ASS 2

```
# 1. Import necessary libraries
import tensorflow as tf
from keras.models import Sequential
from keras.datasets import mnist
from keras import layers
import matplotlib.pyplot as plt
import numpy as np
import random

 2. Load MNIST dataset

(x_train, y_train), (x_test, y_test) = mnist.load_data()
len(x_train)
len(x_test)
len(y_train)
len(y_test)
x_train.shape

# 2. Normalize data (convert pixel values 0–255 → 0–1)
x_train = x_train / 255.0
x_test = x_test / 255.0

# 3. Define the feedforward network architecture Define the network architecture using Keras
model = Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))       # Flatten 28x28 → 784
model.add(layers.Dense(128, activation='relu'))       # Hidden layer
model.add(layers.Dense(10, activation='softmax'))     # Output for 10 digits
model.summary()

# 4.A. Compile the model
model.compile(optimizer='sgd',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 4.B. Train the model
history = model.fit(x_train, y_train,
                    validation_data=(x_test, y_test),
                    epochs=10,
                    verbose=1)

# 5.A. Evaluate the model
test_loss,test_acc=model.evaluate(x_test,y_test)
print("Final Test Loss=%.3f" %test_loss)
print("Final Test Accuracy=%.3f" %test_acc)

# 5.B. Predict on a random test image
n = random.randint(0, 9999)

plt.imshow(x_test[n])
plt.show()

plt.imshow(x_test[n], cmap='gray')
plt.title("Test Image")
plt.axis('off')
plt.show()

predictions = model.predict(x_test)
print("Predicted Number:", np.argmax(predictions[n]))

# 6.A Plot training loss and accuracy
plt.figure(figsize=(12, 5))

# Plot training vs validation accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Test Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Plot training vs validation loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Test Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()

## 6.A Plot training loss and accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Training Loss and accuracy')
plt.ylabel('accuracy/Loss')
plt.xlabel('epoch')
plt.legend(['accuracy', 'val_accuracy','loss','val_loss'])
plt.show()
```