**Name: Adithya M**

**SRN: PES1UG20CS621**

**MATLAB Assignment**

Hill Cipher:

Encryption:

function result = encrypt

text = input('Enter the text to be encrypted:\n','s');

k = [4 1;3 7];

a = double(text);

a = reshape(a,2,(length(text)/2));

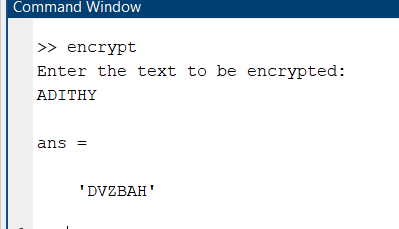
code = mod(k\*(a-65) ,26) + 65;

result = reshape(code,1,length(text));

result = char(result);

end

Output:



Decryption

function decrypt(text)

clc;

close all;

key = [4 1;3 7];

key(1,2) = -key(1,2);

key(2,1) = -key(2,1);

temp = key(1,1);

key(1,1) = key(2,2);

key(2,2) = temp;

[~, C, ~] = gcd(key(1,1)\*key(2,2)-key(1,2)\*key(2,1),26);

d = mod(C,26);

key = mod(d\*key, 26);

a = double(text);

b = reshape(a,2,length(text)/2);

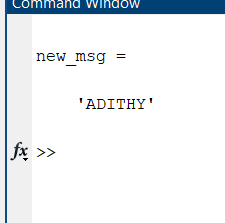
b = b - 65;

y = mod(key\*(b),26) + 65;

new\_msg = reshape(y,1,length(text));

new\_msg = char(new\_msg)

end



Processing

Translation: Moving the Grid

def setup():

size(200, 200)

background(255)

noStroke()

# draw the original position in gray

fill(192)

rect(20, 20, 40, 40)

# draw a translucent red rectangle by changing the coordinates

fill(255, 0, 0, 128)

rect(20 + 60, 20 + 80, 40, 40)

# draw a translucent blue rectangle by translating the grid

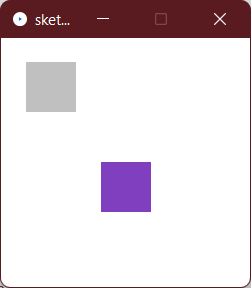
fill(0, 0, 255, 128)

pushMatrix()

translate(60, 80)

rect(20, 20, 40, 40)

popMatrix()



House

def setup():

size(400, 100)

background(255)

for i in xrange(10,350,50):

house(i, 20)

def house(x, y):

triangle(x + 15, y, x, y + 15, x + 30, y + 15)

rect(x, y + 15, 30, 30)

rect(x + 12, y + 30, 10, 15)

def house(x, y):

pushMatrix()

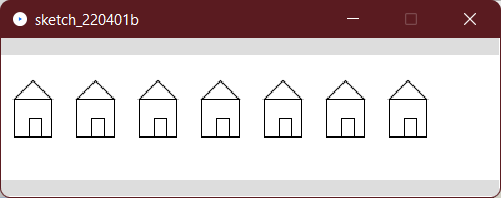
translate(x, y)

triangle(15, 0, 0, 15, 30, 15)

rect(0, 15, 30, 30)

rect(12, 30, 10, 15)

popMatrix()



### Rotation

def setup():

size(200, 200)

background(255)

smooth()

noStroke()

def draw():

if (frameCount % 10 == 0):

fill(frameCount \* 3 % 255, frameCount \* 5 % 255,

frameCount \* 7 % 255)

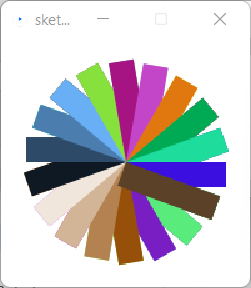
pushMatrix()

translate(100, 100)

rotate(radians(frameCount \* 2 % 360))

rect(0, 0, 80, 20)

popMatrix()



### Scaling

def setup():

size(200,200)

background(255)

stroke(128)

rect(20, 20, 40, 40)

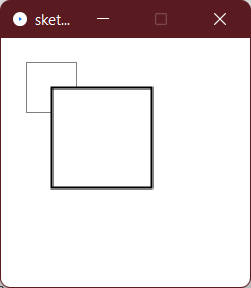
stroke(0)

pushMatrix()

scale(2.0)

rect(20, 20, 40, 40)

popMatrix()



### Order Matters

def setup():

size(200, 200)

background(255)

smooth()

line(0, 0, 200, 0) # draw axes

line(0, 0, 0, 200)

pushMatrix()

fill(255, 0, 0) # red square

rotate(radians(30))

translate(70, 70)

scale(2.0)

rect(0, 0, 20, 20)

popMatrix()

pushMatrix()

fill(255) # white square

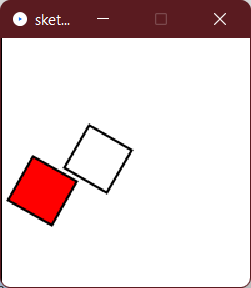
translate(70, 70)

rotate(radians(30))

scale(2.0)

rect(0, 0, 20, 20)

popMatrix()



### Case Study: An Arm-Waving Robot

armAngle = 0

angleChange = 5

ANGLE\_LIMIT = 135

def setup():

size(200, 200)

smooth()

frameRate(30)

def draw():

global armAngle, angleChange, ANGLE\_LIMIT

print armAngle

background(255)

pushMatrix()

translate(50, 50) # place robot so arms are always on screen

drawRobot()

armAngle += angleChange

# if the arm has moved past its limit,

# reverse direction and set within limits.

if (armAngle > ANGLE\_LIMIT or armAngle < 0):

angleChange = -angleChange

armAngle += angleChange

popMatrix()

def drawRobot():

noStroke()

fill(38, 38, 200)

rect(20, 0, 38, 30) # head

rect(14, 32, 50, 50) # body

drawLeftArm()

drawRightArm()

rect(22, 84, 16, 50) # left leg

rect(40, 84, 16, 50) # right leg

fill(222, 222, 249)

ellipse(30, 12, 12, 12) # left eye

ellipse(47, 12, 12, 12) # right eye

def drawLeftArm():

global armAngle

pushMatrix()

translate(12, 32)

rotate(radians(armAngle))

rect(-12, 0, 12, 37) # left arm

popMatrix()

def drawRightArm():

global armAngle

pushMatrix()

translate(66, 32)

rotate(radians(-armAngle))

rect(0, 0, 12, 37) # right arm

popMatrix()

