Name: Biken Maharjan
Discussion: Discussed some problems approaching idea w/ Riken M,Help from T.F
Source: Matplotlib guide,Princeton University Reference note.
Late submission: 2, Extra days left for the semester : 1

Q1) Codes Submitted

```
import numpy as np
from matplotlib import pyplot as plt
from numpy import zeros_like


def flip(p):
    return 1 if np.random.uniform(0.0,1.0) < p else 0
#####################################################################

def Bernoulli_hist(p,m):
    array = [0]*(m+1)
    for i in range(0,m):
        array[i] = flip(p)
    plt.figure(1)
    plt.hist(array,bins =2,rwidth=.4,align ='mid',
             weights = np.zeros_like(array)+1./len(array))
    plt.xlim(0,1)
    plt.title("Bernoulli Histogram")
    plt.xlabel("x")
    plt.ylabel("Pr")
    plt.show()
```

③ Prove that $X \sim G(p)$ is a probability distribution.

(i) $0 \le f(K) = 1$ ← Proof

$\Rightarrow 0 \le p \le 1$

$\Rightarrow 0 \le (1-p) \le 1$

$\Rightarrow K \in N \Rightarrow 0 \le (1-p)^{K-1} \le 1$

$\Rightarrow 0 \le p(1-p)^{K-1} \le 1$

(ii) $\sum_{K} f(K) = 1$ ← Proof

$$\sum_{K=1}^{\infty} p(1-p)^{K-1} \Rightarrow p\left(\frac{1}{1-(1-p)}\right) = p\left(\frac{1}{p}\right) \Rightarrow 1$$

④ $\sum_{i=1}^{\infty} (1-p)^i$

$\Rightarrow$ Using derivation

$$\frac{d}{dp} \sum_{i=1}^{\infty} (1-p)^i \Rightarrow \frac{d}{dp}\left((1-p) + (1-p)^2 + \dots\right) \Rightarrow \sum_{i=1}^{\infty} i(1-p)^{i} = \sum_{i=1}^{\infty} (1+p)(1-p)^{i-1}$$

$$\Rightarrow (1-p) \sum_{j=0}^{\infty} (1-p)^j \Rightarrow \frac{1-p}{p} \Rightarrow \frac{d\Sigma}{dp} = -\frac{1}{p^2}$$

$$\Rightarrow E = p\left[-\left(-\frac{1}{p^2}\right)\right] \Rightarrow \boxed{\frac{1}{p}}$$

⑤ Code Submitted

```
#############################################################
#[5]
def binomial_draw(n,p):
    array = []
    store_success = []
    for i in range(0,n):
        array += [flip(p)]

    for i in range(0,n):
        if array[i] == 1:
            store_success += [array[i]]
    return len(store_success)
```

⑥ Code Submitted

```
#######################################################
#[6]
def binom_trials(n,p,numExpts):
    var = []
    for i in range(0,numExpts):
        var +=[binomial_draw(n,p)]
    return var
```

⑦ Code Submitted & plotted below

```
#############################################################
#[7]
def binom_hist(n,p,numExpts):
    array = [0]*(numExpts+1)
    array = binom_trials(n,p,numExpts)
    plt.figure(1)
    plt.hist(array,bins = numExpts,rwidth=.4,align ='mid',
             weights = np.zeros_like(array)+1./len(array))
    plt.xlim(0,100)
    plt.title("Bernoulli Histogram")
    plt.xlabel("No# of Head")
    plt.ylabel("Pr")
    plt.show()
```

⑧ ⓐ range of $L = \{0,1,2,3\}$

ⓑ Using Binomial formula:

$$Pr[B(n,p) = K] = \binom{n}{K} p^K (1-p)^{n-K} \quad \forall K \in \{0,1,\ldots n\}$$

$Pr(L=0) = 1/8$, $Pr[L=1] = 3/8$, $Pr[L=2] = 3/8$, $Pr[L=3] = 1/8$

ⓒ $S \leftarrow -1$ ⓓ $S(L) = (3-L)-L$

$= 3L-2L$

{

(e) Range of (S): $\{-3, -1, 1, 3\}$

(f) The probability distribution func of S:

Using ~> Binomial Theorem:

$$Pr[S=-3] = \binom{3}{\frac{3-3}{2}}\left(\frac{1}{2}\right)^3 \rightsquigarrow \frac{1}{8}$$

$$Pr[S=-1] = \binom{3}{2} * \left(\frac{1}{2}\right)^3 = \frac{3}{8}$$

$$Pr[S=1] = \binom{3}{1} * \left(\frac{1}{2}\right)^3 = \frac{3}{8}$$

$$Pr[S=3] = \binom{3}{0} * \left(\frac{1}{2}\right)^3 = \frac{1}{8}$$

(g) Code Submitted

```
#8 -> G
def histogram():

    array = []
    s =[]
    length = 4

    for i in range(length):
     array += [3-2*i]

    for x in array:
        k = ((3-x)/2)
        s += [( (fact(3)/(fact(3-k)*fact(k))) *(1.0/8))]
    plt.bar(array,s,align='center', color='red')
    plt.xlabel('s')
    plt.ylabel(' pr[S = s] ')

##########################################################################
```

9(a) Range of $L = \{0,1,2,3,4,5,6,7,8\}$

(b) $Pr[L = l]$

```
#########################################################################
#9 -> B
def histogram_q8():

    array = []
    s =[]
    length = 9

    for i in range(length):
     array += [8-2*i]

    for x in array:
        k = ((8-x)/2)
        s += [( (fact(8)/(fact(8-k)*fact(k))) *(1.0/pow(2,8)))]
    plt.figure(1)
    plt.bar(array,s,align='center', color='blue')
    plt.xlabel('s')
    plt.ylabel(' pr[S = s] ')
#=========================================================================
def call_histograms():
    histogram_q8()
    histogram()
    plt.show()
#########################################################################
```

(10) $Pr[O/S=1] = P[J=0] \cup P[J=2]$

$= P[J=0] \rightsquigarrow \frac{70}{28} \;\big|\; P[J=1] = \frac{56}{28}$

$\Rightarrow \boxed{\frac{186}{2^8}}$

(11) $Pr[O/S=-1] = Pr[J=0] \cup Pr[J=-2]$

$\Rightarrow \boxed{126/28}$

(12) $Pr[O/S=3] = Pr[J=2] \cup P[J=4]$

$= 56/28 + 28/28$

$\Rightarrow \boxed{84/28}$

(13) $Pr[O] = \sum_{s \in Range(S)} Pr[O/S=s]\, Pr[S=s]$

$\approx \left(\frac{126}{256} * \frac{3}{8}\right) + \left(\frac{126}{256} * \frac{3}{8}\right) + \left(\frac{84}{256} * \frac{1}{8}\right) + \left(\frac{84}{256} * \frac{1}{8}\right)$

$\rightsquigarrow \boxed{0.4512}$

(14)

```
#14
def histogram():

    array = []
    s =[]
    length = 101

    for i in range(length):
     array += [100-2*i]

    for x in array:
        k = ((100-x)/2)
        s += [( (fact(100)/(fact(100-k)*fact(k))) *(1.0/pow(2,100)))]
    plt.bar(array,s,align='center', color='red')
    plt.xlabel('s')
    plt.ylabel(' pr[S = s] ')

#########################################################################
def histogram_q8():

    array = []
    s =[]
    length = 201

    for i in range(length):
     array += [200-2*i]

    for x in array:
        k = ((200-x)/2)
        s += [( (fact(200)/(fact(200-k)*fact(k))) *(1.0/pow(2,200)))]
    plt.figure(1)
    plt.bar(array,s,align='center', color='blue')
    plt.xlabel('s')
    plt.ylabel(' pr[S = s] ')
#=========================================================================
def call_histograms():

    histogram()
    histogram_q8()
    plt.show()
#########################################################################
```

```
#################################################################
def find_p():
    p=0
    for s in range(-100,101):
     p+= ((((fact(200)/(fact(200-((200-(s))/2))*fact((200-(s))/2))) *(1.0/pow(2,200)))+
          (((fact(200)/(fact(200-((200-(s-2))/2))*fact((200-(s-2))/2))) *(1.0/pow(2,200)))+
          +(((fact(200)/(fact(200-((200-(s+2))/2))*fact((200-(s+2))/2))) *(1.0/pow(2,200))))) *
          ((fact(100)/(fact(100-((100-(s))/2))*fact((100-(s))/2))) *(1.0/pow(2,100))))
    return(p)
#################################################################
```

Value of P(D) = 0.027 ~ 27%

(15) $E[X] = p$ [Source: princeton's Variance note]

$$J = \sum_{x=1}^{n} X \Rightarrow E[J] = E\sum_{x=1}^{n} X = \sum_{x=1}^{n} E[X] = \sum_{x=1}^{n} p = \boxed{pn}$$

(16) $$\left(\sum_{i=1}^{185} \left(\frac{\binom{2}{1}^{i}}{2}\right) + 1\right) / 2^{256}$$

$$P \Rightarrow \boxed{4.2 * 10^{-82}}$$

(17) $\lambda = P * n$

$\lambda \Rightarrow 4.2 * 10^{-22} * (2000000 \text{ trillion})$

$\lambda \Rightarrow 0.00085$ per sec

$\Rightarrow 3.02$ per hr

$\Rightarrow 72.58$ per day

(18) $E[Y] = \lambda$ #Help from: TA

$$E[Y] = \sum_{y=0}^{\infty} \frac{y e^{-\lambda} \lambda^{y}}{y}$$

$$E[Y] = \sum_{y=1}^{\infty} \frac{y\, e^{-\lambda} \lambda^{y-1}}{y!}$$

$$= \lambda \sum_{x=0}^{\infty} \frac{e^{-\lambda} \lambda^{x}}{x!} = \lambda * \Sigma p_i$$

$$= \lambda * 1 = \boxed{\lambda}$$

(19) Code Submitted & plot

```
import pandas as pd
data_frame = pd.read_csv('real_bitcoin.csv', delim_whitespace=True, header=None)
def histogram_poisson():
    y_series = []

    for i in xrange(59):
     y_series +=[(data_frame.iat[i+1, 2]-data_frame.iat[i, 2])/12.5]

    plt.figure (1)
    plt.hist(y_series,bins=6,rwidth = 0.5 ,align='mid',
             weights=np.zeros_like (y_series) + 1. /len(y_series))
    plt.xlim (100,190)
    plt.title ("PMF")
    plt.xlabel ("x")
    plt.ylabel("Pr[ X = x]")
    plt.show()
##############################################################################
```

(20) Code Submitted

```
#Done 20
def hash_exp(num_zeros):
    b1 = 'wubba lubba'
    b2 = 'dub dub'
    for nonce in xrange(max_nonce):
        hashResult = hashlib.sha256(b1+b2+str(nonce)).hexdigest()
        s = ("1"+ hashResult)
        binary = str("{0:020b}".format(int(s,16)))
        if (int(binary[1:num_zeros+1]) == 0):
            return (hashResult,nonce)
    return 0
```

Q21) Code Submitted

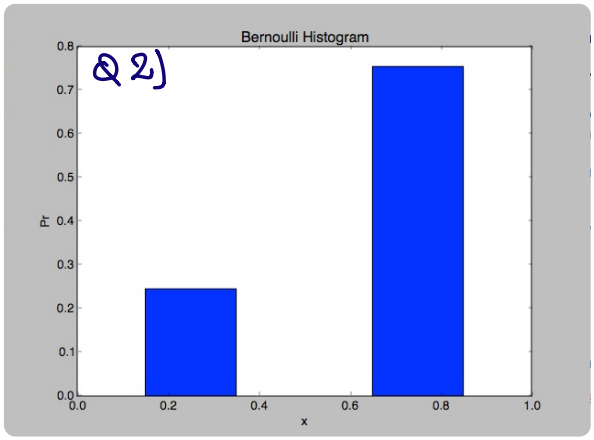
```
#############################################################################
#Done 21
def fake_hash_exp(num_zeros):
    b1 = 'wubba lubba'
    b2 = 'dub dub'
    r = random.randint(0,pow(2,256)-1) # range from 0 to [2^256 -1]
    # s -> 'int', convert this to binary. [DONE]
    # check leading zero from this with [num_zeros]
    hashResult = hashlib.sha256(b1+b2+str(r)).hexdigest()
    string = "1" + hashResult
    binary = str("{0:020b}".format(int(string,16)))
    #print(binary)
    #print(binary[1:num_zeros+1])

    if ( int(binary[1:num_zeros+1]) ) == 0:
        #print("Goes In")
        return 1
    else:
        return 0

#############################################################################
# Done 21 and 22
def run_fake_hash_exp(num_zeros):
    arr = []
    counter = 0
    v2 = []
    for i in range(0,500000):
        arr += [fake_hash_exp(num_zeros)]
        if arr[i] == 0:
            counter +=1
        else:
            v2 += [counter]
            counter = 0
    return(arr,v2)
```

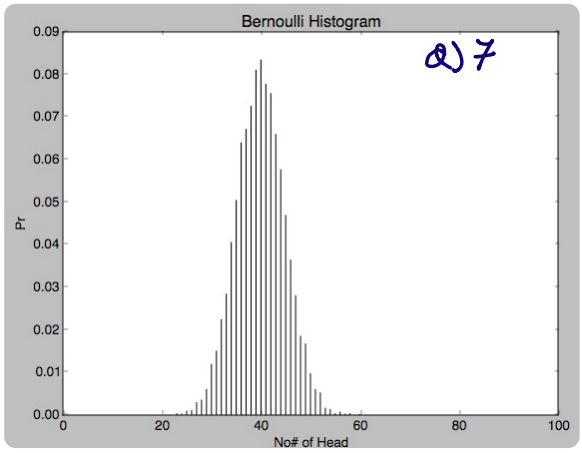
Q23) Code Submitted & plot Submitted

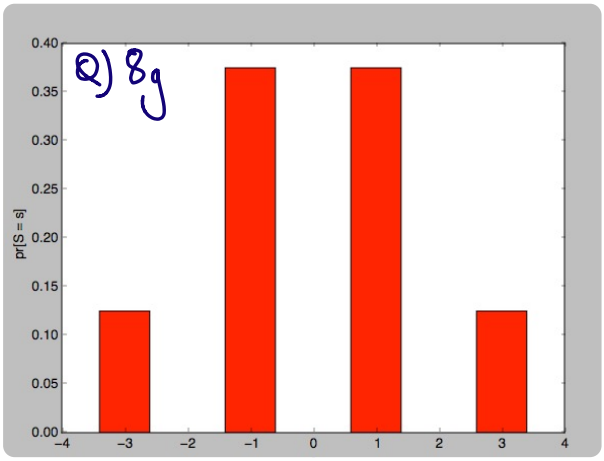
```
#################################################################
#Done 23
def histogram(num_zeros):
    (arr,v2) = run_fake_hash_exp(num_zeros)
    print(v2)
    plt.figure(1)
    plt.hist(v2,bins = 20,rwidth=.4,align ='mid',
             weights = np.zeros_like(v2)+1./len(v2))
    plt.xlim(0,200)
    plt.title("Histogram")
    plt.xlabel("x")
    plt.ylabel("Pr")
    plt.show()
```

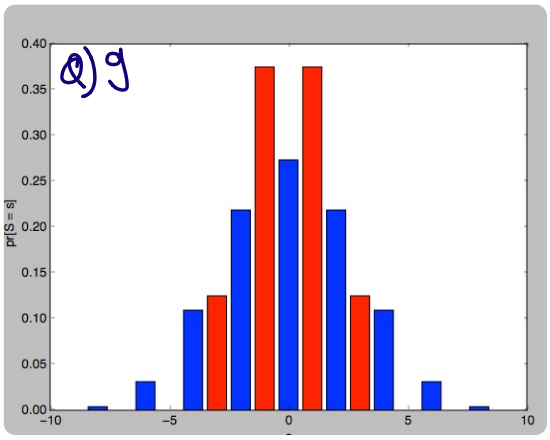
Q24) Code Submitted & plot Submitted

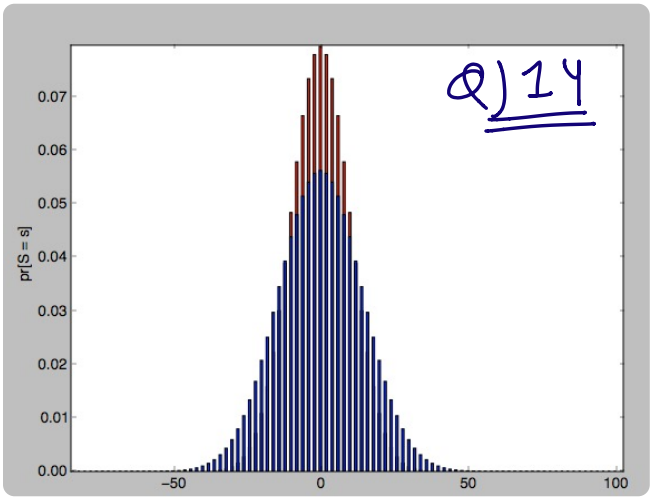
```
#################################################################################
# 24
def run_fake_10min(num_zeros):
    (arr,v2) = run_fake_hash_exp(num_zeros)
    n = 0
    v3 = []
    # slicing 1000 elements and summing it to simulate 10 minutes.
    # v3 = [sum(arr[0:1001]) ,sum(arr[1000:2001]),.....,sum(arr[n,(2*n)+1])]
    for j in range(0,500):
        v3 += [sum(arr[n:n+1001])]
        n += 1000
    return(v3)
#================================================================================
def histogram_1000trials(num_zeros):
    (v3) = run_fake_10min(num_zeros)
    print(v3)
    plt.figure(1)
    plt.hist(v3,bins = 1000,rwidth=.4,align ='mid',
             weights = np.zeros_like(v3)+1./len(v3))
    plt.xlim(0,35)
    plt.title("Histogram")
    plt.xlabel("x")
    plt.ylabel("Pr")
    plt.show()
#################################################################################
```

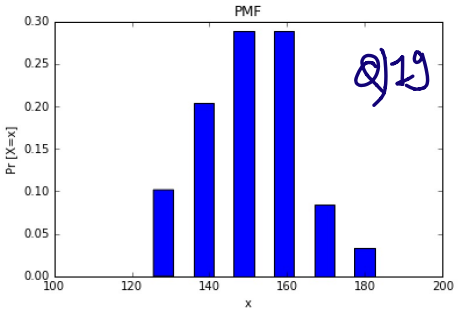

Bernoulli Histogram
Q 2)
Pr
x

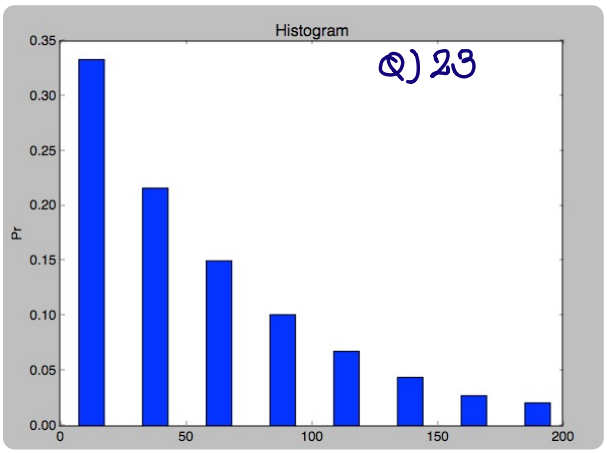

Bernoulli Histogram
Q) 7
Pr
No# of Head

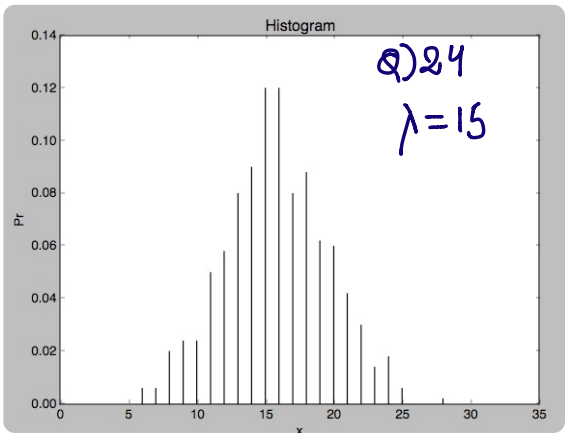

Q) 8)
pr[S = s]

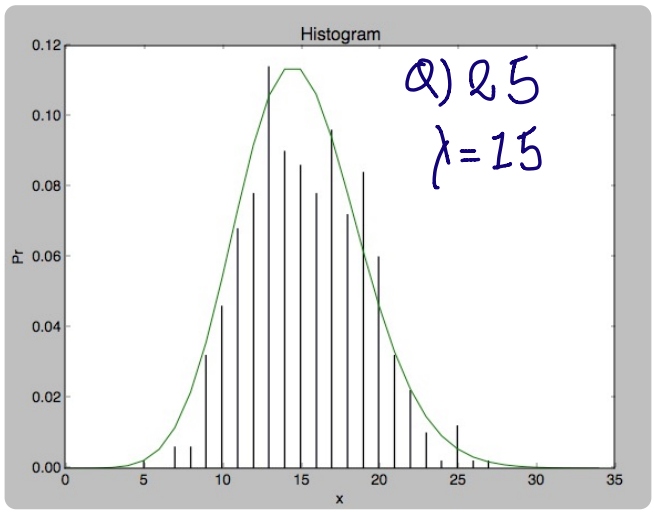

Q) 9
pr[S = s]


Q) 14
pr[S = s]


PMF
Q) 19
Pr [X=x]
x


Histogram
Q) 23
Pr
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
50
100
150
200


Histogram
Q) 24
λ = 15
Pr
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
25
30
35
x


Histogram
Q) 25
λ = 15
Pr
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
25
30
35
x