Python Code:

```
import cvxpy as cp
import numpy as np

# Define system matrices for SYS1
A1 = np.array([[-4, 1], [0, 2]])
B1 = np.array([[1], [0]])

# Define system matrices for SYS2
A2 = np.array([[-3, 2], [4, 1]])
B2 = np.array([[0], [1]])

# Define the decision variable for the control gain K
n = A1.shape[0]  # = A2.shape[0] Dimension of state vector
m = B1.shape[1]  # = B2.shape[1] Dimension of control input
K1 = cp.Variable((m, n))
K2 = cp.Variable((m, n))

# Define the LMI matrices
P1 = cp.Variable((n, n), symmetric=True)
P2 = cp.Variable((n, n), symmetric=True)

# Define the LMI constraints for SYS1
constraints1 = [A1 @ P1 + P1 @ A1.T + B1 @ K1 + K1.T @ B1.T << 0, P1 >> 0.001 *
np.eye(n)]

# Define the LMI constraints for SYS2
constraints2 = [A2 @ P2 + P2 @ A2.T + B2 @ K2 + K2.T @ B2.T << 0, P2 >> 0.001 *
np.eye(n)]

# Create an optimization problem for SYS1
problem1 = cp.Problem(cp.Minimize(0), constraints1)

# Create an optimization problem for SYS2
problem2 = cp.Problem(cp.Minimize(0), constraints2)

# Solve the LMI for SYS1
problem1.solve()

# Solve the LMI for SYS2
problem2.solve()

# Check the optimization results and obtain the stabilizing control gain K
```

```python
if problem1.status == cp.OPTIMAL:
    print('SYS1 is stabilizable.')
    K1 = K1.value
    print("K1 = ", K1)
else:
    print('SYS1 is not stabilizable.')
    K1 = None

if problem2.status == cp.OPTIMAL:
    print('SYS2 is stabilizable.')
    K2 = K2.value
    print("K2 = ", K2)
else:
    print('SYS2 is not stabilizable.')
    K2 = None
```

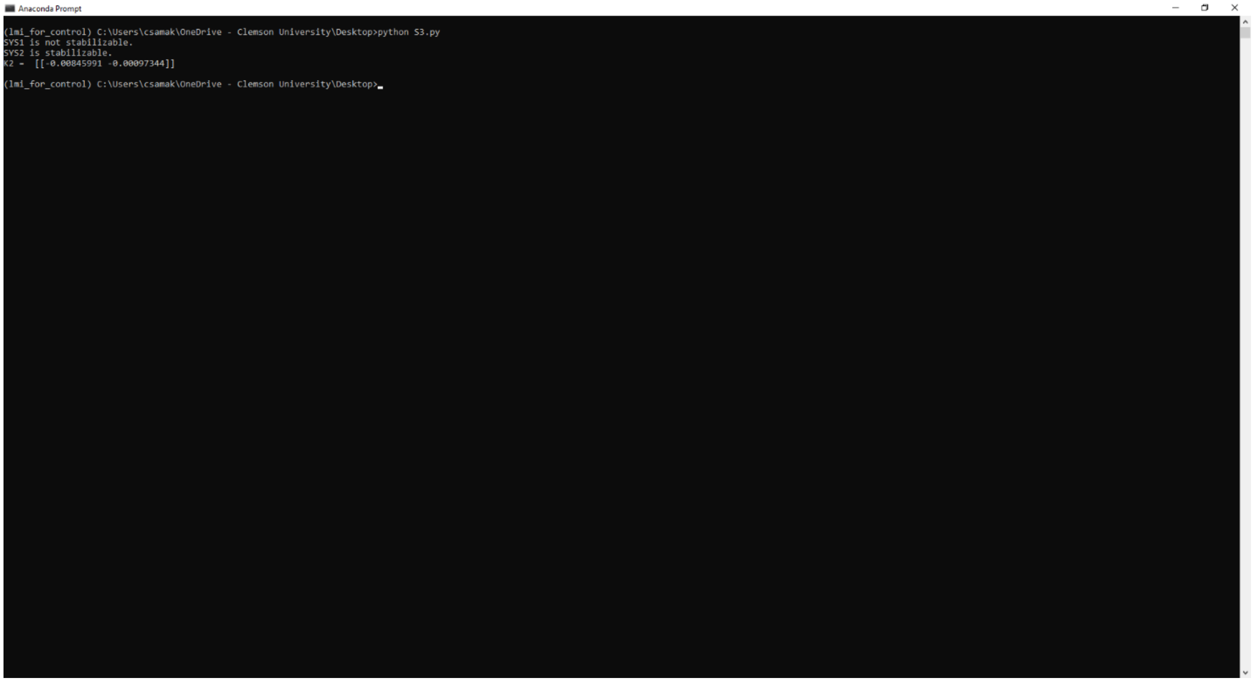
Output:

SYS1 is not stabilizable.

SYS2 is stabilizable.

K2 = [[-0.00845991 -0.00097344]]

Screenshot:



```
(lmi_for_control) C:\Users\csamak\OneDrive - Clemson University\Desktop>python S3.py
SYS1 is not stabilizable.
SYS2 is stabilizable.
K2 =  [[-0.00845991 -0.00097344]]

(lmi_for_control) C:\Users\csamak\OneDrive - Clemson University\Desktop>
```