```python
# Assignment No 6:-

import tensorflow as tf            # For building and training neural networks
import numpy as np                 # For numerical operations

# ✅ Download Shakespeare dataset and truncate to first 100,000 characters to save RAM
path = tf.keras.utils.get_file('shakespeare.txt',
    'https://storage.googleapis.com/download.tensorflow.org/data/shakespeare.txt')
text = open(path, 'r', encoding='utf-8').read()[:100_000]

# ✅ Tokenize words: turn text into sequence of integers (1 index per word)
tok = tf.keras.preprocessing.text.Tokenizer()
tok.fit_on_texts([text])                # Build word-index mapping
seq = tok.texts_to_sequences([text])[0]  # Convert full text to a list of word indices
vocab = len(tok.word_index) + 1         # Total number of unique words (+1 for padding index)

# ✅ Create training data (input sequences and next word as label)
seq_len = 10                            # Number of words to predict next word
X, y = [], []
for i in range(len(seq) - seq_len):
    X.append(seq[i:i+seq_len])          # Input: 10-word sequence
    y.append(seq[i+seq_len])            # Target: next word after the sequence

# ✅ Use only first 10,000 samples to avoid memory crash
X = np.array(X[:10000])                 # Shape: (10000, 10)
y = np.array(y[:10000])                 # Shape: (10000,)
y = tf.keras.utils.to_categorical(y, num_classes=vocab)  # One-hot encode y for softmax

# ✅ Build small LSTM model to learn word sequence patterns
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab, 32, input_length=seq_len),  # Embeds word indices to vectors
    tf.keras.layers.LSTM(64),                                    # Processes sequences
    tf.keras.layers.Dense(vocab, activation='softmax')           # Predict next word
])

# ✅ Compile the model with crossentropy loss and Adam optimizer
model.compile(loss='categorical_crossentropy', optimizer='adam')

# ✅ Train for 5 epochs on batch of 128
model.fit(X, y, epochs=5, batch_size=128)

# ✅ Function to generate new text from a starting word using temperature sampling
def generate(start="BRUTUS", length=50, temp=0.7):
    inp = tok.texts_to_sequences([start])[0][-seq_len:]  # Get last seq_len words of start
    inp = [0] * (seq_len - len(inp)) + inp               # Pad start if it's shorter than seq_len
    inp = tf.expand_dims(inp, 0)                         # Add batch dimension: shape (1, 10)
    out = []                                             # To store generated words

    for _ in range(length):
        p = model(inp)[0].numpy().astype('float32')      # Get prediction from model
        p = np.exp(np.log(p + 1e-8) / temp)              # Apply temperature
        p /= np.sum(p)                                   # Normalize to make valid probability
        w_id = np.random.choice(vocab, p=p)              # Sample next word index
        word = tok.index_word.get(w_id, '')              # Convert index to word
        out.append(word)                                 # Add word to output
        inp = tf.expand_dims([*inp[0][1:], w_id], 0)      # Slide window forward

    return start + ' ' + ' '.join(out)                   # Combine seed and generated text

# ✅ Generate and print example output
print(generate())
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/shakespeare.txt
1115394/1115394 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just
  warnings.warn(
Epoch 1/5
79/79 ━━━━━━━━━━━━━━━━━━━━ 6s 6ms/step - loss: 7.7032
Epoch 2/5
79/79 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.3515
Epoch 3/5
79/79 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.3461
Epoch 4/5
79/79 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 6.2330
Epoch 5/5
```

```
79/79 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6.2090
BRUTUS i that army in arms of against the do spirit the superfluity his otherwise i beheld to state you that and up and you he marcius t
```

```
!cp $path /content/shakespeare.txt
```