Problem Set 05

Several problems in this section will call for you to do nontrivial calculation. It is strongly recommended, but not required, that you write Python programs. The scipy.stats distribution objects may be useful in that regard.
NOTE: For any question that asks to "show (or give) a procedure for ...", you may write a Python function, but you are not required to do so.

1. [20 pts] Derive a formula and explain how to generate a random variable

with the density f (x) = (1.5) $\sqrt{x}$ 0 < x < 1
if your random number generator produces a Standard Uniform random variable U. Use the inverse transform method. What sample is generated if U = 0.001?

The cdf of f(x) is F(x) = $\int_0^x 1.5\sqrt{t}\,dt \ = \ x^{\frac{3}{2}}$

$$F^{-1}(x) \ = \ x^{\frac{2}{3}}$$

With u = 0.001:

$F^{-1}(x) \ = \ 0.001^{\frac{2}{3}}$ = 0.01

This value 0.01 is the sample generated from u = 0.001.

2. [20 pts] For the following random variables, use either a bucket method or the

inverse transform method to **generate random samples from the specified distribution**. Your source of randomness is a Standard Uniform distribution.
Show all the steps required to generate all of the following distributions from Standard Uniform.

2.1.[4 pts] an Exponential random variable with the parameter λ = 2.5

This is a continuous distribution thus we choose inversed transform method to generate samples.

The cdf of exponential distribution is:

F(x) = $1 - e^{-\lambda x}$

$$F^{-1}(x) = -\frac{\ln(1-x)}{\lambda}$$

1/ We generate u value from Uniform U(0,1) using python

```
u = st.uniform().rvs()
```

2/ Then we plug this u seed value into the inverse of cdf to get back a value which is x, a sample generated from the exponential distribution.

```
lam = 2.5
-mth.log(1 - u)/lam
```

Example:

u = 0.6699897902014875

x = 0.44345267448673364

2.2. [4 pts] a Bernoulli random variable with the probability of success 0.77

This is a discrete distribution thus we choose buckets method to generate samples.

1/ We generate u value from Uniform U(0,1) using python

```
u = st.uniform().rvs()
```

2/ We assign this value into buckets to distribute among the probabilities.

```
1 if u <= 0.77 else 0
```

Example:

u = 0.650294746702524

x = 1

2.3. [4 pts] a Binomial random variable with parameters n = 15 and p = 0.4

This is a discrete distribution thus we choose buckets method to generate samples. This binomial distribution has n = 15 and p = 0.4 meaning each rv is a sum of 15 bernoullis each has p =0.4.

1/ We generate u value from Uniform U(0,1) using python.

2/ We assign this value into buckets to distribute among the probabilities. Repeat 15 times

3/ Sum of all the 1s among these 15 Bernoulli's is one single value for binomial distribution.

```
import scipy.stats as st

sum = 0
for index in range(15):
    u = st.uniform().rvs()
    x = 1 if u <= 0.4 else 0
    sum += x
print(sum)
```

Example: x = 7

2.4. [4 pts] a discrete random variable with the distribution P(x), where:

P(0) = 0.2,

P(2) = 0.4,

P(7) = 0.3,

P(11) = 0.1

This is a discrete distribution thus we choose buckets method to generate samples. These buckets according to the distribution above are:

A1 [0, 0.2)

A2 [0.2, 0.6)

A3 [0.6, 0.9)

A4 [0.9, 1)

1/ We generate u value from Uniform U(0,1) using python

2/ We assign this value into buckets to distribute among the probabilities.

```
u = st.uniform().rvs()
print (u)
x = 0
if u <= 0.2:
    x = 0
elif u <= 0.5:
    x = 2
elif u <= 0.75:
    x = 7
else:
```

```
    x = 11
print(x)
```

Example:

u = 0.7144472497439565

x = 7

2.5. [4 pts} a continuous random variable with the density

f (x) = 3x^2 0 < x < 1

This is a continuous distribution thus we choose inversed transform method to generate samples.

The cdf of this distribution is:

F(x) = $\int_0^x t^3 \, dt \; = \; x^3$

$$F^{-1}(x) \; = \; x^{\frac{1}{3}}$$

1/ We generate u value from Uniform U(0,1) using python

```
u = st.uniform().rvs()
```

2/ Then we plug this u seed value into the inverse of cdf to get back a value which is x, a sample generated from the exponential distribution.

```
print (u**1/3)
```

u = 0.5635609631425434

x = 0.1878536543808478

3. [20 pt] Explain how to generate a random variable X that has a pdf

F(x) = 1/2 (1 + x)   −1 ≤ x ≤ 1

F(x) = 0 otherwise

You should use the inverse transform method. Give the details of your method.

This is a continuous distribution thus we choose inversed transform method to generate samples.

The cdf of this distribution is:

F(x) =

$\int_{-1}^{x} \frac{1}{2}(1+t)dt \; = \; \frac{1}{2}\left(\frac{x^2}{2} + x + C\right)$ for t from –1 to x

= $\frac{x^2}{4} + \frac{x}{2} + \frac{1}{4}$ = y
$x^2 + 2x + 1 - 4y$ = 0
Because x > -1 so:
Y= $(-2 \; + \; \sqrt{4} - 4 \cdot (1 - 4y)$) / 2*1 = -1+$\sqrt{4x}$

Y = (-2 + $\sqrt{4 - 4(1 - 4x)}$)/2 = -1+$\sqrt{4x}$ = -1 +2$\sqrt{x}$

1/ We generate u value from Uniform U(0,1) using python

2/ Then we plug this u seed value into the inverse of cdf to get back a value which is x, a sample generated from the exponential distribution.

```
import math
u = st.uniform().rvs()
print (u)
x = -1 + ( math.sqrt(4*u) )
print (x)
```

Example:
u = 0.9999443613943394

x = 0.2360182122642378

4. [20 pts] Explain how to estimate the following probability:

P {X>Y}, where X and Y are independent Poisson random variables with parameters 3 and 5, respectively.
You may assume that you can generate samples from any Poisson distribution.

This is a Poisson discrete distribution thus we choose buckets method to generate samples for P {X>Y}.

1/ We generate u value from Poison P(3) using python
2/ We generate u value from Poison P(5) using python
3/ We assign this value 1 if X>Y else 0. Repeat 100 times (100 or any number which is arbitrary large for more precision)

4/ Find sum of all the 1s and divide by the number of pairs then we get the probability for P {X>Y}

```
sum = 0
for index in range(100):
    x = st.poisson(3).rvs()
    y = st.poisson(5).rvs()
    if x>y:
        sum += 1
print(sum/100)
```

Example : x = 0.19

5. [20 pts] Biased and unbiased coin simulations

5.1. [10 pt]. Describe a procedure that uses a fair coin (50% heads, 50% tails) to simulate a biased coin with 75% heads, 25% tails.

1/ We generate u value from flipping two coins:
HT TH HH TT

p(HT) = p(TH) = p(HH) = p(TT) = 0.25

p(at least 1 T) = 0.75

p(HH) = 0.25

2/ Observe that the probability of at least 1 tail is 0.75 and probability of 2 heads is 0.25, we assign these values into buckets to distribute among the probabilities.
If there are two Heads then x = T else x = H

```
import random
sample = ("HT", "TH", "HH", "TT")
u = random.choice(sample)
if u == "HH":
    x = T
else:
    x = H
print (x)
```

Example: x = H

5.2. [10 pt]. Describe a procedure that emulates a fair coin from the biased coin simulation from 5.1

1/ Let p is probability of getting a Head:

p(HT) = p(TH) =p*q = q*p

p(HH) = p*p

p(TT) = q*q

2/ If you get TT, HH then discard else X= u[0] meaning always take the first coin's result

```
import random
sample = ("HT", "TH", "HH", "TT")
u = random.choice(sample)
if u == "HT" or u == "TH":
    x = u[0]
print (x)
```

Example: x = H