Assignment #1 – Satish Ramachandran

Problem 1:

Execution in lazy mode:

```
'''
Assignment 1 - Problem #1
Executing tensor addition in lazy(default) mode
'''
import tensorflow as tf

with tf.compat.v1.Session() as sess:
   x = tf.constant([100, 101, 102, 103, 104, 105, 106, 107, 108, 109])
   y = tf.constant([34, 28, 45, 67, 89, 93, 24, 49, 11, 7])

   z = tf.add(x,y)

   result = sess.run(z)
   print(result)
```

Execution in eager mode:

NOTE: I'm using TensorFlow 2.0. Eager mode is enabled by default in TF 2.0

```
'''
Assignment 1 - Problem #1(b)
Executing tensor addition in eager mode
'''
import tensorflow as tf

# In Tensorflow 2, the eager execution is enabled by default

x = tf.constant([100, 101, 102, 103, 104, 105, 106, 107, 108, 109])
y = tf.constant([34, 28, 45, 67, 89, 93, 24, 49, 11, 7])
z = tf.add(x,y)
print(z)
```

Problem 2:

```
'''
Assignment1 - Problem 2
'''
import tensorflow as tf
x1 = tf.constant([[1,2,3,4],[5,6,7,8]])
print(x1)
```

```
stackedx1 = tf.stack([x1, x1, x1, x1])
print(stackedx1)
```

```
tf.Tensor(
[[1 2 3 4]
 [5 6 7 8]], shape=(2, 4), dtype=int32)
tf.Tensor(
[[[1 2 3 4]
  [5 6 7 8]]

 [[1 2 3 4]
  [5 6 7 8]]

 [[1 2 3 4]
  [5 6 7 8]]

 [[1 2 3 4]
  [5 6 7 8]]], shape=(4, 2, 4), dtype=int32)
```

Problem 3:

```
'''
Assignment1 - Problem 3
'''
import tensorflow as tf
x1 = tf.constant([[1,2,3,4],[5,6,7,8]])
print(x1)
stackedx1 = tf.stack([x1])
print(stackedx1)
```

```
tf.Tensor(
[[1 2 3 4]
 [5 6 7 8]], shape=(2, 4), dtype=int32)
tf.Tensor(
[[[1 2 3 4]
  [5 6 7 8]]], shape=(1, 2, 4), dtype=int32)
```

Problem 4:

```
'''
Assignment1 - Problem 4
'''
import tensorflow as tf
x1 = tf.constant([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print(x1)
```

```
reshapedx1 = tf.reshape(x1, [6,2], 'reshaped-x1')
print(reshapedx1)
```

```
tf.Tensor(
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]], shape=(3, 4), dtype=int32)
tf.Tensor(
[[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]
 [11 12]], shape=(6, 2), dtype=int32)
```

Problem 5:

```
'''
Assignment1 - Problem 5
'''
import tensorflow as tf

a=tf.Variable(1.12)
b=tf.Variable(2.34)
c=tf.Variable(0.72)
d=tf.Variable(0.81)
f=tf.Variable(19.83)

x = 1 + (a/b) + (c/pow(f,2))
s = (b-a)/(d-c)
r = 1/((1/a) + (1/b) + (1/c) + (1/d))
y = a * b * (1/c) * (pow(f,2)/2)

print(x)
print(s)
print(r)
print(y)
```

```
tf.Tensor(1.4804634, shape=(), dtype=float32)
tf.Tensor(13.555558, shape=(), dtype=float32)
tf.Tensor(0.25357127, shape=(), dtype=float32)
tf.Tensor(715.6765, shape=(), dtype=float32)
```

Problem 6:

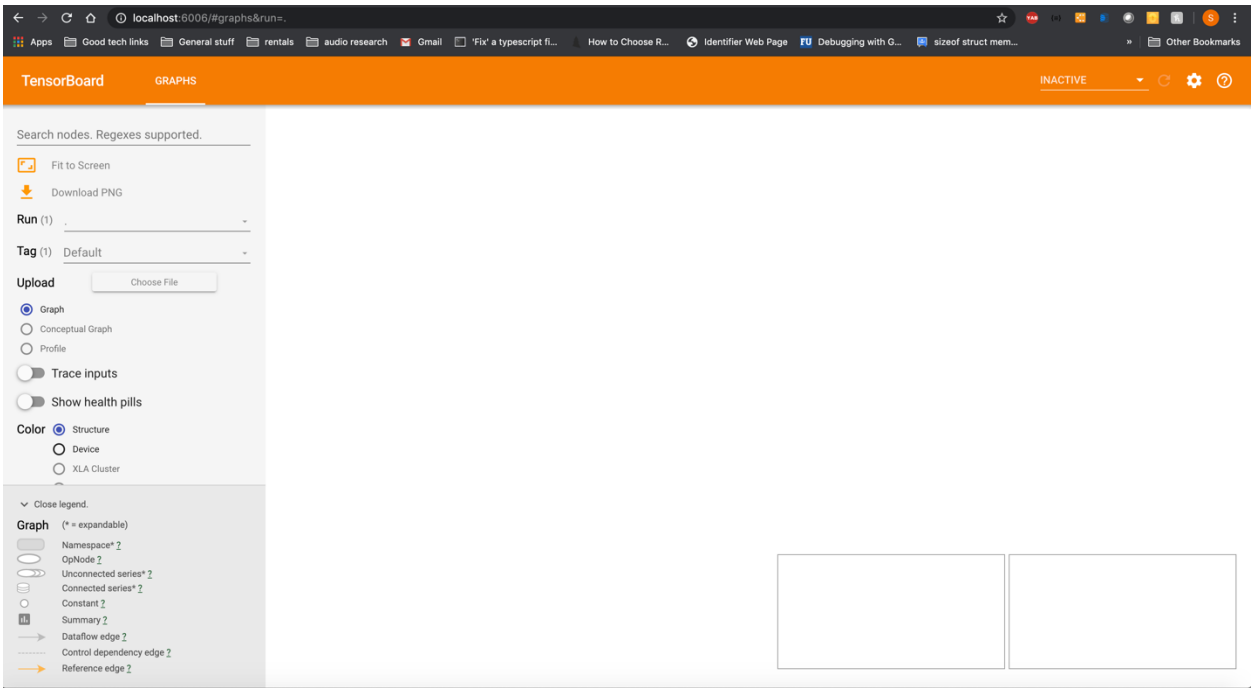
```
'''
Assignment1 - Problem 6
'''
import tensorflow as tf

with tf.compat.v1.Session() as sess:
   x = tf.constant([100, 101, 102, 103, 104, 105, 106, 107, 108, 109])
   y = tf.constant([34, 28, 45, 67, 89, 93, 24, 49, 11, 7])
   writer = tf.compat.v1.summary.FileWriter('./summaries2', sess.graph)
   z = tf.add(x,y)
   res = sess.run(z)
   writer.flush()
```

2020-04-05 19:14:08.195221: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2020-04-05 19:14:08.213211: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x7ffbc97e7970 initialized for platform Host (this does not guarantee that XLA will be used). Devices:
2020-04-05 19:14:08.213232: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Host, Default Version

The graph is not showing up. But, I do see the file in the summaries2 directory.
Need to debug this further.

```
(base) satishramac-a01:Week1 satishramach$ ls -lt ./summaries2/
total 8
-rw-r--r--  1 satishramach  staff  689 Apr  5 19:14 events.out.tfevents.1586139248.satishramac-a01
.vmware.com
(base) satishramac-a01:Week1 satishramach$
```



Problem 7:

```
'''
Assignment1 - Problem 7
'''
import tensorflow as tf

A = tf.constant([[4, -2, 1],[6, 8, -5],[7, 9, 10]])
B = tf.constant([[6, 9, -4],[7, 5, 3],[-8, 2, 1]])
C = tf.constant([[-4, -5, 2],[10, 6, 1],[3, -9, 8]])

A1 = A * (B + C)
A2 = (A * B) + (A * C)

print(A1)
print(A2)

D1 = A * (B * C)
D2 = (A * B) * C

print(D1)
print(D2)

print(tf.equal(A1, A2))
```

```
# tf.equal() returns a tensor of the same size. But, to compare the
# values, we can reduce the tensor across all dimensions
if tf.reduce_all(tf.equal(A1, A2)):
    print('Associative property validated')

if tf.reduce_all(tf.equal(D1, D2)):
    print('Distributive property validated')
```

```
tf.Tensor(
[[  8  -8  -2]
 [102  88 -20]
 [-35 -63  90]], shape=(3, 3), dtype=int32)
tf.Tensor(
[[  8  -8  -2]
 [102  88 -20]
 [-35 -63  90]], shape=(3, 3), dtype=int32)
tf.Tensor(
[[ -96   90   -8]
 [ 420  240  -15]
 [-168 -162   80]], shape=(3, 3), dtype=int32)
tf.Tensor(
[[ -96   90   -8]
 [ 420  240  -15]
 [-168 -162   80]], shape=(3, 3), dtype=int32)
tf.Tensor(
[[ True  True  True]
 [ True  True  True]
 [ True  True  True]], shape=(3, 3), dtype=bool)
Associative property validated
Distributive property validated
```