SOLUTIONS 1

Solution 1.1. The solution for inversion for discrete distribution.

```
import numpy as np
import matplotlib.pyplot as plt

n = 10000

p = np.array([0.2, 0.3, 0.2, 0.1, 0.2]) # probability mass function
s = np.array([0, 1, 2, 3, 4])           # support

cdf = np.cumsum(p) # compute CDF

samples = []

for i in range(n):
    u = np.random.uniform(0, 1)

    # find the first element of cdf that is greater than u
    # this is the index of the state that we will sample
    for k in range(len(cdf)):
        if cdf[k] > u:
            samples.append(s[k])
            # the first time if holds, we break out of the for loop
            break


# plot the pmf and histogram using the stem function
plt.stem(s, p, markerfmt='o', linefmt='k-')
plt.hist(samples, range(6), density=True, rwidth=0.4, color='r', alpha
                                    =0.5, align='left')
# Above function aligns the histogram with the PMF.
plt.title("PMF and the histogram")
plt.xlabel("s")
plt.show()
```

Solution 1.2. Sampler for the exponential distribution

```
import numpy as np
import matplotlib.pyplot as plt

# sample from exponential distribution with inversion

n = 10000 # number of samples

# parameters of the exponential distribution
lam = 1

samples = np.array([]) # list to store samples

for i in range(n):
    u = np.random.uniform(0, 1) # sample from uniform distribution
    x = -(1/lam) * np.log(1 - u) # inverse of the CDF
    samples = np.append(samples, x) # add the sample to the list

# plot the histogram of the samples and the density
x = np.linspace(0, 10, 1000)
y = lam * np.exp(-lam * x)
```

```
plt.hist(samples, bins=100, density=True, rwidth=0.8, color='r', alpha
                                        =0.5)
plt.plot(x, y, 'k-')
plt.title("Histogram of samples and the density")
plt.xlabel("x")
plt.xlim([0, 10])
plt.show()
```

Solution 1.3. Sampling from Gaussian using just uniforms (Box-Muller)

```
import numpy as np
import matplotlib.pyplot as plt

# sample from Gaussian using uniforms

n = 10000 # number of samples

x = np.array([]) # list to store samples
y = np.array([]) # list to store samples

for i in range(n):
    u1 = np.random.uniform(0, 1)
    u2 = np.random.uniform(0, 1)
    A = 2 * np.pi * u1
    R = np.sqrt(-2 * np.log(u2))
    x = np.append(x, R * np.cos(A))
    y = np.append(y, R * np.sin(A))

# plot the histogram of the samples and the density
xx = np.linspace(-5, 5, 1000)
gauss_density = 1/np.sqrt(2 * np.pi) * np.exp(-xx**2 / 2)
fig, axs = plt.subplots(1, 2, figsize=(14, 7))
axs[0].hist(x, bins=100, density=True, rwidth=0.8, color='r', alpha=0.
                                        5)
axs[0].plot(xx, gauss_density, 'k-')
axs[0].set_title("Histogram of samples and the density")
axs[0].set_xlabel("x")
axs[0].set_xlim([-5, 5])
axs[1].hist(y, bins=100, density=True, rwidth=0.8, color='r', alpha=0.
                                        5)
axs[1].plot(xx, gauss_density, 'k-')
axs[1].set_title("Histogram of samples and the density")
axs[1].set_xlabel("y")
axs[1].set_xlim([-5, 5])
plt.show()
```

Solution 1.4. The solution regarding transformation of r.v.s is given in the slides of Lecture 2. The code is provided below.

```
# Assuming x samples generated as in the previous exercise, cont. code
mu = 2
sigma = 3

Z = mu + sigma * x

# plot the histogram of the samples and the density
xx = np.linspace(-7.5, 12.5, 10000)
```

```
gauss_density = 1/(np.sqrt(2 * np.pi) * sigma) * np.exp(-(xx-mu)**2 /
                                        (2 * sigma**2))
plt.hist(Z, bins=100, density=True, rwidth=0.8, color='r', alpha=0.5)
plt.plot(xx, gauss_density, 'k-')
plt.title("Histogram of samples and the density")
plt.xlabel("Z")
plt.xlim([-7.5, 12.5])
plt.show()
```

Solution 1.5. $X \sim \text{Exp}(1)$ and $W = \alpha X^{1/\beta}$.

$$\begin{aligned} F_W(w) &= \text{P}(W \leq w) = P(\alpha X^{\frac{1}{\beta}} \leq w) \\ &= P\left(X \leq \left(\frac{w}{\alpha}\right)^{\beta}\right) = 1 - \exp\left[-\left(\frac{w}{\alpha}\right)^{\beta}\right] \\ \Rightarrow f_W(w) &= \beta\alpha^{-\beta} w^{\beta-1} \exp\left[-\left(\frac{w}{\alpha}\right)^{\beta}\right]. \end{aligned}$$

Algorithm for Weibull:

1. Generate $U \sim U(0, 1)$.
2. Set $X = -\log U$ (so X exponential).
3. Set $W = \alpha X^{\frac{1}{\beta}}$ (so W is Weibull).

The code is provided below:

```
import numpy as np
import matplotlib.pyplot as plt

n = 20000

# sample from weibull
def weibull_density(w, a, b):
    return b * a**(-b) * w**(b-1) * np.exp(- (w / a)**b)

a = 1
b = 2

samples = np.array([]) # list to store samples

for i in range(n):
    u = np.random.uniform(0, 1)
    x = - np.log(1 - u)
    w = a * x**(1/b)
    samples = np.append(samples, w)

xx = np.linspace(0, 4, 1000)
yy = weibull_density(xx, a, b)
plt.plot(xx, yy, 'k-')
plt.hist(samples, bins=100, density=True, rwidth=0.8, color='r', alpha
                                        =0.5)
plt.title("Weibull density and histogram")
plt.xlabel("x")
plt.xlim([0, 4])
plt.show()
```