isri_540)
) |> `biedan .end,.begin set list( q(antien_188) ; q(leen_691) ;
q(anqu);q(mazama) )|> `lear_217 .end,.begin set list( q(zave);
q(anes_387);q(riteza) ; q(usqu_55) ) |>`bianar_126 .end, :>
```

Разобранный результат:

```
[('biedan', ['rasola', 'diza', 'abibele_416', 'isri_540']),
 ('lear_217', ['antien_188', 'leen_691', 'anqu', 'mazama']),
 ('bianar_126', ['zave', 'anes_387', 'riteza', 'usqu_55'])]
```

Пример 2

Входная строка:

```
<: .begin set list(q(edra_629) ; q(endite_532); q(laesza) ;
q(aquat))|> `leenza .end, .begin set list( q(gela_821);q(enis) ;
q(rausis_956) ) |> `cete .end, .begin set list( q(qugube_328) ;
q(leinar) ; q(riores_532)) |> `direus_143 .end, .begin set list(
q(raxebi_964) ; q(arriti) ; q(maat) ) |>`argear_953 .end, :>
```

Разобранный результат:

```
[('leenza', ['edra_629', 'endite_532', 'laesza', 'aquat']),
 ('cete', ['gela_821', 'enis', 'rausis_956']),
 ('direus_143', ['qugube_328', 'leinar', 'riores_532']),
 ('argear_953', ['raxebi_964', 'arriti', 'maat'])]
```

Решение.

```
import re


def main(data_string):
    block_re = re.compile(
        r'\.begin set list\((.*?)\)\s*\|\>\s*(.*?)\s*\.end',
        re.DOTALL)
    blocks = block_re.findall(data_string.replace('\n', ' '))
    results = []
    for block in blocks:
        name = block[1]
        namet = name[1:]
        items_str = block[0]
        items = re.findall(r'q\((.*?)\)', items_str)
        results.append((namet, items))
    return results
```

Задание №8. [(`zain`, 2043), (`cequ_753`, 8887), (`xeus`, -8142)]

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде списка пар.

Пример 1

Входная строка:

```
((.do declare #2043 =>@"zain"; .end. .do declare #8887
=>@"cequ_753";.end. .do declare #-8142=> @"xeus";.end.))
```

Разобранный результат:

```
[('zain', 2043), ('cequ_753', 8887), ('xeus', -8142)]
```

Пример 2

Входная строка:

```
((.do declare#3259=> @"inriis_943"; .end. .do declare #-5838
=>@"maatce"; .end. .do declare #4300 => @"riti_314"; .end. ))
```

Разобранный результат:

```
[('inriis_943', 3259), ('maatce', -5838), ('riti_314', 4300)]
```

Решение.

```
import re

def main(input_string):
    matches = re.findall(r'#(-?\d+)\s*=>\s*@"([^"]+)"', input_string)
    return [(name, int(number)) for number, name in matches]
```

Задание №8. {`edveed_51`: [1056, -8630, -1197],

Задача №8

Реализовать функцию для разбора строки, содержащей данные в текстовом формате. Изучить детали формата по приведенным ниже примерам. Результат вернуть в виде словаря.

Пример 1

Входная строка:

```
do\begin local edveed_51 <==#( 1056 ; -8630 ; -1197 ).\end\begin
local leor_465 <==#(9425; 2792 ; -8292 ;7240 ). \end \begin local
erinso<==#(-7479; 3729;-2923). \end done
```

Разобранный результат:

```
{'edveed_51': [1056, -8630, -1197],
 'leor_465': [9425, 2792, -8292, 7240],
 'erinso': [-7479, 3729, -2923]}
```

Пример 2

Входная строка:

```
do\begin local sobi_801<== #( 5942 ; 671 ). \end \begin local raar
<== #(6701 ; 867 ; -582 ).\end done
```

Разобранный результат:

```
{'sobi_801': [5942, 671], 'raar': [6701, 867, -582]}
```

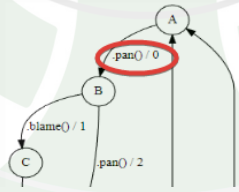
Решение.

```
def main(s):
    ms = re.findall(r'local\s?(.*?)\s?<==\s?#\((.*?)\)', s, re.DOTALL)
    d = {}
    for m in ms:
        r = m[1].split(';')
        d[m[0].strip()] = [int(i.strip()) for i in r]
    return d
```

ЗАДАНИЕ №10.

Для того, чтобы найти номер, пишите первое слово со стрелочкой от A к B.

Пример:



pan

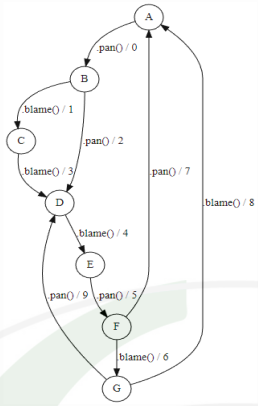
Задание №10. pan

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции test автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    pass


class MealyAutomaton:
    def __init__(self):
        self.state = 'A'

    def pan(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        elif self.state == 'B':
            self.state = 'D'
            return 2
        elif self.state == 'E':
            self.state = 'F'
            return 5
        elif self.state == 'F':
            self.state = 'A'
            return 7
        elif self.state == 'G':
            self.state = 'D'
            return 9
        else:
            raise MealyError("pan")

    def blame(self):
        if self.state == 'B':
            self.state = 'C'
            return 1
        elif self.state == 'C':
            self.state = 'D'
            return 3
        elif self.state == 'D':
            self.state = 'E'
            return 4
        elif self.state == 'F':
            self.state = 'G'
            return 6
        elif self.state == "G":
            self.state = 'A'
            return 8
        else:
            raise MealyError("blame")
```

```
def main():
    return MealyAutomaton()


def raises(func, error):
    output = None
    try:
        output = func()
    except Exception as e:
        assert type(e) == error
    assert output is None


def test():
    o = main()
    raises(o.blame, MealyError)
    assert o.pan() == 0
    assert o.blame() == 1
    raises(o.pan, MealyError)
    assert o.blame() == 3
    assert o.blame() == 4
    assert o.pan() == 5
    assert o.blame() == 6
    assert o.blame() == 8
    assert o.pan() == 0
    assert o.pan() == 2
    assert o.blame() == 4
    assert o.pan() == 5
    assert o.pan() == 7
    o = main()
    assert o.pan() == 0
    assert o.pan() == 2
    assert o.blame() == 4
    assert o.pan() == 5
    assert o.blame() == 6
    assert o.pan() == 9


test()
```

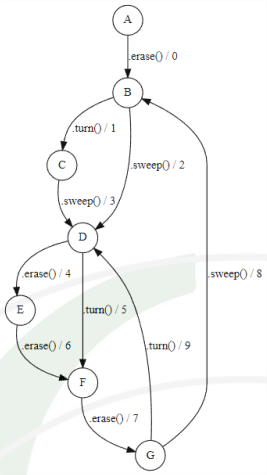
Задание №10. erase

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    pass

class Mealy(object):
    def __init__(self):
        self.condition = 'A'

    def erase(self):
        if self.condition == 'A':
            self.condition = 'B'
            return 0
        elif self.condition == 'D':
            self.condition = 'E'
            return 4
        elif self.condition == 'E':
            self.condition = 'F'
            return 6
        elif self.condition == 'F':
            self.condition = 'G'
            return 7
        else:
            raise MealyError("erase")

    def turn(self):
        if self.condition == 'B':
            self.condition = 'C'
            return 1
        elif self.condition == 'D':
            self.condition = 'F'
            return 5
        elif self.condition == 'G':
            self.condition = 'D'
            return 9
        else:
            raise MealyError("turn")

    def sweep(self):
        if self.condition == 'B':
            self.condition = 'D'
            return 2
        elif self.condition == 'G':
            self.condition = 'B'
            return 8
        elif self.condition == 'C':
            self.condition = 'D'
            return 3
        else:
            raise MealyError("sweep")
```

```
    assert o.erase() == 7
    assert o.turn() == 9
    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
    o = main()
    assert o.erase() == 0
    assert o.sweep() == 2
    o = main()
    assert o.erase() == 0
    assert o.sweep() == 2
    assert o.turn() == 5
    o = main()
    assert o.erase() == 0
    assert o.sweep() == 2
    assert o.erase() == 4
    assert o.erase() == 6
    o = main()
    assert o.erase() == 0
    assert o.sweep() == 2
    assert o.erase() == 4
    assert o.erase() == 6
    assert o.erase() == 7
    o = main()
    assert o.erase() == 0
    assert o.sweep() == 2
    assert o.erase() == 4
    assert o.erase() == 6
    assert o.erase() == 7
    assert o.turn() == 9
    o = main()
    assert o.erase() == 0
    assert o.sweep() == 2
    assert o.erase() == 4
    assert o.erase() == 6
    assert o.erase() == 7
    assert o.sweep() == 8
    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
    assert o.erase() == 4
    assert o.erase() == 6
    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
```

```
def main():
    return Mealy()

def raises(method, error):
    output = None
    try:
        output = method()
    except Exception as e:
        assert type(e) == error
    assert output is None

def test():
    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
    assert o.erase() == 4
    assert o.erase() == 6
    assert o.erase() == 7
    assert o.sweep() == 8

    o = main()
    assert o.erase() == 0
    # assert o.turn() == 1
    assert o.sweep() == 2
    assert o.turn() == 5
    assert o.erase() == 7
    assert o.turn() == 9

    o = main()
    assert o.erase() == 0
    # assert o.() == 1
    assert o.sweep() == 2
    assert o.turn() == 5
    assert o.erase() == 7
    assert o.sweep() == 8

    o = main()
    assert o.erase() == 0
    assert o.sweep() == 2
    assert o.turn() == 5
    assert o.erase() == 7
    assert o.sweep() == 8
    o = main()
    assert o.erase() == 0
    assert o.sweep() == 2
    assert o.erase() == 4
    assert o.erase() == 6
    assert o.erase() == 7
    assert o.sweep() == 8
    o = main()
    assert o.erase() == 0
    assert o.sweep() == 2
    assert o.turn() == 5
    assert o.erase() == 7
    assert o.turn() == 9
    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
    assert o.turn() == 5
    assert o.erase() == 7
    assert o.turn() == 9
    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
    assert o.erase() == 4
    assert o.erase() == 6
    assert o.erase() == 7
    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
    assert o.erase() == 4
    assert o.erase() == 6
    assert o.erase() == 7
    assert o.turn() == 9
    o = main()
    assert o.erase() == 0
    assert o.sweep() == 2
    assert o.turn() == 5
    assert o.erase() == 7
    assert o.sweep() == 8
    assert o.turn() == 1
    assert o.sweep() == 3
    assert o.erase() == 4
    assert o.erase() == 6
    assert o.erase() == 7
    assert o.turn() == 9
    assert o.turn() == 5
    assert o.erase() == 7

    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
    assert o.turn() == 5
    assert o.erase() == 7
    assert o.turn() == 9
    assert o.turn() == 5
    assert o.erase() == 7
    assert o.sweep() == 8
    assert o.sweep() == 2
    assert o.erase() == 4
    assert o.erase() == 6
    assert o.erase() == 7

    o = main()
    assert o.erase() == 0
    raises(lambda: o.erase(), MealyError)

    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
    assert o.erase() == 4
    assert o.erase() == 6
    assert o.erase() == 7
    assert o.turn() == 9
    raises(lambda: o.sweep(), MealyError)
    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
    raises(lambda: o.sweep(), MealyError)

    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
    assert o.turn() == 5
    assert o.erase() == 7
    assert o.turn() == 9
    assert o.turn() == 5
    assert o.erase() == 7
    assert o.sweep() == 8
    assert o.sweep() == 2
    assert o.erase() == 4
    assert o.erase() == 6
    assert o.erase() == 7
```

```
    raises(lambda: o.erase(), MealyError)
    o = main()
    assert o.erase() == 0
    assert o.turn() == 1
    assert o.sweep() == 3
    assert o.erase() == 4
    assert o.erase() == 6
    raises(lambda: o.turn(), MealyError)
```

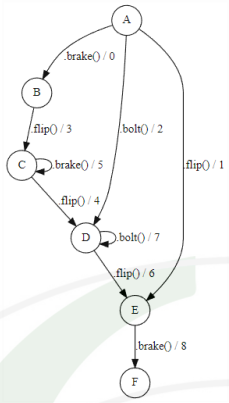
Задание №10. brake

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    def __init__(self, method_name):
        self.method_name = method_name

class StateMachine:
    state = 'A'

    def brake(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        elif self.state == 'C':
            return 5
        elif self.state == 'E':
            self.state = 'F'
            return 8
        else:
            raise (MealyError('brake'))

    def flip(self):
        if self.state == 'A':
            self.state = 'E'
            return 1
        elif self.state == 'B':
            self.state = 'C'
            return 3
        elif self.state == 'C':
            self.state = 'D'
            return 4
        elif self.state == 'D':
            self.state = 'E'
            return 6
        else:
            raise (MealyError('flip'))

    def bolt(self):
        if self.state == 'A':
            self.state = 'D'
            return 2
        elif self.state == 'D':
            return 7
        else:
            raise (MealyError('bolt'))
    def main():
        return StateMachine()
def test():
    test_exceptions = [
        ('B', 'brake'),
        ('B', 'bolt'),
        ('C', 'bolt'),
        ('D', 'brake'),
        ('E', 'bolt'),
        ('E', 'flip'),
        ('F', 'brake'),
        ('F', 'bolt'),
        ('F', 'flip'),
    ]
    test_result = [
        ('A', 'brake', 0, 'B'),
        ('A', 'flip', 1, 'E'),
        ('A', 'bolt', 2, 'D'),
        ('B', 'flip', 3, 'C'),
        ('C', 'flip', 4, 'D'),
        ('C', 'brake', 5, 'C'),
        ('D', 'flip', 6, 'E'),
        ('D', 'bolt', 7, 'D'),
        ('E', 'brake', 8, 'F'),
    ]
    sm = main()
    for start_state, action in test_exceptions:
        sm.state = start_state
        try:
            if action == 'flip':
                sm.flip()
            elif action == 'brake':
                sm.brake()
            else:
                sm.bolt()
        except MealyError:
            pass
    for start_state, action, expected_return,
expected_state in test_result:
        sm.state = start_state
        if action == 'flip':
            assert sm.flip() == expected_return
            assert sm.state == expected_state
        elif action == 'brake':
            assert sm.brake() == expected_return
            assert sm.state == expected_state
        else:
            assert sm.bolt() == expected_return
            assert sm.state == expected_state
```

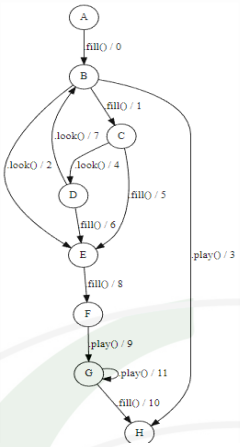
Задание №10. fill

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    pass


class MealyAutomaton:
    def __init__(self):
        self.state = 'A'

    def fill(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        elif self.state == 'B':
            self.state = 'C'
            return 1
        elif self.state == 'C':
            self.state = 'E'
            return 5
        elif self.state == 'D':
            self.state = 'E'
            return 6
        elif self.state == 'E':
            self.state = 'F'
            return 8
        elif self.state == 'G':
            self.state = 'H'
            return 10
        else:
            raise MealyError("fill")

    def look(self):
        if self.state == 'B':
            self.state = 'E'
            return 2
        elif self.state == 'C':
            self.state = 'D'
            return 4
        elif self.state == 'D':
            self.state = 'B'
            return 7
        else:
            raise MealyError("look")

    def play(self):
        if self.state == 'B':
            self.state = 'H'
            return 3
        elif self.state == 'F':
            self.state = 'G'
            return 9
        elif self.state == 'G':
            return 11
        else:
            raise MealyError("play")


def main():
    return MealyAutomaton()


def raises(func, error):
    output = None
    try:
        output = func()
    except Exception as e:
        assert type(e) == error
    assert output is None


def test():
    o = main()
    assert o.fill() == 0
    assert o.fill() == 1
    assert o.look() == 4
    assert o.look() == 7
    assert o.look() == 2
    assert o.fill() == 8
    assert o.play() == 9
    assert o.play() == 11
    assert o.fill() == 10
    raises(lambda: o.fill(), MealyError)
    raises(lambda: o.play(), MealyError)
```

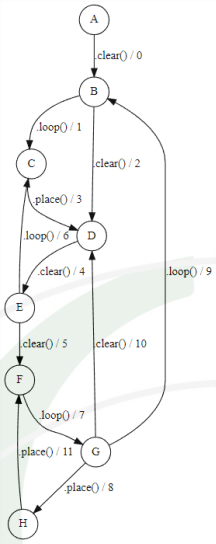
Задание №10. clear

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    pass


class StateMachine:
    def __init__(self):
        self.state = 'A'

    def clear(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        elif self.state == 'B':
            self.state = 'D'
            return 2
        if self.state == 'D':
            self.state = 'E'
            return 4
        if self.state == 'E':
            self.state = 'F'
            return 5
        if self.state == 'G':
            self.state = 'D'
            return 10
        raise MealyError('clear')

    def loop(self):
        if self.state == 'B':
            self.state = 'C'
            return 1
        if self.state == 'E':
            self.state = 'C'
            return 6
        if self.state == 'F':
            self.state = 'G'
            return 7
        if self.state == 'G':
            self.state = 'B'
            return 9
        raise MealyError('loop')

    def place(self):
        if self.state == 'C':
            self.state = 'D'
            return 3
        if self.state == 'G':
            self.state = 'H'
            return 8
        if self.state == 'H':
            self.state = 'F'
            return 11
        raise MealyError('place')


def main():
    return StateMachine()


def raises(func, error):
    output = None
    try:
        output = func()
    except Exception as e:
        assert type(e) == error
    assert output is None


def test():
    o = main()
    assert o.clear() == 0
    assert o.loop() == 1
    assert o.place() == 3
    assert o.clear() == 4
    assert o.clear() == 5
    assert o.loop() == 7
    assert o.place() == 8
    assert o.place() == 11
    o = main()
    assert o.clear() == 0
    assert o.clear() == 2
    assert o.clear() == 4
    assert o.loop() == 6
    assert o.place() == 3
    assert o.clear() == 4
    assert o.clear() == 5
    assert o.loop() == 7
    assert o.loop() == 9
```

```
o = main()
assert o.clear() == 0
assert o.clear() == 2
assert o.clear() == 4
assert o.clear() == 5
assert o.loop() == 7
assert o.clear() == 10
raises(lambda: o.place(), MealyError)
assert o.clear() == 4
assert o.loop() == 6
raises(lambda: o.loop(), MealyError)
raises(lambda: o.clear(), MealyError)
```

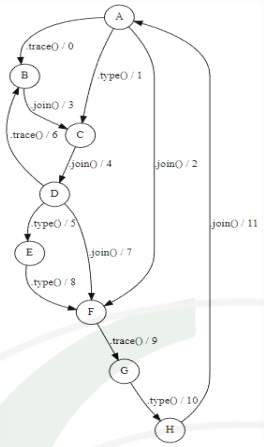
Задание №10. trace

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    pass


class MealyAutomaton:
    def __init__(self):
        self.state = 'A'

    def trace(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        elif self.state == 'D':
            self.state = 'B'
            return 6
        elif self.state == 'F':
            self.state = 'G'
            return 9
        else:
            raise MealyError("trace")

    def type(self):
        if self.state == 'A':
            self.state = 'C'
            return 1
        elif self.state == 'D':
            self.state = 'E'
            return 5
        elif self.state == 'E':
            self.state = 'F'
            return 8
        elif self.state == 'G':
            self.state = 'H'
            return 10
        else:
            raise MealyError("type")

    def join(self):
        if self.state == 'A':
            self.state = 'F'
            return 2
        elif self.state == 'B':
            self.state = 'C'
            return 3
        elif self.state == 'C':
            self.state = 'D'
            return 4
        elif self.state == 'D':
            self.state = 'F'
            return 7
        elif self.state == 'H':
            self.state = 'A'
            return 11
        else:
            raise MealyError("join")


def main():
    return MealyAutomaton()


def raises(func, error):
    output = None
    try:
        output = func()
    except Exception as e:
        assert type(e) == error
    assert output is None


def test():
    o = main()
    assert o.join() == 2
    assert o.trace() == 9
    assert o.type() == 10
    assert o.join() == 11
    assert o.type() == 1
    assert o.join() == 4
    assert o.trace() == 6
    assert o.join() == 3
    raises(lambda: o.type(), MealyError)
    assert o.join() == 4
    assert o.type() == 5
    assert o.type() == 8
    assert o.trace() == 9
    assert o.type() == 10
    assert o.join() == 11
    assert o.trace() == 0
    assert o.join() == 3

    o = main()
    assert o.trace() == 0
    assert o.join() == 3
    assert o.join() == 4
    assert o.join() == 7
    assert o.trace() == 9
    assert o.type() == 10
    assert o.join() == 11
    assert o.join() == 2
    assert o.trace() == 9
```

```
    assert o.type() == 10
    raises(lambda: o.trace(), MealyError)
    assert o.join() == 11
    assert o.type() == 1
    assert o.join() == 4
    assert o.type() == 5
    assert o.type() == 8
    assert o.trace() == 9
    raises(lambda: o.join(), MealyError)
```

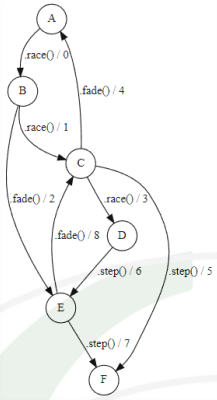
Задание №10. race

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    pass


class StateMachine:
    def __init__(self):
        self.state = 'A'

    def race(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        if self.state == 'B':
            self.state = 'C'
            return 1
        if self.state == 'C':
            self.state = 'D'
            return 3
        raise MealyError('race')

    def fade(self):
        if self.state == 'B':
            self.state = 'E'
            return 2
        if self.state == 'C':
            self.state = 'A'
            return 4
        if self.state == 'E':
            self.state = 'C'
            return 8
        raise MealyError('fade')

    def step(self):
        if self.state == 'C':
            self.state = 'F'
            return 5
        if self.state == 'D':
            self.state = 'E'
            return 6
        if self.state == 'E':
            self.state = 'F'
            return 7
        raise MealyError('step')


def main():
    return StateMachine()
```

```
def raises(func, error):
    output = None
    try:
        output = func()
    except Exception as e:
        assert type(e) == error
    assert output is None


def test():
    o = main()
    assert o.race() == 0
    assert o.race() == 1
    assert o.fade() == 4
    assert o.race() == 0
    assert o.race() == 1
    assert o.race() == 3
    assert o.step() == 6
    assert o.fade() == 8
    assert o.step() == 5
    o = main()
    assert o.race() == 0
    assert o.fade() == 2
    assert o.step() == 7
    raises(lambda: o.step(), MealyError)
    raises(lambda: o.fade(), MealyError)
    raises(lambda: o.race(), MealyError)
```

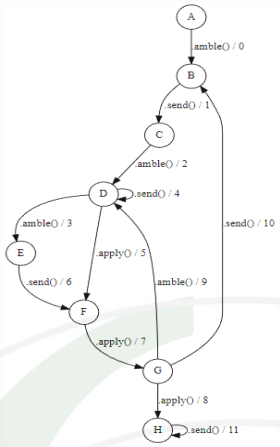
Задание №10. amble

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
from struct import unpack_from, calcsize
from pprint import pprint

FMT = dict(
    char='c',
    int8='b',
    uint8='B',
    int16='h',
    uint16='H',
    int32='i',
    uint32='I',
    int64='q',
    uint64='Q',
    float='f',
    double='d'
)


def parse(buf, offs, ty, order='<'):
    pattern = FMT[ty]
    size = calcsize(pattern)
    value = unpack_from(order + pattern, buf,
offs)[0]
    return value, offs + size


def parse_d(buf, offs):
    d1, offs = parse(buf, offs, 'uint32')
    d2, offs = parse(buf, offs, 'int32')

    d3 = []
    for _ in range(4):
        val, offs = parse(buf, offs, 'int16')
        d3.append(val)

    d4, offs = parse(buf, offs, 'int8')
    d5, offs = parse(buf, offs, 'uint8')
    d6, offs = parse(buf, offs, 'uint64')
    return dict(D1=d1, D2=d2, D3=d3, D4=d4,
D5=d5, D6=d6), offs


def parse_c(buf, offs):
    c1, offs = parse(buf, offs, 'int64')
    c2, offs = parse(buf, offs, 'uint32')
    c3, offs = parse(buf, offs, 'int8')
    return dict(C1=c1, C2=c2, C3=c3), offs
```

```
def parse_b(buf, offs):
    b1 = []
    for _ in range(6):
        c_offs, offs = parse(buf, offs,
'uint32')
        val, _ = parse_c(buf, c_offs)
        b1.append(val)

    b2 = []
    for _ in range(6):
        val, offs = parse(buf, offs, 'char')
        b2.append(val)
    b2 = b''.join(b2).decode('utf-8')

    b3 = []
    for _ in range(3):
        val, offs = parse(buf, offs, 'double')
        b3.append(val)

    return dict(B1=b1, B2=b2, B3=b3), offs


def parse_a(buf, offs):
    b_offs, offs = parse(buf, offs, "uint16")
    a1, _ = parse_b(buf, b_offs)
    a2, offs = parse(buf, offs, 'uint32')
    d_offs, offs = parse(buf, offs, "uint16")
    a3, _ = parse_d(buf, d_offs)
    return dict(A1=a1, A2=a2, A3=a3), offs


def main(stream):
    return parse_a(stream, 4)[0]
```

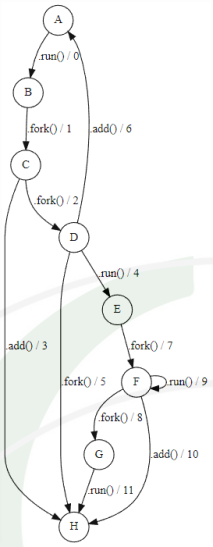
Задание №10. run

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    def __init__(self, method_name):
        self.method_name = method_name


class StateMachine:
    state = 'A'

    def run(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        elif self.state == 'D':
            self.state = 'E'
            return 4
        elif self.state == 'F':
            return 9
        elif self.state == 'G':
            self.state = 'H'
            return 11
        else:
            raise (MealyError('run'))

    def fork(self):
        if self.state == 'B':
            self.state = 'C'
            return 1
        elif self.state == 'C':
            self.state = 'D'
            return 2
        elif self.state == 'D':
            self.state = 'H'
            return 5
        elif self.state == 'E':
            self.state = 'F'
            return 7
        elif self.state == 'F':
            self.state = 'G'
            return 8
        else:
            raise (MealyError('fork'))
```

```
    def add(self):
        if self.state == 'D':
            self.state = 'A'
            return 6
        elif self.state == 'C':
            self.state = 'H'
            return 3
        elif self.state == 'F':
            self.state = 'H'
            return 10
        else:
            raise (MealyError('add'))


def main():
    return StateMachine()


def test():
    test_exceptions = [
        ('A', 'fork'),
        ('A', 'add'),
        ('B', 'add'),
        ('B', 'run'),
        ('C', 'run'),
        ('E', 'run'),
        ('E', 'add'),
        ('G', 'add'),
        ('G', 'fork'),
        ('H', 'add'),
        ('H', 'fork'),
        ('H', 'run'),
    ]
    test_result = [
        ('A', 'run', 0, 'B'),
        ('B', 'fork', 1, 'C'),
        ('C', 'fork', 2, 'D'),
        ('C', 'add', 3, 'H'),
        ('D', 'run', 4, 'E'),
        ('D', 'fork', 5, 'H'),
        ('D', 'add', 6, 'A'),
        ('E', 'fork', 7, 'F'),
        ('F', 'fork', 8, 'G'),
        ('F', 'run', 9, 'F'),
        ('F', 'add', 10, 'H'),
```

```
        ('G', 'run', 11, 'H'),
    ]
    sm = main()
    for start_state, action in test_exceptions:
        sm.state = start_state
        try:
            if action == 'fork':
                sm.fork()
            elif action == 'add':
                sm.add()
            else:
                sm.run()
        except MealyError:
            pass
    for start_state, action, expected_return,
expected_state in test_result:
        sm.state = start_state
        if action == 'fork':
            assert sm.fork() == expected_return
            assert sm.state == expected_state
        elif action == 'add':
            assert sm.add() == expected_return
            assert sm.state == expected_state
        else:
            assert sm.run() == expected_return
            assert sm.state == expected_state
```

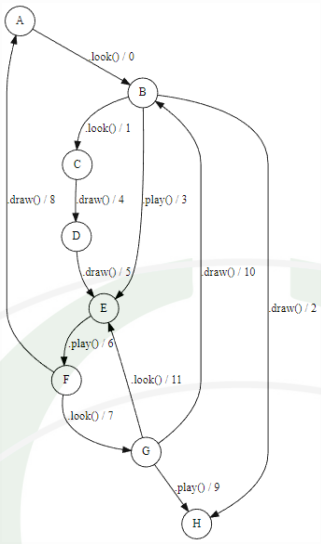
Задание №10. look

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
from struct import unpack_from, calcsize

FMT = dict(
    char='c',
    int8='b',
    uint8='B',
    int16='h',
    uint16='H',
    int32='i',
    uint32='I',
    int64='q',
    uint64='Q',
    float='f',
    double='d'
)


def parse(buf, offs, ty, order='>'):
    pattern = FMT[ty]
    size = calcsize(pattern)
    value = unpack_from(order + pattern, buf,
offs)[0]
    return value, offs + size


def parse_d(buf, offs):
    d1_size, offs = parse(buf, offs, 'uint16')
    d1_offset, offs = parse(buf, offs,
'uint32')
    d1 = []

    for _ in range(d1_size):
        e1, d1_offset = parse(buf, d1_offset,
'int8')
        e2, d1_offset = parse(buf, d1_offset,
'uint64')
        e3_size, d1_offset = parse(buf,
d1_offset, 'uint32')
        e3_offset, d1_offset = parse(buf,
d1_offset, 'uint16')
        e3 = []
        for __ in range(e3_size):
            val, e3_offset = parse(buf,
e3_offset, 'uint8')
            e3.append(val)

        d1.append(dict(E1=e1, E2=e2, E3=e3))

    d2 = []
    for _ in range(2):
        val, offs = parse(buf, offs, 'uint64')
        d2.append(val)

    return dict(D1=d1, D2=d2), offs


def parse_c(buf, offs):
    c1, offs = parse(buf, offs, 'uint32')
    c2, offs = parse(buf, offs, 'int32')
    return dict(C1=c1, C2=c2), offs


def parse_b(buf, offs):
    b1_offset, offs = parse(buf, offs,
'uint32')
    b1, _ = parse_c(buf, b1_offset)
    b2, offs = parse(buf, offs, 'int32')
    b3, offs = parse(buf, offs, 'int32')
    b4, offs = parse(buf, offs, 'uint64')
    b5, offs = parse(buf, offs, 'uint64')
    b6_offset, offs = parse(buf, offs,
'uint32')
    b6, _ = parse_d(buf, b6_offset)

    return dict(B1=b1, B2=b2, B3=b3, B4=b4,
B5=b5, B6=b6, ), offs


def parse_a(buf, offs):
    a1, offs = parse(buf, offs, 'double')
    a2, offs = parse_b(buf, offs)

    a3_size, offs = parse(buf, offs, 'uint32')
    a3_offset, offs = parse(buf, offs,
'uint32')
    a3 = []
```

```
    for _ in range(a3_size):
        val, a3_offset = parse(buf, a3_offset,
'uint16')
        a3.append(val)
    return dict(A1=a1, A2=a2, A3=a3), offs


def main(stream):
    return parse_a(stream, 5)[0]
```

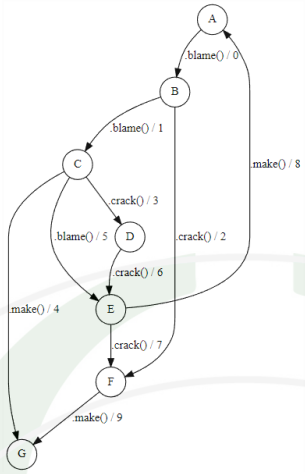
Задание №10. blame

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    def __init__(self, method_name):
        self.method_name = method_name


class StateMachine:
    state = 'A'

    def blame(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        elif self.state == 'B':
            self.state = 'C'
            return 1
        elif self.state == 'C':
            self.state = 'E'
            return 5
        else:
            raise (MealyError('blame'))

    def crack(self):
        if self.state == 'B':
            self.state = 'F'
            return 2
        elif self.state == 'C':
            self.state = 'D'
            return 3
        elif self.state == 'D':
            self.state = 'E'
            return 6
        elif self.state == 'E':
            self.state = 'F'
            return 7
        else:
            raise (MealyError('crack'))

    def make(self):
        if self.state == 'C':
            self.state = 'G'
            return 4
        elif self.state == 'E':
            self.state = 'A'
            return 8
        elif self.state == 'F':
            self.state = 'G'
            return 9
        else:
            raise (MealyError('make'))


def main():
    return StateMachine()


def test():
    test_exceptions = [
        ('A', 'make'),
        ('A', 'crack'),
        ('B', 'make'),
        ('D', 'make'),
        ('D', 'blame'),
        ('E', 'blame'),
        ('F', 'crack'),
        ('F', 'blame'),
        ('G', 'make'),
        ('G', 'blame'),
        ('G', 'crack'),
    ]
    test_result = [
        ('A', 'blame', 0, 'B'),
        ('B', 'blame', 1, 'C'),
        ('B', 'crack', 2, 'F'),
        ('C', 'crack', 3, 'D'),
        ('C', 'make', 4, 'G'),
        ('C', 'blame', 5, 'E'),
        ('D', 'crack', 6, 'E'),
        ('E', 'crack', 7, 'F'),
        ('E', 'make', 8, 'A'),
        ('F', 'make', 9, 'G'),
    ]
    sm = main()
    for start_state, action in test_exceptions:
        sm.state = start_state
        try:
```

```
            if action == 'make':
                sm.make()
            elif action == 'crack':
                sm.crack()
            else:
                sm.blame()
        except MealyError:
            pass
    for start_state, action, expected_return,
expected_state in test_result:
        sm.state = start_state
        if action == 'make':
            assert sm.make() == expected_return
            assert sm.state == expected_state
        elif action == 'crack':
            assert sm.crack() == expected_return
            assert sm.state == expected_state
        else:
            assert sm.blame() == expected_return
            assert sm.state == expected_state
```

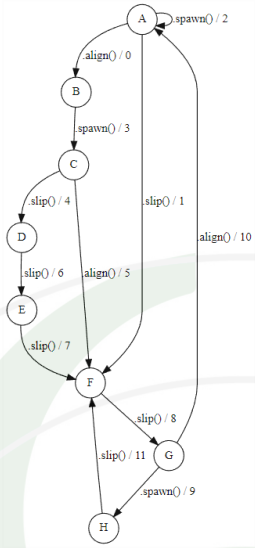
Задание №10. align

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    def __init__(self, method_name):
        self.method_name = method_name


class StateMachine:
    state = 'A'

    def slip(self):
        if self.state == 'A':
            self.state = 'F'
            return 1
        elif self.state == 'C':
            self.state = 'D'
            return 4
        elif self.state == 'D':
            self.state = 'E'
            return 6
        elif self.state == 'E':
            self.state = 'F'
            return 7
        elif self.state == 'F':
            self.state = 'G'
            return 8
        elif self.state == 'H':
            self.state = 'F'
            return 11
        else:
            raise (MealyError('slip'))

    def spawn(self):
        if self.state == 'A':
            return 2
        elif self.state == 'B':
            self.state = 'C'
            return 3
        elif self.state == 'G':
            self.state = 'H'
            return 9
        else:
            raise (MealyError('spawn'))

    def align(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        elif self.state == 'C':
            self.state = 'F'
            return 5
        elif self.state == 'G':
            self.state = 'A'
            return 10
        else:
            raise (MealyError('align'))


def main():
    return StateMachine()


def test():
    test_exceptions = [
        ('B', 'align'),
        ('B', 'slip'),
        ('C', 'spawn'),
        ('D', 'align'),
        ('D', 'spawn'),
        ('E', 'align'),
        ('E', 'spawn'),
        ('F', 'align'),
        ('F', 'spawn'),
        ('H', 'align'),
        ('H', 'spawn'),
        ('G', 'spawn')
    ]
    test_result = [
        ('A', 'align', 0, 'B'),
        ('A', 'slip', 1, 'F'),
        ('A', 'spawn', 2, 'A'),
        ('B', 'spawn', 3, 'C'),
        ('C', 'slip', 4, 'D'),
        ('C', 'align', 5, 'F'),
        ('D', 'slip', 6, 'E'),
        ('E', 'slip', 7, 'F'),
        ('F', 'slip', 8, 'G'),
        ('G', 'spawn', 9, 'H'),
        ('G', 'align', 10, 'A'),
```

```
        ('H', 'slip', 11, 'F')
    ]
    sm = main()
    for start_state, action in test_exceptions:
        sm.state = start_state
        try:
            if action == 'slip':
                sm.slip()
            elif action == 'spawn':
                sm.spawn()
            else:
                sm.align()
        except MealyError:
            pass
    for start_state, action, expected_return,
expected_state in test_result:
        sm.state = start_state
        if action == 'slip':
            assert sm.slip() == expected_return
            assert sm.state == expected_state
        elif action == 'spawn':
            assert sm.spawn() == expected_return
            assert sm.state == expected_state
        else:
            assert sm.align() == expected_return
            assert sm.state == expected_state
```

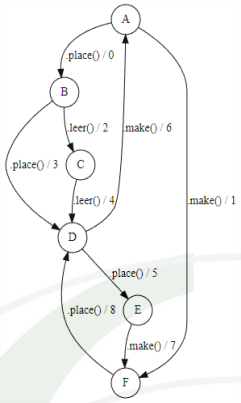
Задание №10. place

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции test автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    pass


class StateMachine:
    def __init__(self):
        self.state = 'A'

    def place(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        if self.state == 'B':
            self.state = 'D'
            return 3
        if self.state == 'F':
            self.state = 'D'
            return 8
        if self.state == 'D':
            self.state = 'E'
            return 5
        raise MealyError('place')

    def make(self):
        if self.state == 'A':
            self.state = 'F'
            return 1
        if self.state == 'D':
            self.state = 'A'
            return 6
        if self.state == 'E':
            self.state = 'F'
            return 7
        raise MealyError('make')

    def leer(self):
        if self.state == 'B':
            self.state = 'C'
            return 2
        if self.state == 'C':
            self.state = 'D'
            return 4
        raise MealyError('leer')
```

```
def main():
    return StateMachine()


def raises(func, error):
    output = None
    try:
        output = func()
    except Exception as e:
        assert type(e) == error
    assert output is None


def test():
    o = main()
    assert o.place() == 0
    assert o.place() == 3
    assert o.make() == 6
    assert o.make() == 1
    assert o.place() == 8
    assert o.place() == 5
    assert o.make() == 7
    o = main()
    assert o.place() == 0
    assert o.leer() == 2
    assert o.leer() == 4
    o.state = 'X'
    raises(lambda: o.place(), MealyError)
    raises(lambda: o.leer(), MealyError)
    raises(lambda: o.make(), MealyError)
```

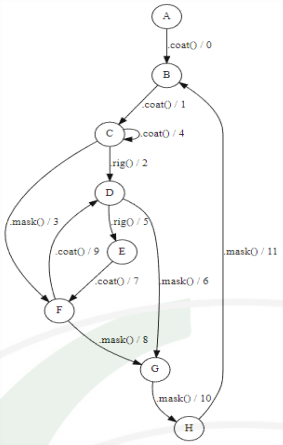
Задание №10. coat

Задача №10

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class StateMachine:
    def __init__(self):
        self.state = 'A'

    def coat(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        if self.state == 'B':
            self.state = 'C'
            return 1
        if self.state == 'C':
            self.state = 'C'
            return 4
        if self.state == 'E':
            self.state = 'F'
            return 7
        if self.state == 'F':
            self.state = 'D'
            return 9
        raise MealyError('coat')

    def rig(self):
        if self.state == 'C':
            self.state = 'D'
            return 2
        if self.state == 'D':
            self.state = 'E'
            return 5
        raise MealyError('rig')

    def mask(self):
        if self.state == 'C':
            self.state = 'F'
            return 3
        if self.state == 'F':
            self.state = 'G'
            return 8
        if self.state == 'D':
            self.state = 'G'
            return 6
        if self.state == 'G':
            self.state = 'H'
            return 10
        if self.state == 'H':
            self.state = 'B'
            return 11
        raise MealyError('mask')
```

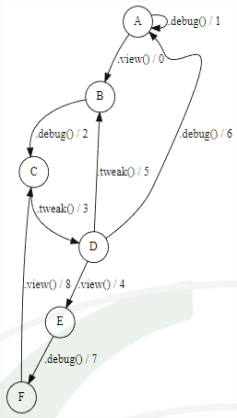
```
class MealyError(Exception):
    pass

def raises(method, error):
    output = None
    try:
        output = method()
    except Exception as e:
        assert type(e) == error
    assert output is None

def main():
    return StateMachine()

def test():
    sm = StateMachine()
    assert sm.state == 'A'
    assert sm.coat() == 0
    assert sm.state == 'B'
    assert sm.coat() == 1
    assert sm.state == 'C'
    assert sm.coat() == 4
    assert sm.state == 'C'
    sm.state = 'E'
    raises(lambda: sm.coat(), MealyError)
    assert sm.state == 'F'
    assert sm.coat() == 9
    assert sm.state == 'D'
    sm.state = 'D'
    raises(lambda: sm.coat(), MealyError)
    assert sm.state == 'C'
    assert sm.rig() == 2
    assert sm.state == 'D'
    assert sm.rig() == 5
    assert sm.state == 'E'
    sm.state = 'D'
    raises(lambda: sm.rig(), MealyError)
    assert sm.state == 'C'
    assert sm.mask() == 3
    assert sm.state == 'F'
    assert sm.mask() == 8
    assert sm.state == 'G'
    assert sm.mask() == 6
    assert sm.state == 'G'
    assert sm.mask() == 10
    assert sm.state == 'H'
    assert sm.mask() == 11
    assert sm.state == 'B'
    raises(lambda: sm.mask(), MealyError)
```

Задание №10. view

Реализовать конечный автомат Мили в виде класса. Начальным состоянием автомата является A. Методы возвращают числовые значения.

Если вызываемый метод не реализован для некоторого состояния, необходимо вызвать пользовательское исключение MealyError. При возникновении исключения должно передаваться имя метода, вызвавшего это исключение.

Реализовать в отдельной функции `test` автоматическое тестирование автомата Мили на основе покрытия ветвей. Требуемая степень покрытия: 100%.



Решение.

```
class MealyError(Exception):
    pass


class StateMachine:
    def __init__(self):
        self.state = 'A'

    def view(self):
        if self.state == 'A':
            self.state = 'B'
            return 0
        if self.state == 'D':
            self.state = 'E'
            return 4
        if self.state == 'F':
            self.state = 'C'
            return 8
        raise MealyError('view')

    def debug(self):
        if self.state == 'A':
            return 1
        if self.state == 'B':
            self.state = 'C'
            return 2
        if self.state == 'D':
            self.state = 'A'
            return 6
        if self.state == 'E':
            self.state = 'F'
            return 7
        raise MealyError('debug')

    def tweak(self):
        if self.state == 'C':
            self.state = 'D'
            return 3
        if self.state == 'D':
            self.state = 'B'
            return 5
        raise MealyError('tweak')


def main():
    return StateMachine()
```

```
def raises(func, error):
    output = None
    try:
        output = func()
    except Exception as e:
        assert type(e) == error
    assert output is None


def test():
    o = main()
    assert o.debug() == 1
    assert o.view() == 0
    assert o.debug() == 2
    assert o.tweak() == 3
    assert o.view() == 4
    assert o.debug() == 7
    assert o.view() == 8
    assert o.tweak() == 3
    assert o.tweak() == 5
    assert o.debug() == 2
    assert o.tweak() == 3
    assert o.debug() == 6
    assert o.view() == 0
    raises(lambda: o.tweak(), MealyError)
    raises(lambda: o.view(), MealyError)
    assert o.debug() == 2
    raises(lambda: o.debug(), MealyError)


test()
```