Аметов К.Л. ПИ-231(2)
Задание: Минимизировать ДКА Мили, заданный таблицей выходов и переходов. Построить эквивалентный автомат 2 рода. Провести проверку в JFLAP

№ 1

A \ Q	1	2	3	4	5	6	7	8	9
a	4, *x*	3, *y*	7, *x*	5, *x*	8, *x*	7, *x*	2, *y*	5, *x*	2, *y*
b	7, *y*	4, *y*	9, *x*	9, *y*	7, *y*	9, *y*	8, *x*	9, *y*	5, *x*

Построим множества π1 эквивалентных состояний: это состояния, не различимые никаким однобуквенным воздействием: π1={A,B,C,D} , Axy={1,4,5,6,8} Byx={9,7} Cxx={3}Dyy={2}. Индексами для удобства обозначены реакции на воздействия a,b соответственно.
Класс Axy был расщеплен, процедура была завершена за два шага.
В самом низу картинки представлен эквивалентынй гомоморфный нормализованный автомат мили с прежними обозначениями.

A\Q	1	2	3	4	5	6	7	8	9
a	4x	3y	7x	5x	8x	7x	2y	5x	2y
b	7y	4y	9x	9y	7y	9y	8x	9y	5x

	Axy					Byx		Cxx	Dyy
A\Q	1	4	5	6	8	9	7	3	2
a	A	A	A	B	A	D	D	B	C
b	B	B	B	B	B	A	A	B	A

	B0				B1		B2	B3	B4
A\Q	1	4	5	8	9	7	3	2	6
a	B0	B0	B0	B0	B3	B3	B1	B2	B1
b	B1	B1	B1	B1	B0	B0	B1	B0	B1

	B0	B1	B2	B3	B4
A\Q	1	9	3	2	6
a	B0	B3	B1	B2	B1
b	B1	B0	B1	B0	B1

A\Q	1	9	3	2	6
a	1/x	2/y	9x	3y	9x
b	9/y	1/x	9x	1y	9y

Убедимся на частном примере, что автоматные прелбразования двух машин одинаковы. Выберем в исходном автомате в качестве начального состояния 4 и подадим на его вход цепочку “abba”. Так же проверим нормализованный автомат.

Исходная машина				
Момент времени	t	T+1	T+2	T+3
Анализируемый символ	a	b	b	a
Реакция	x	y	x	x
состояние к след.момент	4	9	5	8

Нормализованная машина				
Момент времени	t	T+1	T+2	T+3
Анализируемый символ	a	b	b	a
Реакция	x	y	x	x
состояние к след.момент	1	9	1	1

Перейдем к автомату 2 рода(Мура), построю его и покажу в табличном виде

A\Q	1	9	3	2	6
a	1/x	2/y	9x	3y	9x
b	9/y	1/x	9x	1y	9y

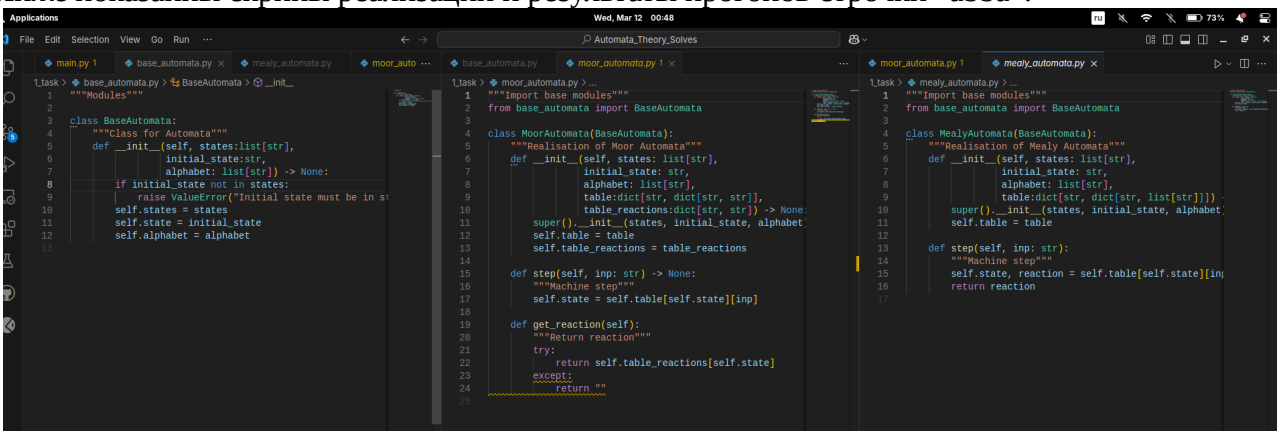
Состояния	Реак при дост	Реал
1	1/x	A0/x
	1/y	A1/y
9	9/y	A2/y
	9x	A3/x
3	3/y	A4/y
2	2/y	A5/y
6	-	A6/-

	A0	A1	A2	A3	A4	A5	A6
A/Q	x	y	y	x	y	y	-
a	A0	A0	A5	A5	A3	A4	A3
b	A2	A2	A0	A0	A3	A1	A2

Проверю автомат мура

Момент времени	t	T+1	T+2	T+3	T+4
Анализируемый символ	a	b	b	a	-
Реакция	x	x	y	x	x
состояние к след.моменту	A0	A2	A0	A0	-
состояние в тек момент	A0	A0	A2	A0	A0

P.S
Так же была выполненна программная реализация автомата Мура и Мили, построение эквивалентного автомату Мили автомата мура и минимизация автомата на ЯП Python.
Ниже показанны скрины реализации и результаты прогонов строчки “abba”.



Реализация построения эквивалентного автомату Мили, автомата Мура

1_task > transformation.py > mealy_to_moor

```python
"""Modules"""
from collections import defaultdict
from moor_automata import MoorAutomata
from mealy_automata import MealyAutomata

def mealy_to_moor(automata: MealyAutomata) -> MoorAutomata:
    """Transforms a Mealy automaton into a Moore automaton."""
    state_reactions = defaultdict(set)
    for state in automata.states:
        for symbol in automata.alphabet:
            next_state, reaction = automata.table[state][symbol]
            state_reactions[next_state].add(reaction)

    new_states = {}
    reaction_table = {}
    old_to_new_state = {}
    state_counter = 0

    for state in automata.states:
        if not state_reactions[state]:
            state_reactions[state].add("")  # Для состояния без реакций

        for reaction in state_reactions[state]:
            new_state = f"A{state_counter}"
            new_states[(state, reaction)] = new_state
            reaction_table[new_state] = reaction
            old_to_new_state[state] = new_state
            state_counter += 1

    transition_table = defaultdict(dict)
    for new_state, old_state in new_states.items():
        # print(1, new_state, old_state)
        for symbol in automata.alphabet:
            next_old_state, reaction = automata.table[new_state[0]][symbol]
            transition_table[old_state][symbol] = new_states[(next_old_state, reaction)]

    return MoorAutomata(
        states=transition_table.keys(),
        initial_state=old_to_new_state[automata.state],
        alphabet=automata.alphabet,
        table=transition_table,
        table_reactions=reaction_table
    )
```

Реализация минимизации

```python
from mealy_automata import MealyAutomata
from collections import defaultdict
from moor_automata import MoorAutomata
from transformation import mealy_to_moor

def minimization(automata: MealyAutomata):
    reactions = {}
    ind = 1
    table = automata.table
    old_states_to_new = {}
    for s, i in table.items():
        reaction = tuple(i[a][1] for a in automata.alphabet)
        if reaction not in reactions:
            reactions[reaction] = f"A{ind}"
            ind+=1
    for s, i in table.items():
        reaction = tuple(i[a][1] for a in automata.alphabet)
        old_states_to_new[s] = reactions[reaction]

    table1= defaultdict(dict)
    for s, i in table.items():
        for a in automata.alphabet:
            table1[s][a] = old_states_to_new[table[s][a][0]]

^
    table2 = defaultdict(dict)
    ind2 = 66
    while True:
        old_states_to_new1 = {}
        reactions1 ={}
        ind = 1
        for s, i in table1.items():
            reaction1 = tuple(j for _,j in i.items())
            if reaction1 not in reactions1:
                reactions1[reaction1] = f"{chr(ind2)}{ind}"
                ind+=1
        for s, i in table1.items():
            reaction1 = tuple(j for _,j in i.items())
            old_states_to_new1[s] = reactions1[reaction1]
        for s, i in table.items():
            for a in alphabet:
                table1[s][a] = old_states_to_new1[table[s][a][0]]
        ind2+=1
        if table1 == table2:
            break
        table2 = table1
    new_to_old = {j:i for i,j in old_states_to_new1.items()}
    table2= defaultdict(dict)
    for s, i in table.items():
        for a in alphabet:
            table2[old_states_to_new1[s]][a] = (table1[s][a], table[s][a][1])
    table3 =defaultdict(dict)
    for s,i in table2.items():
        for a in alphabet:
            table3[new_to_old[s]][a] = (new_to_old[table2[s][a][0]], table2[s][a][1])
    return table3
```

Прогон строчки “abba” на исходном, нормализованном и эквивалентном автомате второго рода.

```python
# Таблица переходов и реакций для автомата Мили
table = {
    "1": {"a": ["4", "x"], "b": ["7", "y"]},
    "2": {"a": ["3", "y"], "b": ["4", "y"]},
    "3": {"a": ["7", "x"], "b": ["9", "x"]},
    "4": {"a": ["5", "x"], "b": ["9", "y"]},
    "5": {"a": ["8", "x"], "b": ["7", "y"]},
    "6": {"a": ["7", "x"], "b": ["9", "y"]},
    "7": {"a": ["2", "y"], "b": ["8", "x"]},
    "8": {"a": ["5", "x"], "b": ["9", "y"]},
    "9": {"a": ["2", "y"], "b": ["5", "x"]}
}


# Список состояний
states = list(table.keys())

# Начальное состояние
initial_state = "8"

# Алфавит входных символов
alphabet = ["a", "b"]

automata = MealyAutomata(states, initial_state, alphabet, table)
print("Результат работы над строчкой 'abba' автомата Мили "+"".join([automata.step(i) for i in "abba"]))

automata.table = minimization(automata)
automata.states = automata.table.keys()
automata.state = "8"
print("Результат работы над строчкой 'abba' нормализованного автомата Мили "+"".join([automata.step(i) for i in "abba"]))

res = ""
moor = mealy_to_moor(automata)
for i in "abba":
    moor.step(i)
    res += moor.get_reaction()
print(f"Результат работы над строчкой 'abba' автомата Мура {res}")
```

```
(.venv) kemran@kemran:~/Automata_Theory_Solves$  cd /home/kemran/Automata_
ter/../../debugpy/launcher 39693 -- /home/kemran/Automata_Theory_Solves/1_
Результат работы над строчкой 'abba' автомата Мили хухх
Результат работы над строчкой 'abba' нормализованного автомата Мили хухх
Результат работы над строчкой 'abba' автомата Мура хухх
(.venv) kemran@kemran:~/Automata_Theory_Solves$
```

Вывод: выполнена минимизация автомата Мили, построен эквивалентный нормализованный автомат Мура; автоматные преобразования для тестовой четырех-символьной цепочки совпадают.