UTN
Facultad Regional
VILLA MARIA

INGENIERIA EN SISTEMAS DE INFORMACIÓN

SINTAXIS Y SEMANTICA DE LOS LENGUAJES

TRABAJO PRÁCTICO NUMERO 2

Profesor:

- **Ing. Mario Rinaldi.**

JTP:

- **Ing. Jorge Palombarini.**

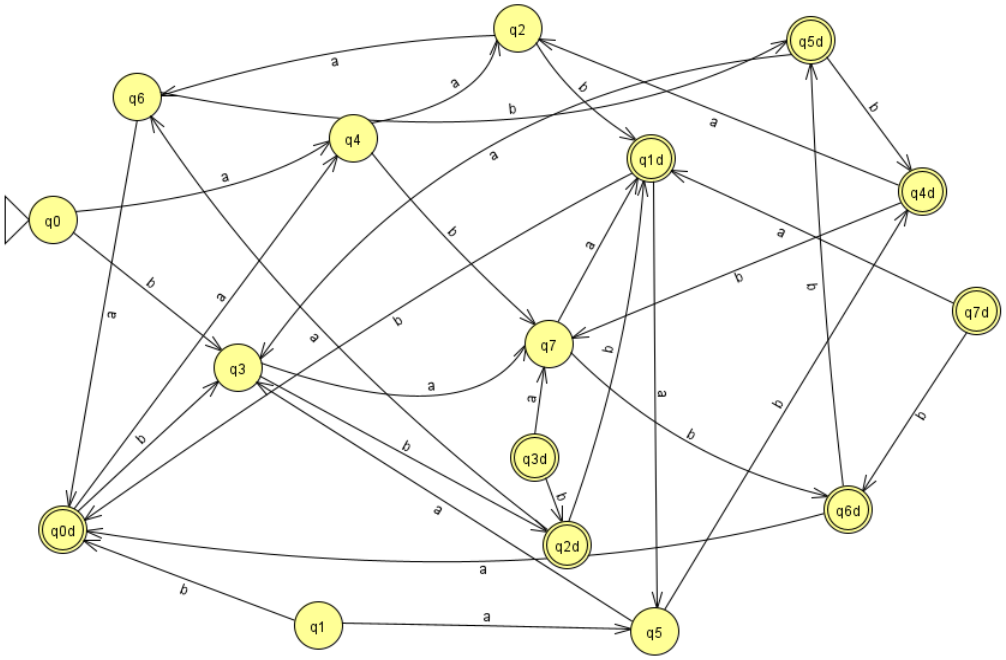
Integrantes:

- **Aimbinder Tiago Gabriel**
- **Flores Mauricio Fernando**
- **Nuñez Fabricio**
- **Tabilo Ivo Ezequiel**

#CÓDIGO
#IMPOSIBLE corregir la salida del código en python. Para la próxima acomodarlo para que sea legible, identificar a que ejercicio corresponde cada salida. (En ese sentido sirvió que hayan adjuntado las imagenes en el TP).
#FALTA la VERIFICACIÓN en la totalidad de los ejercicios.
#Correcto el planteo de las trazas aleatorias, y el autómata mínimo.

Ejercicio 1

a) A qué posición de niveladores corresponde cada estado?



CODIGO:

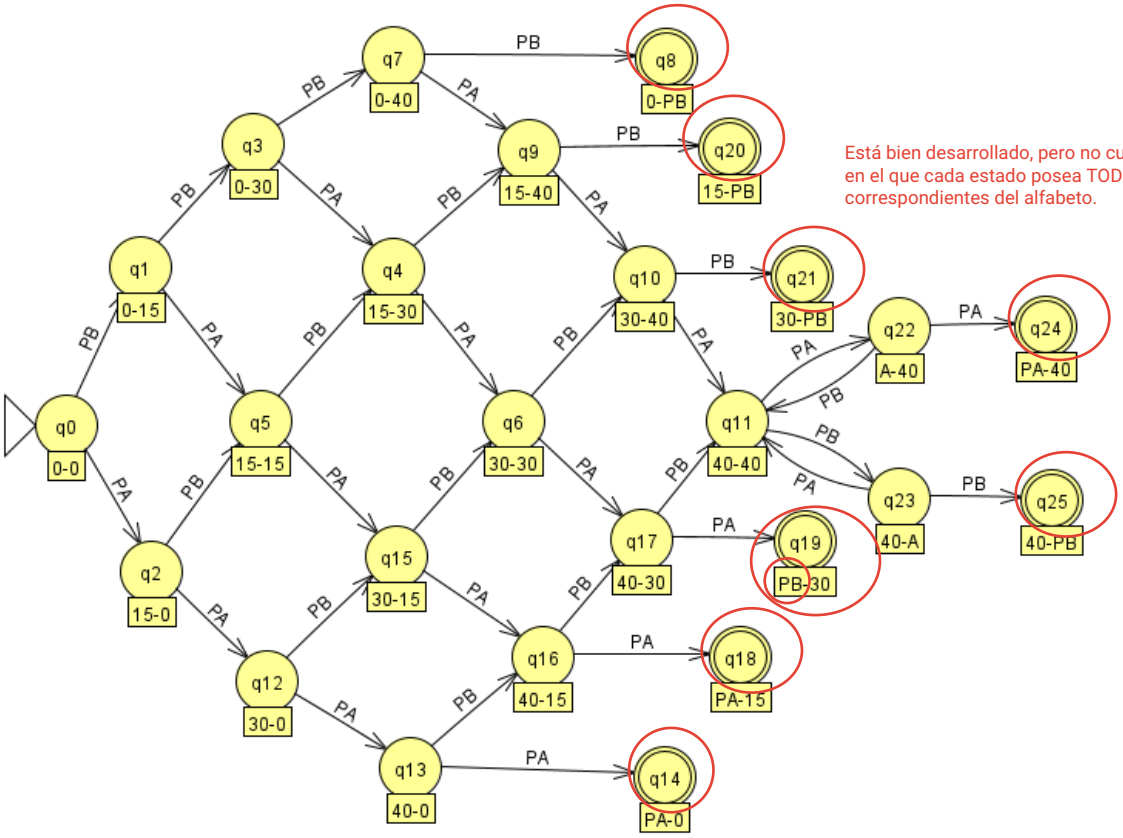
b)

```
import random
from automata.fa.dfa import DFA
# DFA which matches all binary strings ending in an odd number of '1's
dfa = DFA(
        states={'q0', 'q1', 'q2', 'q3', 'q4', 'q5', 'q6', 'q7', 'q0d', 'q1d', 'q2d', 'q3d', 'q4d',
                'q5d', 'q6d', 'q7d'},
        input_symbols={'a', 'b'},
        transitions={
            'q0': {'a': 'q4', 'b': 'q3'},
            'q1': {'a': 'q5', 'b': 'q0d'},
            'q2': {'a': 'q6', 'b': 'q1d'},
            'q3': {'a': 'q7', 'b': 'q2d'},
            'q4': {'a': 'q2', 'b': 'q7'},
            'q5': {'a': 'q3', 'b': 'q4d'},
            'q6': {'a': 'q0d', 'b': 'q5d'},
            'q7': {'a': 'q1d', 'b': 'q6d'},
            'q0d': {'a': 'q4', 'b': 'q3'},
            'q1d': {'a': 'q5', 'b': 'q0d'},
            'q2d': {'a': 'q6', 'b': 'q1d'},
            'q3d': {'a': 'q7', 'b': 'q2d'},
            'q4d': {'a': 'q2', 'b': 'q7'},
            'q5d': {'a': 'q3', 'b': 'q4d'},
            'q6d': {'a': 'q0d', 'b': 'q5d'},
            'q7d': {'a': 'q1d', 'b': 'q6d'},
        },
        initial_state='q0',
        final_states={'q0d', 'q1d', 'q2d', 'q3d',
```

```
                    'q4d', 'q5d', 'q6d', 'q7d'}
    )
symbols = ['a', 'b']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')
```

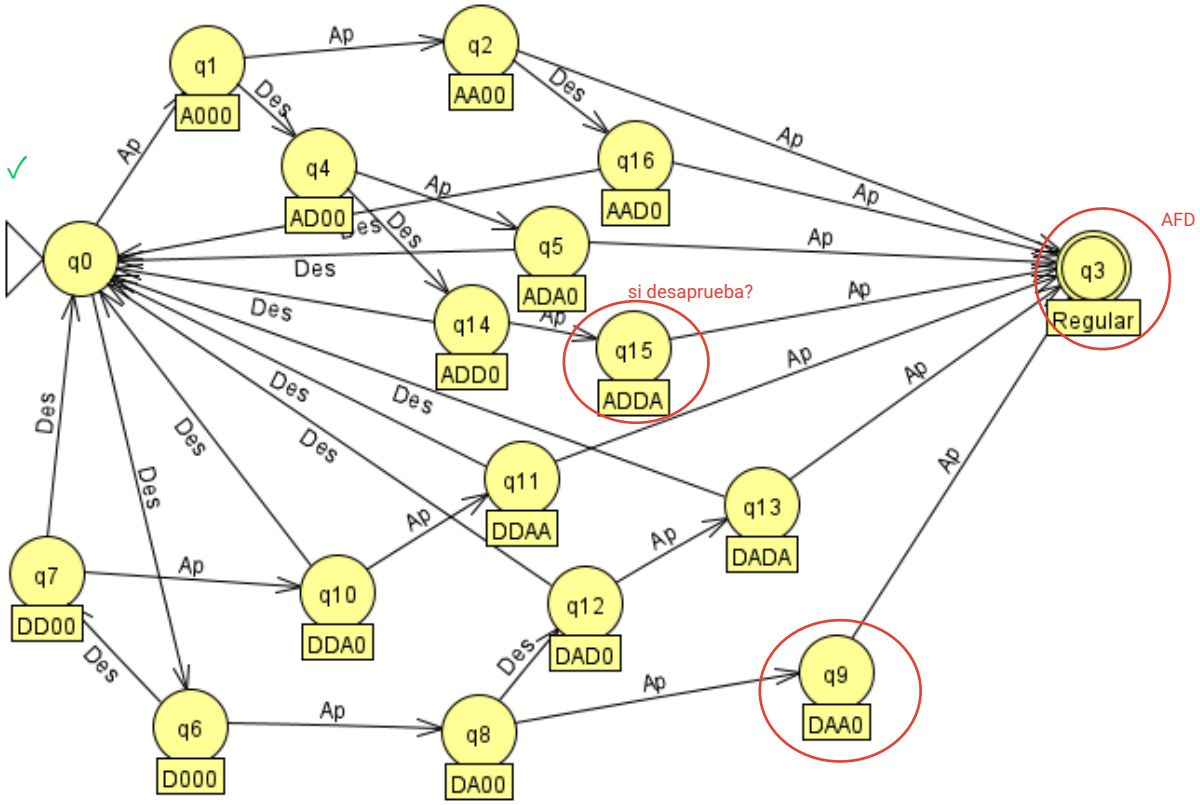
```
bbbaaa rejected
aaaabbbb accepted
aa rejected
abbab rejected
bab accepted
baba accepted
abaaa rejected
aa rejected
```

b)

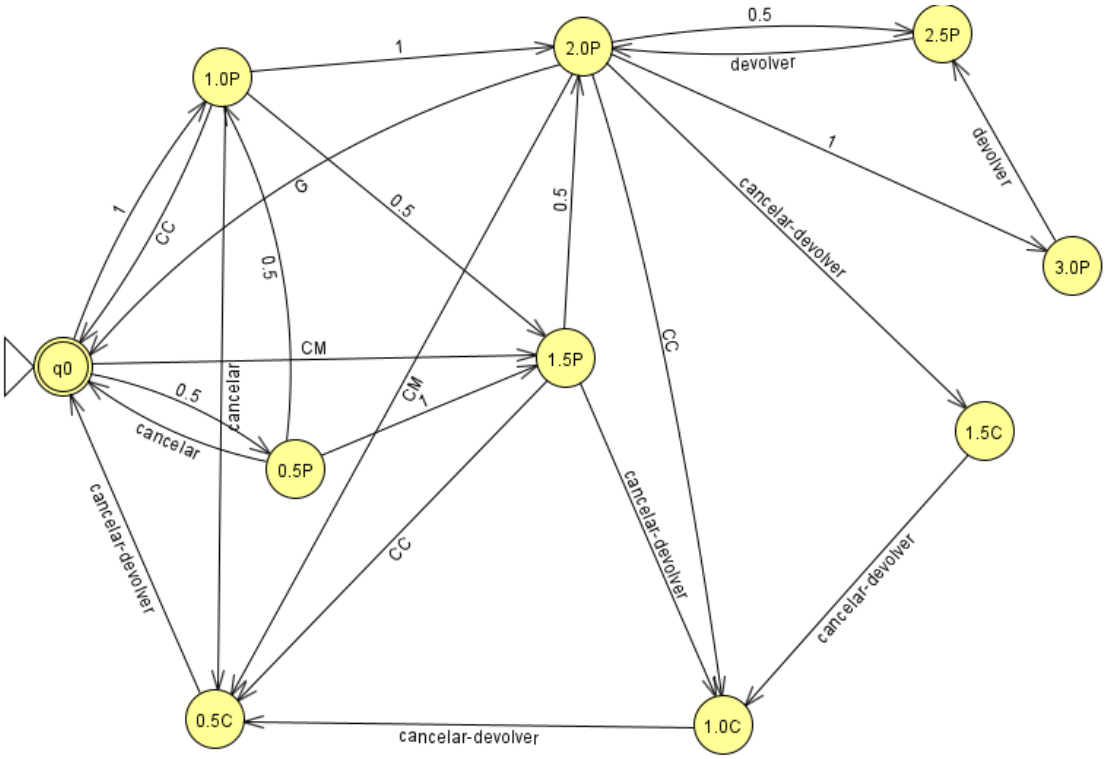


Está bien desarrollado, pero no cumple con la condición de AFD, en el que cada estado posea TODAS las transiciones correspondientes del alfabeto.

c)



d)



Aimbinder Tiago Gabriel, Flores Mauricio Fernando, Nuñez Fabricio, Tabilo Ivo Ezequiel.

Ejercicio 2

AUTOMATA 1

a)

Estado inicial= {q1} El estado inicial no está dentro de un conjunto, no lleva llaves.

Estado final= {q2} F es un CONJUNTO de estados finales (en este caso de un solo elemento. Si lleva llaves)

Transiciones =

input/Estado	0	1
q1	q1	q2
q2	q3	q2
q3	q2	q2

Estados= {q1, q2, q3}

ALFABETO
~~Lenguaje=~~ {0,1}

L = (W|W termina en 1 o en dos ceros consecutivos) X Para conjuntos deben usar llaves NO paréntesis, y lo que definieron toma inválidas como 00,000,etc

Ejemplos de string= {1, 11, 100, 1001, 01011]

b)

AUTOMATA 2

Estado inicial= S

Estado final= {q1, r1}

Transiciones =

input/Estado	0	1
S	q1	r1
q1	q1	q2
q2	q1	q2

r1	r2	r1
r2	r2	r1

Estados= {S, q1, q2, r1, r2} ✓

~~Lenguaje~~= {a, b}

L = (W|W empieza y termina con el mismo símbolo) ✓ Llaves, no paréntesis para conjuntos.

Ejemplo de string = {aaba, abbba, bb, baab, bababb} ✓

AUTOMATA 3

Estado inicial= ~~{~~q0~~, q1, q2}~~ X Estado inicial es uno solo, SIEMPRE, no es un conjunto.

Estado final {q0~~(reset)~~}

Transiciones= ✓

input/Estado	0	1	2
q0~~(reset)~~	q0	q1	q2
q1	q1	q2	q0
q2	q2	q0	q1

Estados= {q0, q1, q2} ✓

~~Lenguaje=~~ {0, 1, 2} Falta <reset> en el alfabeto

L = (W|W tal que la suma de los números sea múltiplo de 3) ✓

Ejemplo string= {21, 111, 2211, 1212, 222} ✓

AUTOMATA 4

Estado inicial= {q0}

Estado final ={q1}

Transiciones =

input/Estado	letra	digito
q0	q1	q0
q1	q1	q1

Estados= {q0, q1}

~~Lenguaje=~~ {letra, digito}

L = (W|W contiene al menos una “letra”)

Ejemplos de string = {letra digito letra, digito letra, letra, digito letra digito, letra digito letra letra}

AUTOMATA 5

Estado inicial= {q0} no lleva llaves

Estado final= {q2}

Transiciones=

input/Estado	a	b
q0	q1	q0
q1	q2	q1
q2	q3	q2
q3	q3	q3

Estados= {q0, q1, q2, q3}

~~Lenguaje=~~ {a, b}

L = (W|W contiene dos “a”)

Ejemplo string = {bbaba, aabb, ababb, aa, abbba}

AUTOMATA 6

Estado inicial= {q0}

Estado final= {q0}

Transiciones=

input/Estado	a	b
q0	q1	q2
q1	q2	q0
q2	q2	q2

Estados= {q0, q1, q2}

~~Lenguaje=~~ {a, b}

L = (W|W permite solo la cadena “ab” y que esta se repita cuantas veces quiera el usuario)

También permite la cadena vacía

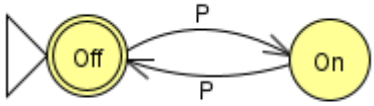
Ejemplo string = {ab, abab, ababab, abababab, ababababab}

Ejercicio 3

El programa lo que hace es mostrar como salida deseada una cadena ingresada cuya suma es 3 o un múltiplo de este (incluye al 0). Al llegar a la suma de 3, un múltiplo de este o simplemente 0 lo q hace el programa es volver al estado inicial que es también la salida deseada.

También se puede resetear

Ejercicio 4



Transiciones

~~input/~~Estado	P
Off	On
On	Off

Estado inicial = {Off}

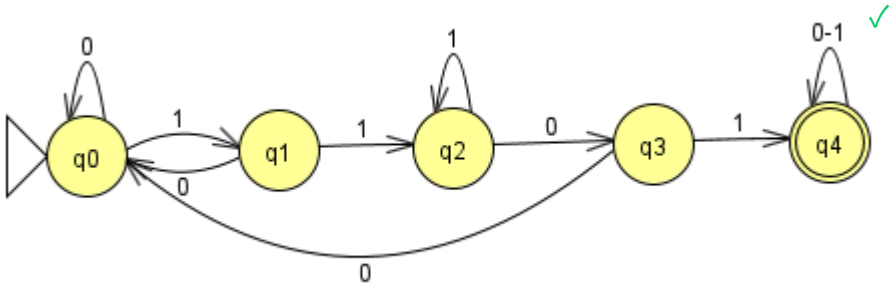
Estado final = {On, Off}

Estados= {Off, On}

~~Lenguaje=~~ {P}

Ejercicio 5

a)



Estado inicia l= {q0]

Estado final = {q4}

Estados = {q0, q1, q2, q3, q4}

~~Lenguaje~~ = {0,1}

Transiciones =

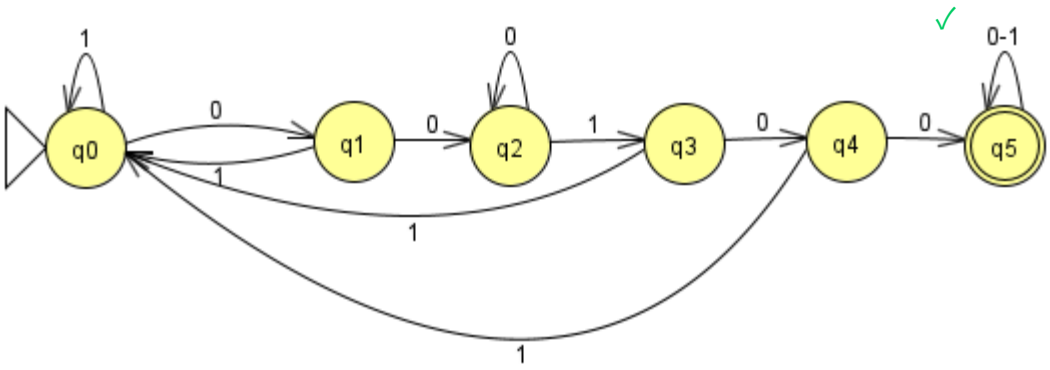
input/estado	0	1
q0	q0	q1
q1	q0	q2
q2	q3	q2
q3	q0	q4
q4	q4	q4

```python
import random
from automata.fa.dfa import DFA

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3', 'q4'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q0', '1': 'q1'},
        'q1': {'0': 'q0', '1': 'q2'},
        'q2': {'0': 'q3', '1': 'q2'},
        'q3': {'0': 'q0', '1': 'q4'},
        'q4': {'0': 'q4', '1': 'q4'}
    },
    initial_state='q0',
    final_states={'q4'}
)
```

```python
symbols = ['0', '1']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
01 rejected
01001 rejected
10010000 rejected
1001011 rejected
01111 rejected
11010 accepted
0010 rejected
00010101 rejected
Minimo
```

b)



Estado inicia l= {q0]

Estado final = {q5} ✓

Estados = {q0, q1, q2, q3, q4, q5} ✓

~~Lenguaje~~ = {0,1} ✓

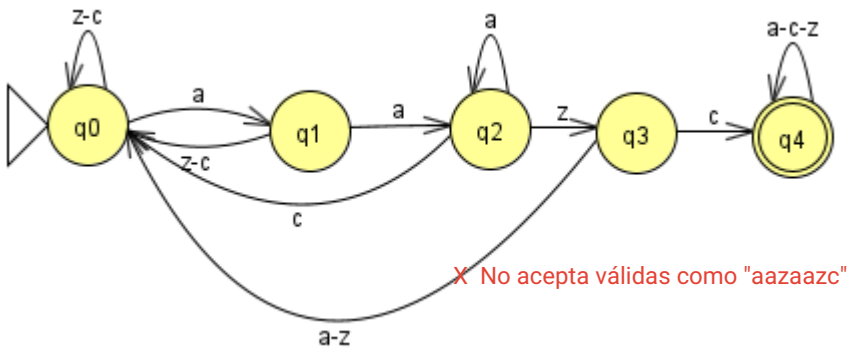
Transiciones = ✓

input/estado	0	1
q0	q1	q0
q1	q2	q0
q2	q2	q3
q3	q4	q0
q4	q5	q0
q5	q5	q5

```python
import random
from automata.fa.dfa import DFA

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3', 'q4', 'q5'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q1', '1': 'q0'},
        'q1': {'0': 'q2', '1': 'q0'},
        'q2': {'0': 'q2', '1': 'q3'},
        'q3': {'0': 'q4', '1': 'q0'},
        'q4': {'0': 'q5', '1': 'q0'},
        'q5': {'0': 'q5', '1': 'q5'}
    },
    initial_state='q0',
    final_states={'q5'}
)
symbols = ['0', '1']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")

```

```
100011 rejected
00101 rejected
1100 rejected
110010 rejected
00001101 rejected
00001101 rejected
0001001 accepted
0011101 rejected
Es Minimo
```

c)



X No acepta válidas como "aazaazc"

Estado inicia l= {q0]

Estado final = {q4} ✓

Estados = {q0, q1, q2, q3, q4} ✓

~~Lenguaje~~ = {0,1} X

Transiciones =

X

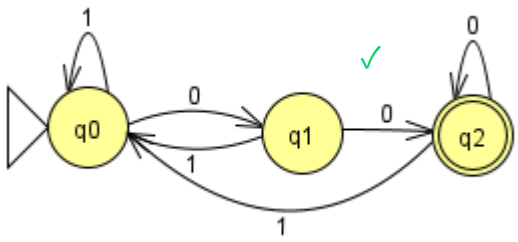
input/estado	a	c	z
q0	q1	q0	q0
q1	q2	q0	q0
q2	q2	q0	q3
q3	X q0	q4	q0
q4	q4	q4	q4

```python
import random
from automata.fa.dfa import DFA

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3', 'q4'},
    input_symbols={'a', 'z', 'c'},
    transitions={
        'q0': {'a': 'q1', 'z': 'q0', 'c': 'q0'},
        'q1': {'a': 'q2', 'z': 'q0', 'c': 'q0'},
        'q2': {'a': 'q2', 'z': 'q3', 'c': 'q0'},
        'q3': {'a': 'q0', 'z': 'q0', 'c': 'q4'},
        'q4': {'a': 'q4', 'z': 'q0', 'c': 'q0'}
    },
    initial_state='q0',
    final_states={'q4'}
)
```

```python
symbols = ['a', 'z', 'c']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
zz rejected
zazaaazc accepted
zca rejected
az rejected
ccccccz rejected
zazz rejected
azzzzz rejected
czzaac rejected
Es Minimo
```

d)



Estado inicia l= {q0]

Estado final = {q2}

Estados = {q0, q1, q2}

~~Lenguaje~~ = {0,1}

Transiciones =

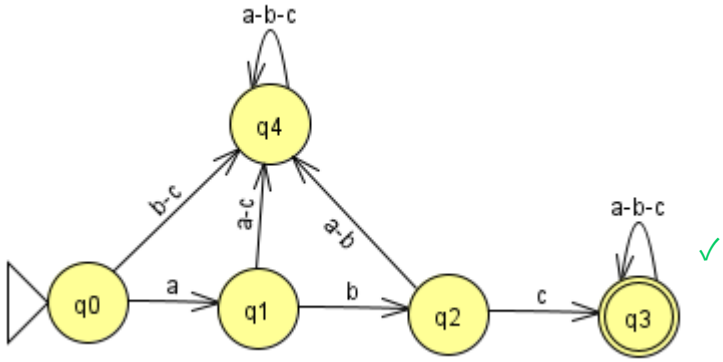
input/estado	0	1
q0	q1	q0
q1	q2	q0
q2	q2	q0

```python
import random
from automata.fa.dfa import DFA

dfa = DFA(
    states={'q0', 'q1', 'q2'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q1', '1': 'q0'},
        'q1': {'0': 'q2', '1': 'q0'},
        'q2': {'0': 'q2', '1': 'q0'}
    },
    initial_state='q0',
    final_states={'q2'}
)
symbols = ['0', '1']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
1010 rejected
1111 rejected
000000 accepted
1000 accepted
01 rejected
1101100 accepted
01 rejected
01 rejected
Es Minimo
```

Aimbinder Tiago Gabriel, Flores Mauricio Fernando, Nuñez Fabricio, Tabilo Ivo Ezequiel.

e)



Estado inicia l= {q0]

Estado final = {q3}

Estados = {q0, q1, q2, q3, q4}

~~Lenguaje~~ = {a, b, c}

Transiciones =

input/estado	a	b	c
q0	q1	q4	q4
q1	q4	q2	q4
q2	q4	q4	q3
q3	q3	q3	q3
q4	q4	q4	q4

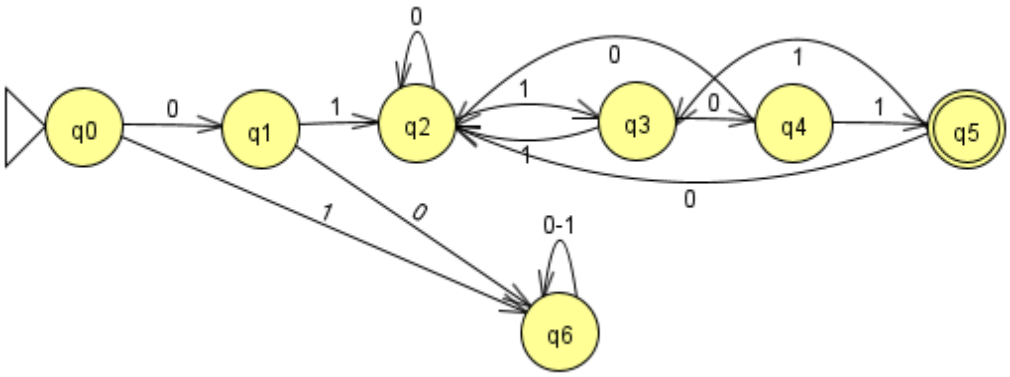
```python
import random
from automata.fa.dfa import DFA

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3', 'q4'},
    input_symbols={'a', 'b', 'c'},
    transitions={
        'q0': {'a': 'q1', 'b': 'q4', 'c': 'q4'},
        'q1': {'a': 'q4', 'b': 'q2', 'c': 'q4'},
        'q2': {'a': 'q4', 'b': 'q4', 'c': 'q3'},
        'q3': {'a': 'q3', 'b': 'q3', 'c': 'q3'},
        'q4': {'a': 'q4', 'b': 'q4', 'c': 'q4'}
    },
    initial_state='q0',
    final_states={'q3'}
)
symbols = ['a', 'b', 'c']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

Aimbinder Tiago Gabriel, Flores Mauricio Fernando, Nuñez Fabricio, Tabilo Ivo Ezequiel.

```
aaaccbc rejected
aaaac rejected
aac rejected
abcabcba accepted
abac rejected
aaa rejected
cacbb rejected
cabbaab rejected
Es Minimo
```

f)

No acepta válida a "0101"



Estado inicia l= {q0]

Estado final = {q5}

Estados = {q0, q1, q2, q3, q4, q5, q6}

~~Lenguaje~~ = {0,1}✓

Transiciones =

X

input/estado	0	1
q0	q1	q6
q1	q6	q2
q2	q2	q3
q3	q4	q2
q4	q2	q5
q5	q2	q3

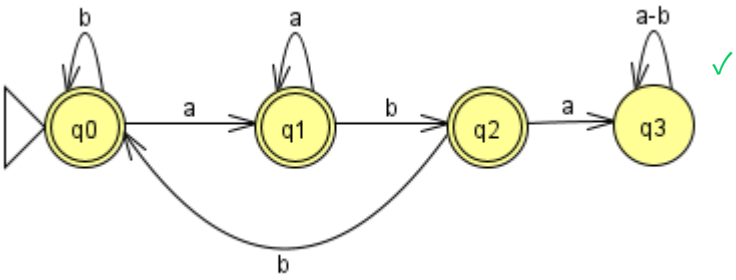
q6	q6	q6

```python
import random
from automata.fa.dfa import DFA

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3', 'q4', 'q5', 'q6'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q1', '1': 'q6'},
        'q1': {'0': 'q6', '1': 'q2'},
        'q2': {'0': 'q2', '1': 'q3'},
        'q3': {'0': 'q4', '1': 'q2'},
        'q4': {'0': 'q2', '1': 'q5'},
        'q5': {'0': 'q2', '1': 'q3'},
        'q6': {'0': 'q6', '1': 'q6'}
    },
    initial_state='q0',
    final_states={'q5'}
)
```

```python
symbols = ['0', '1']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
001101 rejected
1100000 rejected
1101111 rejected
10101100 rejected
110 rejected
001001 rejected
010101 accepted
0100100 rejected
Es Minimo
```

g)



Estado inicia l= {q0]

Estado final = {q0, q1, q2}

Estados = {q0, q1, q2, q3}

~~Lenguaje~~ = {a, b}

Transiciones =

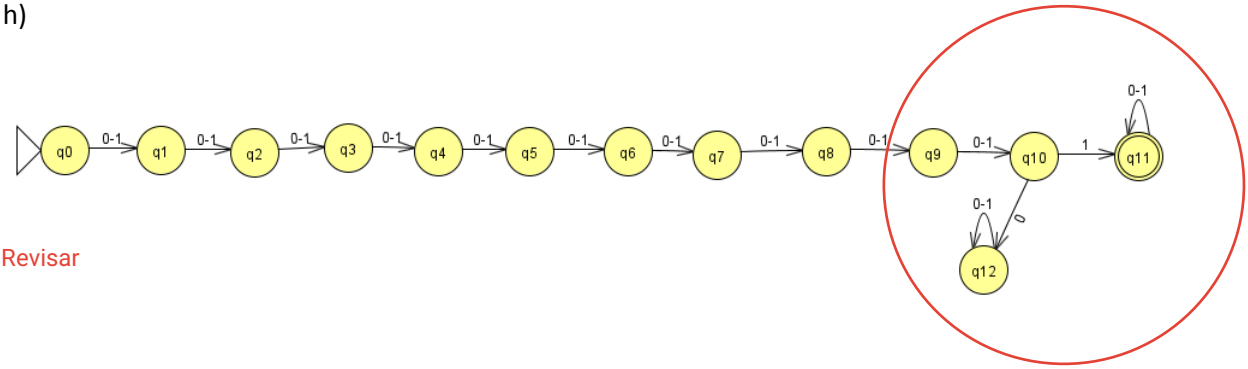
input/estado	a	b
q0	q1	q0
q1	q1	q2
q2	q3	q0
q3	q3	q3

```python
import random
from automata.fa.dfa import DFA

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3'},
    input_symbols={'a', 'b'},
    transitions={
        'q0': {'a': 'q1', 'b': 'q0'},
        'q1': {'a': 'q1', 'b': 'q2'},
        'q2': {'a': 'q3', 'b': 'q0'},
        'q3': {'a': 'q3', 'b': 'q3'}
    },
    initial_state='q0',
    final_states={'q0', 'q1', 'q2'}
)
symbols = ['a', 'b']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
bbaa accepted
aaba rejected
baabab rejected
abbaaba rejected
bab accepted
bbbbbb accepted
baaa accepted
bbbaabaa rejected
Es Minimo
```

Acepta cadenas no válidas con 0 en la décima posición.

h)



Revisar

Estado inicia l= {q0]

Estado final = {q11}

Estados = {q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12}

~~Lenguaje~~ = {0, 1} ✓

Transiciones =

input/estado	0	1
q0	q1	q1
q1	q2	q2
q2	q3	q3
q3	q4	q4
q4	q5	q5
q5	q6	q6
q6	q7	q7

q7	q8	q8
q8	q9	q9
q9	q10	q10
q10	q12	q11
q11	q11	q11
q12	q12	q12

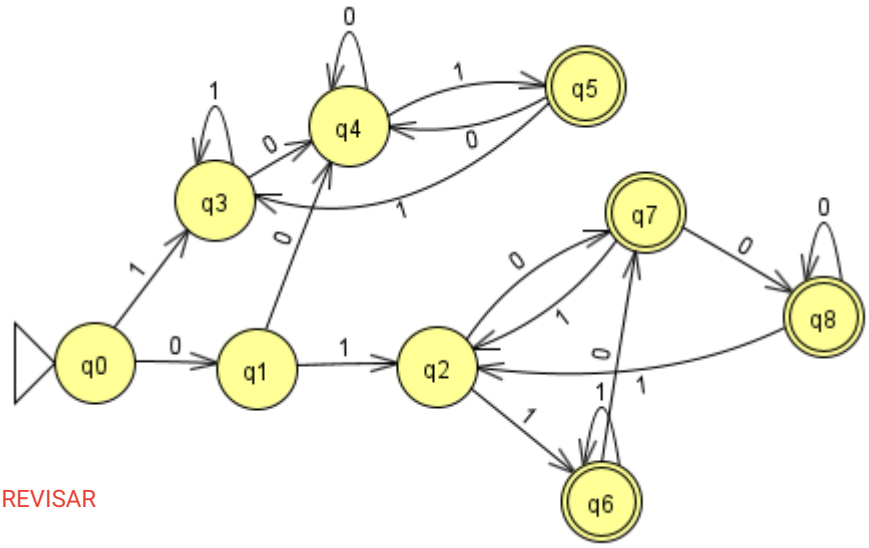
```python
from automata.fa.dfa import DFA
import random

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3', 'q4', 'q5', 'q6', 'q7', 'q8', 'q9', 'q10', 'q11', 'q12'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q1', '1': 'q1'},
        'q1': {'0': 'q2', '1': 'q2'},
        'q2': {'0': 'q3', '1': 'q3'},
        'q3': {'0': 'q4', '1': 'q4'},
        'q4': {'0': 'q5', '1': 'q5'},
        'q5': {'0': 'q6', '1': 'q6'},
        'q6': {'0': 'q7', '1': 'q7'},
        'q7': {'0': 'q8', '1': 'q8'},
        'q8': {'0': 'q9', '1': 'q9'},
        'q9': {'0': 'q10', '1': 'q10'},
        'q10': {'0': 'q12', '1': 'q11'},
        'q11': {'0': 'q11', '1': 'q11'},
        'q12': {'0': 'q12', '1': 'q12'}
    },
    initial_state='q0',
    final_states={'q11'}
)
```

```
symbols = ['0', '1']

for i in range(8):
    cadena = ''
    for j in range(random.randint(10, 12)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
00011010001 accepted
11010100001 accepted
1110100110 rejected
01111111101 accepted
011001011010 accepted
10111100111 accepted
00011110111 accepted
1110011011 rejected
Es Minimo
```

i)

No acepta válidas como "01,0101,etc"



REVISAR

Estado inicia l= {q0]

Estado final = {q5, q6, q7, q8}

Estados = {q0, q1, q2, q3, q4, q5, q6, q7, q8}

~~Lenguaje~~ = {0, 1} ✓

Transiciones =

input/estado	0	1
q0	q1	q3
q1	q4	q2
q2	q7	q6
q3	q4	q3
q4	q4	q5
q5	q4	q3
q6	q7	q6
q7	q8	q2
q8	q8	q2

```python
from automata.fa.dfa import DFA
import random

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3', 'q4', 'q5', 'q6', 'q7', 'q8'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q1', '1': 'q3'},
        'q1': {'0': 'q4', '1': 'q2'},
        'q2': {'0': 'q7', '1': 'q6'},
        'q3': {'0': 'q4', '1': 'q3'},
        'q4': {'0': 'q4', '1': 'q5'},
        'q5': {'0': 'q4', '1': 'q3'},
        'q6': {'0': 'q7', '1': 'q6'},
        'q7': {'0': 'q8', '1': 'q2'},
        'q8': {'0': 'q8', '1': 'q2'}
    },
    initial_state='q0',
    final_states={'q5', 'q6', 'q7', 'q8'}
)
```

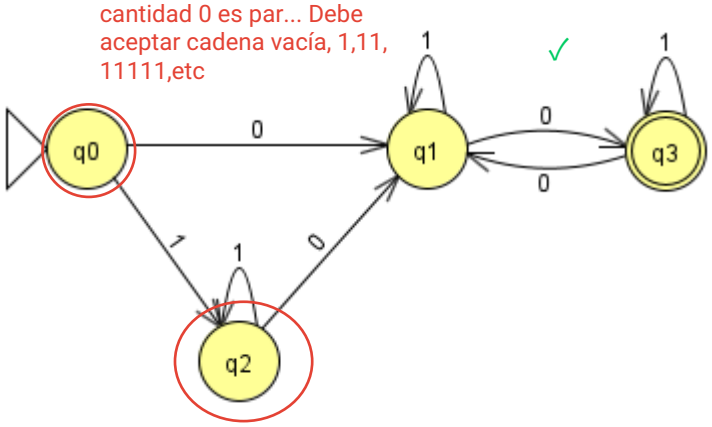
```python
symbols = ['0', '1']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
010 accepted
1101110 rejected
01011 accepted
01001 rejected
111 rejected
1000 rejected
001 accepted
00101011 rejected
No Es Minimo
```

Debería aceptarla

j)

cantidad 0 es par... Debe aceptar cadena vacía, 1,11, 11111,etc



Estado inicia l= {q0]

Estado final = {q3}

Estados = {q0, q1, q2, q3}

~~Lenguaje~~ = {0, 1}

Transiciones =

X

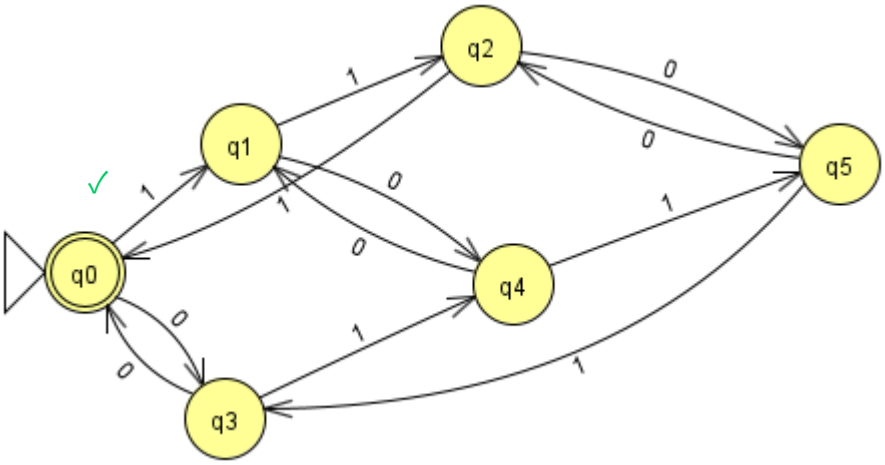
input/estado	0	1
q0	q1	q2
q1	q3	q1
q2	q1	q2
q3	q1	q3

```python
from automata.fa.dfa import DFA
import random

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q1', '1': 'q2'},
        'q1': {'0': 'q3', '1': 'q1'},
        'q2': {'0': 'q1', '1': 'q2'},
        'q3': {'0': 'q1', '1': 'q3'}
    },
    initial_state='q0',
    final_states={'q3'}
)
```

```python
symbols = ['0', '1']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
110111 rejected
00001010 accepted
10 rejected
110010 rejected
0100 rejected
01111 rejected
01000010 accepted
00010010 accepted
No Es Minimo
```

k)



Estado inicia l= {q0]

Estado final = {q0} ✓

Estados = {q0, q1, q2, q3, q4, q5} ✓

~~Lenguaje~~ = {0, 1} ✓

Transiciones = ✓

input/estado	0	1
q0	q3	q1
q1	q4	q2
q2	q5	q0
q3	q0	q4
q4	q1	q5

q5	q2	q3

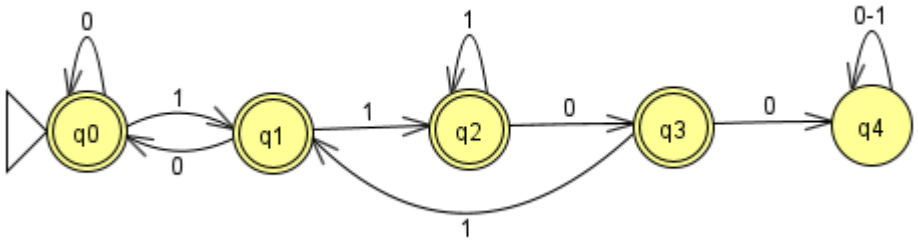
```python
from automata.fa.dfa import DFA
import random

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3', 'q4', 'q5'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q3', '1': 'q1'},
        'q1': {'0': 'q4', '1': 'q2'},
        'q2': {'0': 'q5', '1': 'q0'},
        'q3': {'0': 'q0', '1': 'q4'},
        'q4': {'0': 'q1', '1': 'q5'},
        'q5': {'0': 'q2', '1': 'q3'}
    },
    initial_state='q0',
    final_states={'q0'}
)
```

```python
symbols = ['0', '1']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
1110 rejected
1110000 accepted
100 rejected
00000 rejected
11000 rejected
10000 rejected
10101 accepted
1000010 rejected
Es Minimo
```

I)

Acepta cadenas NO válidas como 110100,1101000,etc



Estado inicia I= {q0]

Estado final = {q0, q1, q2, q3} ✓

Estados = {q0, q1, q2, q3, q4} ✓

~~Lenguaje~~ = {0, 1} ✓

Transiciones = REVISAR

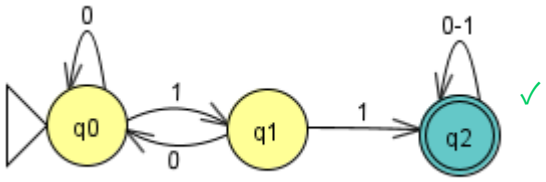
input/estado	0	1
q0	q0	q1
q1	q0	q2
q2	q3	q2
q3	q4	q1
q4	q4	q4

```python
from automata.fa.dfa import DFA
import random

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3', 'q4'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q0', '1': 'q1'},
        'q1': {'0': 'q0', '1': 'q2'},
        'q2': {'0': 'q3', '1': 'q2'},
        'q3': {'0': 'q4', '1': 'q1'},
        'q4': {'0': 'q4', '1': 'q4'}
    },
    initial_state='q0',
    final_states={'q0', 'q1', 'q2', 'q3'}
)
```

```python
symbols = ['0', '1']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
10001 accepted
1111000 rejected
0110000 rejected
101011 accepted
01 accepted
01010010 accepted
00100111 accepted
10011110 accepted
Es Minimo
```

m)



Estado inicia l= {q0]

Estado final = {q2}

Estados = {q0, q1, q2}

~~Lenguaje~~ = {0, 1}

Transiciones =

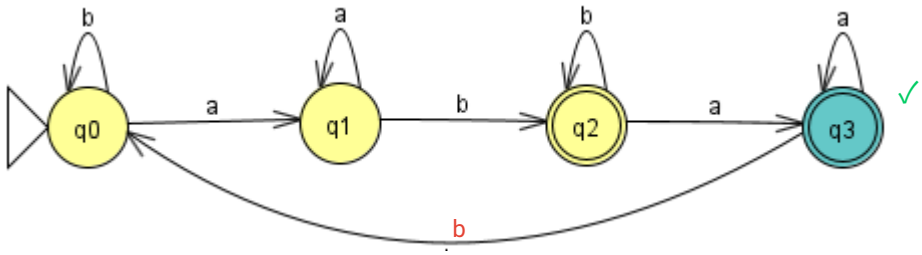
input/estado	0	1
q0	q0	q1
q1	q0	q2
q2	q2	q2

```python
from automata.fa.dfa import DFA
import random

dfa = DFA(
    states={'q0', 'q1', 'q2'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q0', '1': 'q1'},
        'q1': {'0': 'q0', '1': 'q2'},
        'q2': {'0': 'q2', '1': 'q2'}
    },
    initial_state='q0',
    final_states={'q2'}
)
```

```python
symbols = ['0', '1']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
10000 rejected
111100 accepted
010 rejected
0110101 accepted
111001 accepted
10101110 accepted
0100000 rejected
11 accepted
Es Minimo
```

n)



Estado inicia l= {q0]

Estado final = {q2, q3}

Estados = {q0, q1, q2, q3}

~~Lenguaje~~ = {a, b }

Transiciones =

input/estado	a	b
q0	q1	q0
q1	q1	q2
q2	q3	q2
q3	q3	q0

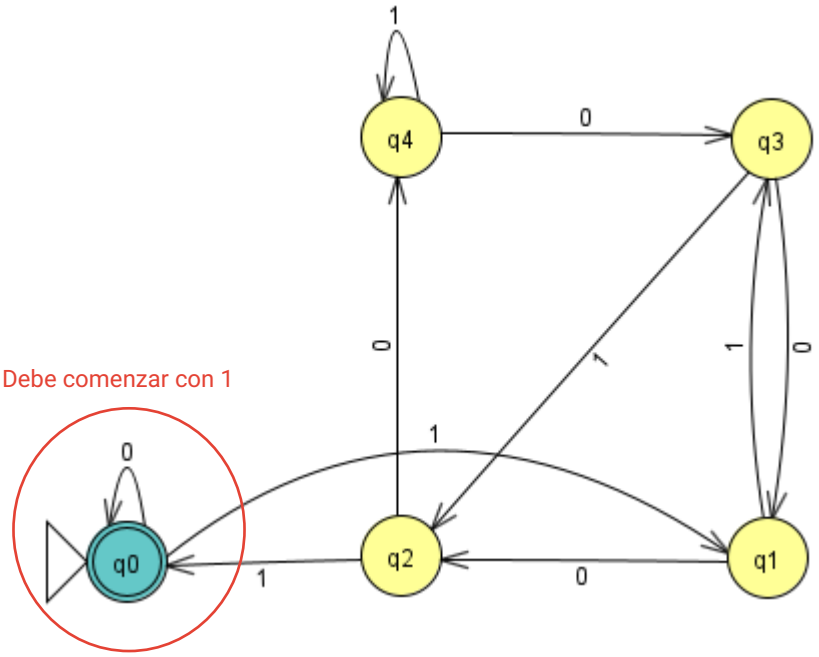
```python
from automata.fa.dfa import DFA
import random

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3'},
    input_symbols={'a', 'b'},
    transitions={
        'q0': {'a': 'q1', 'b': 'q0'},
        'q1': {'a': 'q1', 'b': 'q2'},
        'q2': {'a': 'q3', 'b': 'q2'},
        'q3': {'a': 'q3', 'b': 'q0'}
    },
    initial_state='q0',
    final_states={'q2', 'q3'}
)
```

```python
symbols = ['a', 'b']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

✓

```
babb accepted
abaaaa accepted
bba rejected
bbaa rejected
bbaaaba accepted
ab accepted
aabaaab rejected
babaaa accepted
Es Minimo
```

Ejercicio 6



Debe comenzar con 1

Estado inicia l= {q0]

Estado final = {q0} ✓

Estados = {q0, q1, q2, q3, q4} ✓

~~Lenguaje~~ = {0, 1} ✓

Transiciones =

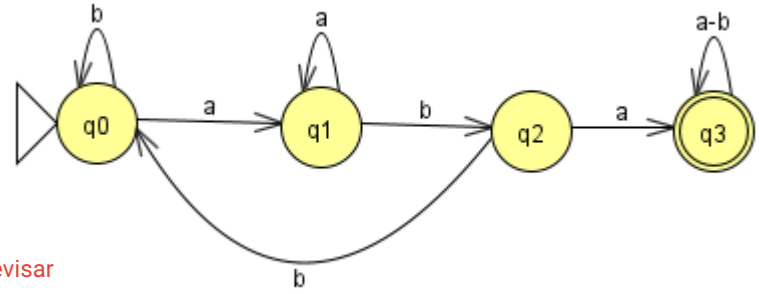
X Está bien encarado lo del múltiplo de 5 en binario, pero debe comenzar con 1

input/estado	0	1
q0	q0	q1
q1	q2	q3
q2	q4	q0

q3	q1	q2
q4	q3	q4

Ejercicio 7

Acepta cadenas no válidas como abbaba, abbabab, aaba, etc.



Revisar

Estado inicia l= {q0]

Estado final = {q3}

Estados = {q0, q1, q2, q3}

Lenguaje = {a, b}

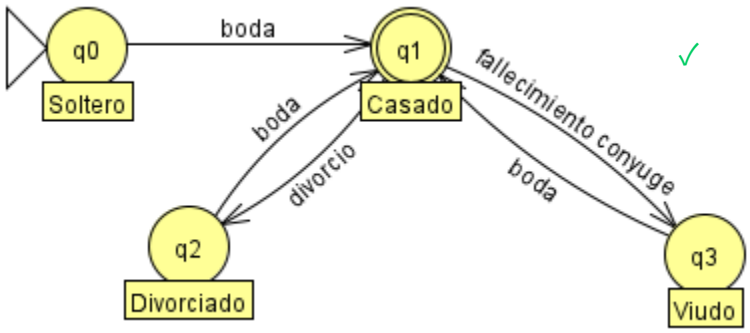
Transiciones =

input/estado	a	b
q0	q1	q0
q1	q1	q2
q2	q3	q0
q3	q3	q3

Ejercicio 8

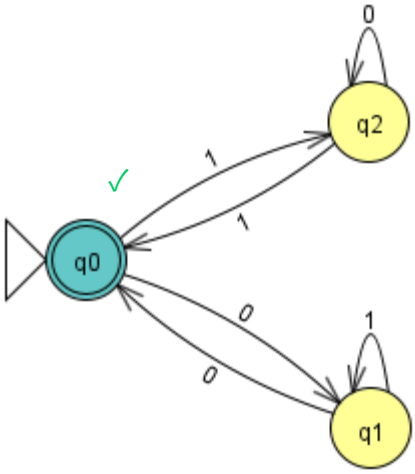
Ejemplo 1

En este ejemplo creamos un AFD simulando el estado civil de una persona, cuya salida deseada es que la persona este casada.✓



Ejemplo 2

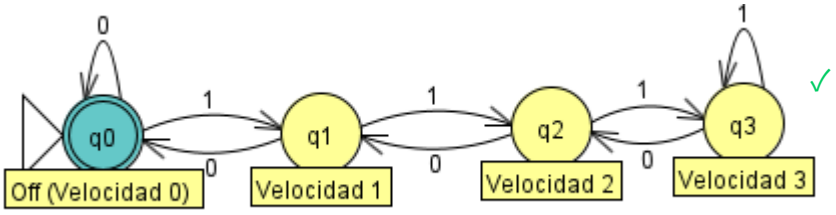
En este ejemplo creamos un AFD de un ascensor simplificado donde q0= inmóvil q2=ascender y q1= descender y la salida deseada siempre es que el ascensor termine inmóvil. El ascensor seguir ascendiendo o descendiendo hasta que el usuario indique lo contrario✓



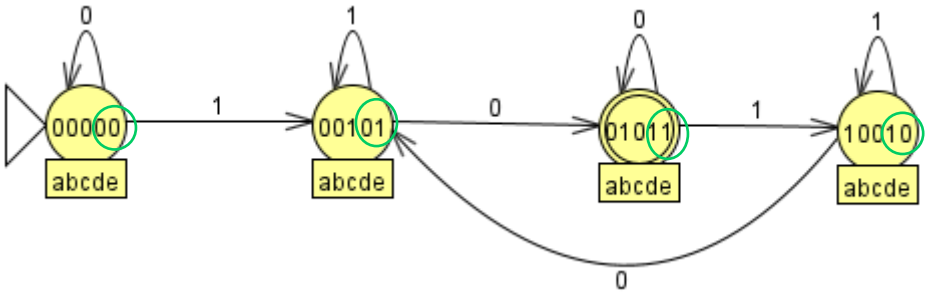
Que representaría el alfabeto {0,1} ?

Ejemplo 3

El ejemplo 3 representa los cambios de velocidad de una batidora y como salida deseada es off (velocidad 0) de esta. ✓



Ejercicio 9



✓ Para la aceptación solo es necesario conocer los estados de y1=D, y2=E

Estado inicia l= {q0]

Estado final = {q2} ✓

Estados = {q0, q1, q2, q3} ✓

~~Lenguaje~~ = {0, 1} ✓

Transiciones = ✓

input/estado	0	1
q0 (00000)	q0	q1
q1 (00101)	q2	q1
q2 (01011)	q2	q3
q3 (10010)	q1	q3

```python
from automata.fa.dfa import DFA
import random

dfa = DFA(
    states={'q0', 'q1', 'q2', 'q3'},
    input_symbols={'0', '1'},
    transitions={
        'q0': {'0': 'q0', '1': 'q1'},
        'q1': {'0': 'q2', '1': 'q1'},
        'q2': {'0': 'q2', '1': 'q3'},
        'q3': {'0': 'q1', '1': 'q3'}
    },
    initial_state='q0',
    final_states={'q2'}
)
symbols = ['0', '1']

for i in range(8):
    cadena = ''
    for j in range(random.randint(2, 8)):
        cadena = cadena + (random.choice(symbols))

    if dfa.accepts_input(cadena):
        print(cadena + ' accepted')
    else:
        print(cadena + ' rejected')

dfaMin = dfa.minify()
if dfaMin == dfa:
    print("Es Minimo")
else:
    print("No Es Minimo")
```

```
000001 rejected
10000 accepted
010 accepted
101111 rejected
10001 rejected
0011110 accepted
011 rejected
00000 rejected
Es Minimo
```

Aimbinder Tiago Gabriel, Flores Mauricio Fernando, Nuñez Fabricio, Tabilo Ivo Ezequiel.