Bio:
My name is Christian Prather I am very excited about this course! I love robotics and in particular have always been drawn to computer vision and its applications. I currently work at a startup where a lot of my job has involved self learning OpenCV in both C++ and Python so I am very glad I can finally learn it in a proper setting. I hope to get a better understanding of the underlying algorithms used in OpenCV as well as learn how I may develop new applications of the software and math in robotics, i.e using it to better do the projects I currently work on for my job.

Christer Pather | Computer Vision HW0

1) a) $A = \begin{bmatrix} 4 & -2 \\ 1 & 1 \end{bmatrix}$ $|A| = 4 - (-2) = 6$

b) Trace $= 4+1 = 5$

c) $A^{-1} = \frac{1}{6} \cdot \begin{bmatrix} 1 & 2 \\ -1 & 4 \end{bmatrix} = \begin{bmatrix} 1/6 & 1/3 \\ -1/6 & 2/3 \end{bmatrix}$

d) $\begin{bmatrix} 4-\lambda & -2 \\ 1 & 1-\lambda \end{bmatrix} = 0$

$(4-\lambda)(1-\lambda) + 2 = 0$

$4 - 4\lambda - \lambda + \lambda^2 + 2 = 0$

e) $\lambda^2 - 5\lambda + 6 = 0$

$\lambda = 3$ or 2

$\begin{bmatrix} 4 & -2 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = 3\begin{bmatrix} x \\ y \end{bmatrix}$

$4x - 2y = 3\cancel{x}$

$x + y = 3y$

$x - 2y = 0$

$x - 2y = 0$

$y = \frac{x}{2}$ $\begin{bmatrix} 1 \\ .5 \end{bmatrix} \rightarrow \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

2) a) $\begin{bmatrix} 4 & -2 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 4 \\ 5 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 18 \\ 8 & 3 \end{bmatrix}$

b) $\begin{bmatrix} 3 & 4 \\ 5 & -1 \end{bmatrix} \cdot \begin{bmatrix} 4 & -2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 16 & -2 \\ 19 & -11 \end{bmatrix}$

3) a) $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}\begin{bmatrix} -1 & 2 & -3 \end{bmatrix}$ $-1 + 4 - 9 = 6$

b) $(2)(-3) - 3(2)$

$-6 \quad -6$ $= -12 = x$

$(3)(-1) - (1)(-3) = 0 = y$

$-3 \quad -2$

$(1)(2) - 2(-1) = 4$

$2 \quad +2$

$\begin{bmatrix} -12 \\ 0 \\ 4 \end{bmatrix}$

4)

a) 2, 3, 5, 7 $\frac{4}{10}$

b) 4.5

c) $\frac{1}{10} \cdot \frac{1}{10} = \frac{1}{100}$

5) black card $= \frac{26}{52}$ $\frac{1}{2} \cdot \frac{1}{13} = \boxed{\frac{1}{26}}$

6) 10000 total so $M_1 = .3$ $M_2 = .4$ $M_3 = .3$

$(.8)(.3) + (.9)(.3) + (.6)(.3) = \boxed{0.78}$

7) $P(A|B) = \frac{P(AB)}{P(B)}$ $P(A) = (0.01)$
$P(B) = (0.95)$

$P(AB) = \frac{(.95)(.01)}{.0198}$ $= 53.31\%$ 90 min

Total Time 40 minutes
Resource mathisfun.com

```python
# Christian Prather
# Took approx 25 minutes
# Had to use numpy offical documentation heavily
import numpy as np

mat = np.zeros((9,9))
def norm_dist(mean, sigma):
    for row in range (9):
        for column in range(9):
            mat[row][column] = (1/(2 * np.pi * sigma **2)) * np.e **(-(((row - mean) ** 2) + ((column - mean) ** 2)/ (2 * sigma ** 2)))
    print(mat)
def main():
    A = np.matrix('4 -2; 1 1')
    B = np.matrix('3 4; 5 -1')
    X = np.array([1,2,3])
    Y = np.array([-1,2,-3])
    print(A)
    print(B)

    print("Det {}".format(np.linalg.det(A)))
    print("Trace {}".format(A.trace()))
    print("Inv {}".format(np.linalg.inv(A)))
    values, vectors = np.linalg.eig(A)
    print("Eigenvalues {} Eigenvector {}".format(values, vectors))
    print("AB {}".format(np.matmul(A, B)))
    print("BA {}".format(np.matmul(B, A)))
    print("X dot Y {}".format(np.dot(X,Y)))
    print("X cross Y {}".format(np.cross(X,Y)))
    norm_dist(0, 1.0)
if __name__ == "__main__":
    main()
```