APP WEEK-13 LAB

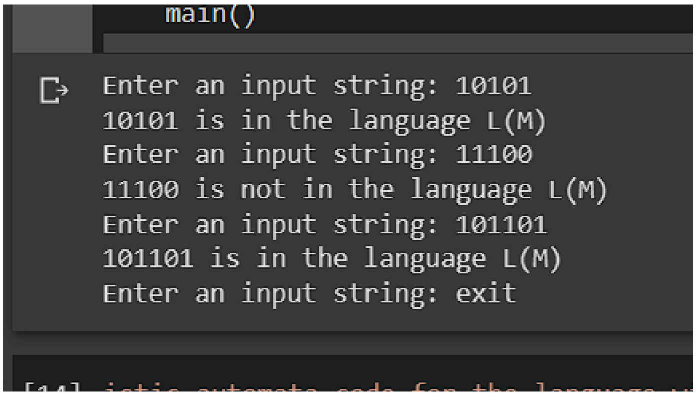
Q1.

Write a deterministic automata code for the language L(M)={w | wε{0,1}*} and W is a string that does not contain consecutive 0's

Code:

```
class Automata:
    def __init__(self):
        self.states = {"q0", "q1"}
        self.final_states = {"q0"}
        self.transitions = {("q0", "0"): "q1",
                            ("q0", "1"): "q0",
                            ("q1", "0"): "q1",
                            ("q1", "1"): "q0"}
        self.current_state = "q0"

    def process_input(self, input_str):
        for symbol in input_str:
            if (self.current_state, symbol) in self.transitions:
                self.current_state = self.transitions[(self.current_state, symbol)]
            else:
                return False
        return self.current_state in self.final_states
def main():
    automata = Automata()
    while True:
        input_str = input("Enter an input string: ")
        if input_str == "exit":
            break
        if automata.process_input(input_str):
            print(f"{input_str} is in the language L(M)")
        else:
            print(f"{input_str} is not in the language L(M)")
if __name__ == "__main__":
    main()
```

SnapShot:



```
    main()
```

```
Enter an input string: 10101
10101 is in the language L(M)
Enter an input string: 11100
11100 is not in the language L(M)
Enter an input string: 101101
101101 is in the language L(M)
Enter an input string: exit
```

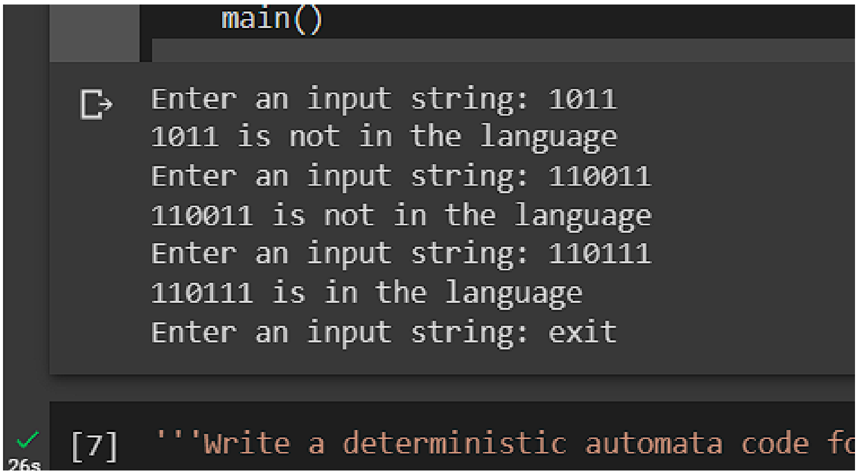
Q2.

Write a deterministic automata code for the language with Σ={0,1} accepts the set of all strings with three consecutive 1's

Code:

```
class Automata:
    def __init__(self):
        self.states = {"q0", "q1", "q2", "q3"}
        self.final_states = {"q3"}
        self.transitions = {("q0", "0"): "q0",
                            ("q0", "1"): "q1",
                            ("q1", "0"): "q0",
                            ("q1", "1"): "q2",
                            ("q2", "0"): "q0",
                            ("q2", "1"): "q3",
                            ("q3", "0"): "q0",
                            ("q3", "1"): "q3"}
        self.current_state = "q0"
    def process_input(self, input_str):
        for symbol in input_str:
            if (self.current_state, symbol) in self.transitions:
                self.current_state = self.transitions[(self.current_state, symbol)]
            else:
                return False
        return self.current_state in self.final_states
def main():
    automata = Automata()
    while True:
        input_str = input("Enter an input string: ")
        if input_str == "exit":
            break
        if automata.process_input(input_str):
            print(f"{input_str} is in the language")
        else:
            print(f"{input_str} is not in the language")
if __name__ == "__main__":
    main()
```

SnapShot:



```
    main()

Enter an input string: 1011
1011 is not in the language
Enter an input string: 110011
110011 is not in the language
Enter an input string: 110111
110111 is in the language
Enter an input string: exit

[7]  '''Write a deterministic automata code fo
```

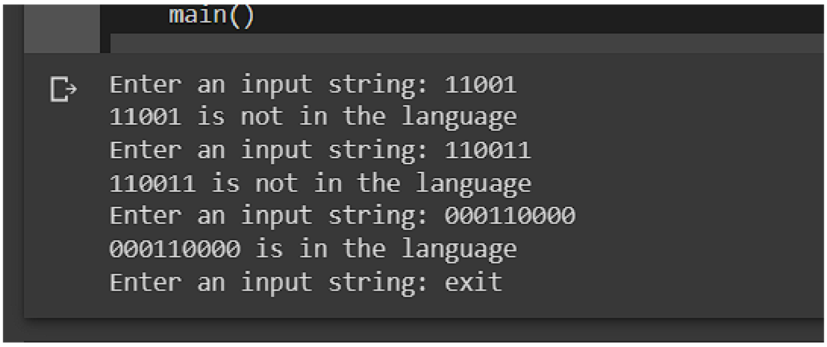
Q3.

Write a deterministic automata code for the language with Σ={0,1} accept seven number of 0's and even number of 1's

Code:

```
class Automata:
    def __init__(self):
        self.states = {"q0", "q1", "q2", "q3", "q4", "q5", "q6", "q7", "q8"}
        self.final_states = {"q8"}
        self.transitions = {("q0", "0"): "q1",
                            ("q0", "1"): "q0",
                            ("q1", "0"): "q2",
                            ("q1", "1"): "q3",
                            ("q2", "0"): "q4",
                            ("q2", "1"): "q3",
                            ("q3", "0"): "q2",
                            ("q3", "1"): "q3",
                            ("q4", "0"): "q5",
                            ("q4", "1"): "q6",
                            ("q5", "0"): "q7",
                            ("q5", "1"): "q6",
                            ("q6", "0"): "q5",
                            ("q6", "1"): "q6",
                            ("q7", "0"): "q8",
                            ("q7", "1"): "q6",
                            ("q8", "0"): "q8",
                            ("q8", "1"): "q6"}
        self.current_state = "q0"
    def process_input(self, input_str):
        for symbol in input_str:
            if (self.current_state, symbol) in self.transitions:
                self.current_state = self.transitions[(self.current_state, symbol)]
            else:
                return False
        return self.current_state in self.final_states
def main():
    automata = Automata()
    while True:
        input_str = input("Enter an input string: ")
        if input_str == "exit":
            break
        if automata.process_input(input_str):
            print(f"{input_str} is in the language")
        else:
            print(f"{input_str} is not in the language")
if __name__ == "__main__":
    main()
```

SnapShot:



```
    main()
```

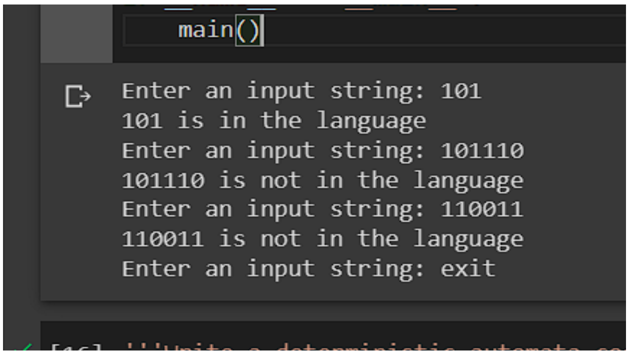
```
Enter an input string: 11001
11001 is not in the language
Enter an input string: 110011
110011 is not in the language
Enter an input string: 000110000
000110000 is in the language
Enter an input string: exit
```

Q4. Write a deterministic automata code for the language with Σ={0,1} accepts the only input 101

Code:

```
class Automata:
    def __init__(self):
        self.states = {"q0", "q1", "q2"}
        self.final_states = {"q2"}
        self.transitions = {("q0", "1"): "q1",
                            ("q1", "0"): "q2",
                            ("q2", "1"): "q2"}
        self.current_state = "q0"
    def process_input(self, input_str):
        for symbol in input_str:
            if (self.current_state, symbol) in self.transitions:
                self.current_state = self.transitions[(self.current_state, symbol)]
            else:
                return False
        return self.current_state in self.final_states
def main():
    automata = Automata()
    while True:
        input_str = input("Enter an input string: ")
        if input_str == "exit":
            break
        if automata.process_input(input_str):
            print(f"{input_str} is in the language")
        else:
            print(f"{input_str} is not in the language")
if __name__ == "__main__":
    main()
```

SnapShot:



```
    main()

Enter an input string: 101
101 is in the language
Enter an input string: 101110
101110 is not in the language
Enter an input string: 110011
110011 is not in the language
Enter an input string: exit
```

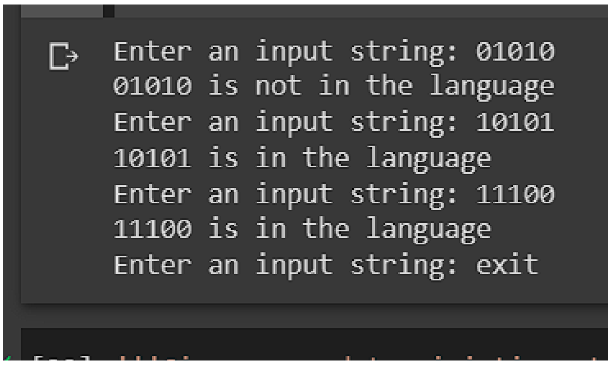
Q5.

Write a deterministic automata code for the language with Σ={0,1} accepts those string which starts with 1 and ends with 0

Code:

```
class Automata:
    def __init__(self):
        self.states = {"q0", "q1", "q2"}
        self.final_states = {"q2"}
        self.transitions = {("q0", "1"): "q1",
                            ("q1", "0"): "q2",
                            ("q1", "1"): "q1",
                            ("q2", "0"): "q2",
                            ("q2", "1"): "q2"}
        self.current_state = "q0"
    def process_input(self, input_str):
        for symbol in input_str:
            if (self.current_state, symbol) in self.transitions:
                self.current_state = self.transitions[(self.current_state, symbol)]
            else:
                return False
        return self.current_state in self.final_states
def main():
    automata = Automata()
    while True:
        input_str = input("Enter an input string: ")
        if input_str == "exit":
            break
        if automata.process_input(input_str):
            print(f"{input_str} is in the language")
        else:
            print(f"{input_str} is not in the language")
if __name__ == "__main__":
    main()
```

SnapShot:



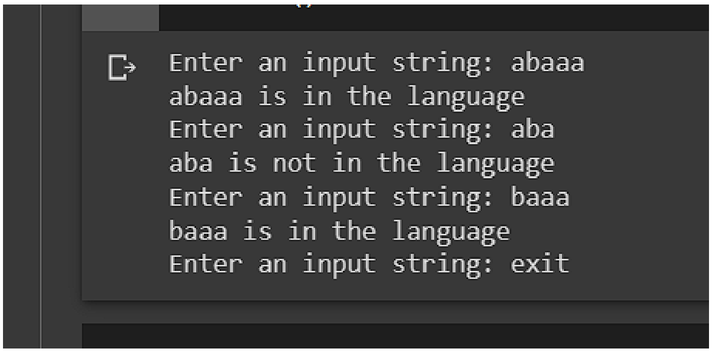
```
Enter an input string: 01010
01010 is not in the language
Enter an input string: 10101
10101 is in the language
Enter an input string: 11100
11100 is in the language
Enter an input string: exit
```

Q6. Give a non-deterministic automata code for (a|b)*aa

Code:

```
class Automata:
    def __init__(self):
        self.states = {"q0", "q1", "q2", "q3"}
        self.final_states = {"q3"}
        self.transitions = {("q0", "a"): {"q0", "q1"},
                            ("q0", "b"): {"q0"},
                            ("q1", "a"): {"q2"},
                            ("q2", "a"): {"q3"},
                            ("q2", "b"): {"q0", "q1", "q2", "q3"},
                            ("q3", "a"): {"q3"},
                            ("q3", "b"): {"q0", "q1", "q2", "q3"}}
        self.start_state = "q0"
    def process_input(self, input_str):
        current_states = {self.start_state}
        for symbol in input_str:
            next_states = set()
            for state in current_states:
                if (state, symbol) in self.transitions:
                    next_states |= self.transitions[(state, symbol)]
            current_states = next_states
        return bool(current_states & self.final_states)
def main():
    automata = Automata()
    while True:
        input_str = input("Enter an input string: ")
        if input_str == "exit":
            break
        if automata.process_input(input_str):
            print(f"{input_str} is in the language")
        else:
            print(f"{input_str} is not in the language")
if __name__ == "__main__":
    main()
```

SnapShot:



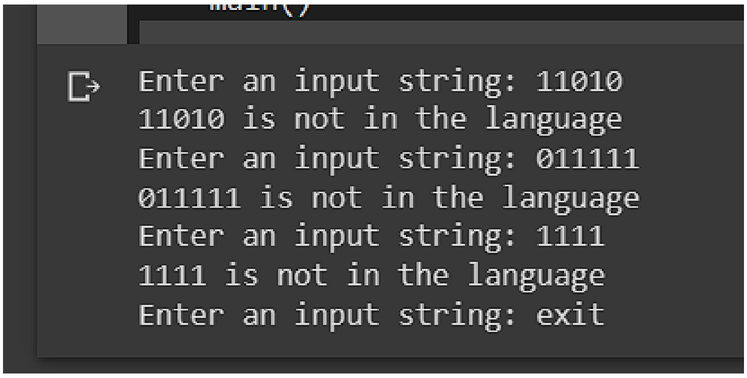
```
Enter an input string: abaaa
abaaa is in the language
Enter an input string: aba
aba is not in the language
Enter an input string: baaa
baaa is in the language
Enter an input string: exit
```

Q7. Give a non-deterministic automata code for these to fall binary strings that have either the number of 0’s odd, or the number of 1’s not a multiple of 3, or both

Code:

```
import sympy

# Define the variables
x, y = sympy.symbols('x y')

# Define the equations
eq1 = sympy.Eq(x + y, 2)
eq2 = sympy.Eq(2*x + y, 0)

# Solve the system of equations
sol = sympy.solve((eq1, eq2), (x, y))

# Print the solution
print("The solution is:", sol) class Automata:
    def __init__(self):
        self.states = {"q0", "q1", "q2", "q3", "q4", "q5", "q6"}
        self.final_states = {"q6"}
        self.transitions = {("q0", "0"): {"q1"},
                            ("q0", "1"): {"q2"},
                            ("q1", "0"): {"q0"},
                            ("q1", "1"): {"q3"},
                            ("q2", "0"): {"q4"},
                            ("q2", "1"): {"q0"},
                            ("q3", "0"): {"q5"},
                            ("q3", "1"): {"q2"},
                            ("q4", "0"): {"q6"},
                            ("q4", "1"): {"q4"},
                            ("q5", "0"): {"q5"},
                            ("q5", "1"): {"q6"},
                            ("q6", "0"): {"q6"},
                            ("q6", "1"): {"q6"}}
        self.start_state = "q0"
    def process_input(self, input_str):
        current_states = {self.start_state}
        for symbol in input_str:
            next_states = set()
            for state in current_states:
                if (state, symbol) in self.transitions:
                    next_states |= self.transitions[(state, symbol)]
            current_states = next_states
        return bool(current_states & self.final_states)
def main():
    automata = Automata()
    while True:
        input_str = input("Enter an input string: ")
        if input_str == "exit":
            break
        if automata.process_input(input_str):
            print(f"{input_str} is in the language")
        else:
            print(f"{input_str} is not in the language")
if __name__ == "__main__":
    main()
```

SnapShot:



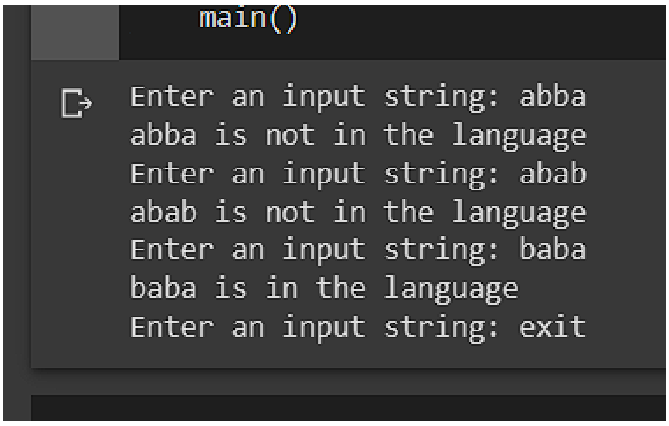
```
main()
Enter an input string: 11010
11010 is not in the language
Enter an input string: 011111
011111 is not in the language
Enter an input string: 1111
1111 is not in the language
Enter an input string: exit
```

Q8. Give a non-deterministic automata code for the language L=(ab)*(ba)*Uaa

Code:

```
class Automata:
    def __init__(self):
        self.states = {"q0", "q1", "q2", "q3", "q4", "q5", "q6"}
        self.final_states = {"q1", "q3", "q6"}
        self.transitions = {("q0", "a"): {"q0", "q1"},
                            ("q0", "b"): {"q2"},
                            ("q1", "b"): {"q2"},
                            ("q2", "a"): {"q3", "q4"},
                            ("q3", "a"): {"q6"},
                            ("q4", "b"): {"q5"},
                            ("q5", "a"): {"q6"},
                            ("q6", "a"): {"q6"},
                            ("q6", "b"): {"q2"}}
        self.start_state = "q0"

    def process_input(self, input_str):
        current_states = {self.start_state}
        for symbol in input_str:
            next_states = set()
            for state in current_states:
                if (state, symbol) in self.transitions:
                    next_states |= self.transitions[(state, symbol)]
            current_states = next_states
        return bool(current_states & self.final_states)
def main():
    automata = Automata()
    while True:
        input_str = input("Enter an input string: ")
        if input_str == "exit":
            break
        if automata.process_input(input_str):
            print(f"{input_str} is in the language")
        else:
            print(f"{input_str} is not in the language")
if __name__ == "__main__":
    main()
```

<u>SnapShot:</u>



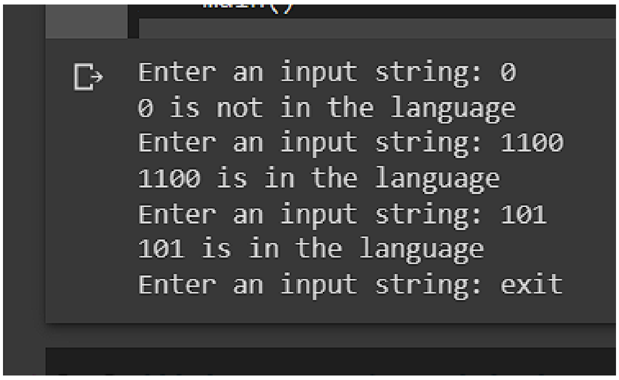
```
    main()

Enter an input string: abba
abba is not in the language
Enter an input string: abab
abab is not in the language
Enter an input string: baba
baba is in the language
Enter an input string: exit
```

Q9. Give a non-deterministic automata code for the language L that have atleast two consecutive 0's or 1's

<u>Code:</u>

```
class Automata:
    def __init__(self):
        self.states = {"q0", "q1", "q2"}
        self.final_states = {"q2"}
        self.transitions = {("q0", "0"): {"q0", "q1"},
                           ("q0", "1"): {"q0", "q1"},
                           ("q1", "0"): {"q2"},
                           ("q1", "1"): {"q2"},
                           ("q2", "0"): {"q2"},
                           ("q2", "1"): {"q2"}}
        self.start_state = "q0"
    def process_input(self, input_str):
        current_states = {self.start_state}
        for symbol in input_str:
            next_states = set()
            for state in current_states:
                if (state, symbol) in self.transitions:
                    next_states |= self.transitions[(state, symbol)]
            current_states = next_states
        return bool(current_states & self.final_states)
def main():
    automata = Automata()
    while True:
        input_str = input("Enter an input string: ")
        if input_str == "exit":
            break
        if automata.process_input(input_str):
            print(f"{input_str} is in the language")
        else:
            print(f"{input_str} is not in the language")
if __name__ == "__main__":
    main()
```

SnapShot:



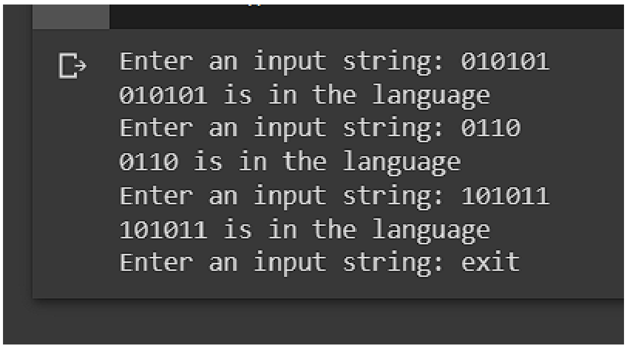
```
Enter an input string: 0
0 is not in the language
Enter an input string: 1100
1100 is in the language
Enter an input string: 101
101 is in the language
Enter an input string: exit
```

Q10. Give a non-deterministic automata code for the language L=(01U010)*

Code:

```
import numpy as np

# Define the coefficient matrix and the right-hand side vector
A = np.array([[3, 7], [4, -2]])
b = np.array([[12], [5]])

# Solve the system of equations using matrix inversion
x = np.linalg.inv(A) @ b

# Print the solution
print("The solution is:", x) class Automata:
    def __init__(self):
        self.states = {"q0", "q1", "q2", "q3"}
        self.final_states = {"q0", "q1", "q2", "q3"}
        self.transitions = {("q0", "0"): {"q1", "q3"},
                            ("q0", "1"): {"q2"},
                            ("q1", "0"): {"q1", "q3"},
                            ("q1", "1"): {"q2"},
                            ("q2", "0"): {"q0"},
                            ("q2", "1"): {"q2"},
                            ("q3", "0"): {"q1", "q3"},
                            ("q3", "1"): {"q2"}}
        self.start_state = "q0"
    def process_input(self, input_str):
        current_states = {self.start_state}
        for symbol in input_str:
            next_states = set()
            for state in current_states:
                if (state, symbol) in self.transitions:
                    next_states |= self.transitions[(state, symbol)]
            current_states = next_states
        return bool(current_states & self.final_states)
def main():
    automata = Automata()
    while True:
        input_str = input("Enter an input string: ")
        if input_str == "exit":
            break
        if automata.process_input(input_str):
            print(f"{input_str} is in the language")
        else:
            print(f"{input_str} is not in the language")
if __name__ == "__main__":
    main()
```

SnapShot:



```
Enter an input string: 010101
010101 is in the language
Enter an input string: 0110
0110 is in the language
Enter an input string: 101011
101011 is in the language
Enter an input string: exit
```