Домашнее задание

Обязательная часть:

1.

файл task1

2.

файл task2

3.

Любимый C++, поэтому придётся искать по питону

pass statement

делает ничего. Требуется, когда синтаксически требуется стейтмент, но никакого действия не требуется. Или просто оставить на месте, где в будущем подразумевается код.
https://docs.python.org/3/tutorial/controlflow.htmlpass-statements

Оказывается в питоне можно отдельно обозначать, откуда задаётся каждая переменная в функции. Не то чтобы обязательно, но полезно с точки зрения читаемости.
def f(pos1, pos2, /, pos-or-kwd, *, kwd1, kwd2):

https://docs.python.org/3/tutorial/controlflow.htmlspecial-parameters

Также я покапался в библиотеках и нашел там полезные вещи

Performance Measurement:
Чтобы узнавать время действия определённой части кода. Полезно для асимптотической оценки алгоритмов и определения эффективности на практике.
»> from timeit import Timer
»> Timer('t=a; a=b; b=t', 'a=1; b=2').timeit()
0.57535828626024577
»> Timer('a,b = b,a', 'a=1; b=2').timeit()
0.54962537085770791
https://docs.python.org/3/tutorial/stdlib.htmlperformance-measurement

String Pattern Matching:
Позволяет создавать регулярные выражения и проверять на них строки
»> import re
»> re.findall(r'/bf[a-z]*', 'which foot or hand fell fastest')
foot', 'fell', 'fastest'

»> re.sub(r'(/b[a-z]+) /1', r'/1', 'cat in the the hat')
'cat in the hat'
https://docs.python.org/3/tutorial/stdlib.htmlstring-pattern-matching

Quality Control:
Эту штуку я не до конца понял, но теперь знаю, что она существует. Это позволяет удобно строить тесты к своей программе и находить их, звучит полезно
def average(values):
"Computes the arithmetic mean of a list of numbers.

»> print(average([20, 30, 70]))
40.0
"
return sum(values) / len(values)

import doctest
doctest.testmod() automatically validate the embedded tests
https://docs.python.org/3/tutorial/stdlib.htmlquality-control

4.

Любой КДА является в глубине души ориентированным графом. В нём есть следующие сущности

Вершина-состояние, которое идентифицируется по натуральному числу и булевской переменной (0, 1). Переменная говорит - является ли оно терминальным.

Рёбра-переходы, которые задаются символом ->, принимающим описание двух вершин, которые он соединяет, а также часть алфавита ,которая соответствует этому направлению в скобках "[]".

Добавим несколько соглашений: Выделим как отдельную сущность Сток, это пусть будет 0 0. Начальная вершина отмечается натуральным числом 1

Также скажем, что пусть символы "(),"ничего не значят с точки зрения

языка, а существуют для удобства записи и читаемости.

время привести пару КДА в нашей записи.

возьмём 1 автомат из дз:

-> (1 0) (2 1) [0], -> (2 1) (0 0) [0..9],
-> (1 0) (3 0) [1..4, 6..9], -> (1 0) (4 1) [5],
-> (3 0) (4 1) [0 5], -> (4 1) (3 0) [1..4, 6..9],
-> (3 0) (3 0) [1..4, 6..9], -> (4 1) (4 1) [0 5]

Здесь символы ".."я не описал что значит, но здесь это подразумевает список элементов и добавлено чтобы глазам было не так больно.

Пусть 2 автомат описывает проверку числа кратности 3(example2):
-> (1 0) (2 1) [0], -> (2 1) (0 0) [0..9],
-> (1 0) (3 0) [3 6 9], -> (1 0) (4 0) [1 4 7], -> (1 0) (5 0) [2 5 8],
-> (3 1) (4 0) [1 4 7], -> (3 1) (5 0) [2 5 8], -> (3 1) (3 1) [0 3 6 9],
-> (4 0) (3 1) [2 5 8], -> (4 0) (5 0) [1 4 7], -> (4 0) (4 0) [0 3 6 9],
-> (5 0) (3 1) [1 4 7], -> (5 0) (4 0) [2 5 8], -> (5 0) (5 0) [0 3 6 9],

Автомат 3 описывает слова из языка племени ау. Слова племени ау состоят из двух букв а и у, и являются словами только если звуков а и звуков у чётное число(example3):
-> (1 1) (2 0) [a], -> (1 1) (3 0) [y]
-> (2 0) (1 1) [a], -> (2 0) (4 0) [y]
-> (3 0) (1 1) [y], -> (3 0) (4 0) [a]
-> (4 0) (2 0) [y], -> (4 0) (3 0) [a]