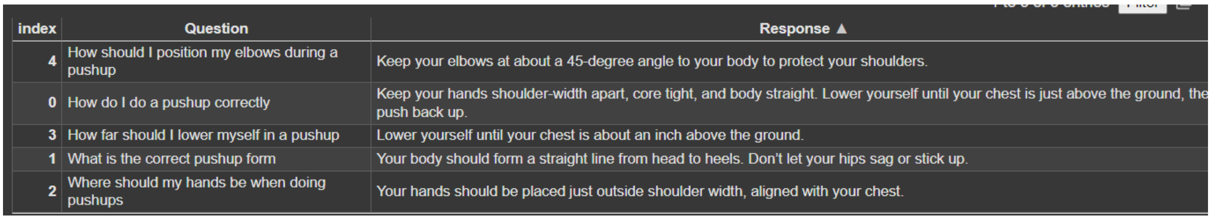
Required libraries:

```
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
from torch.utils.data import DataLoader, Dataset
import pandas as pd
import random
import re
```

Loading the dataset:

```
file_path = "/content/drive/My Drive/cleaned_fitness_dataset.xlsx"
df = pd.read_excel(file_path)
df.head()
```

Output:



index	Question	Response ▲
4	How should I position my elbows during a pushup	Keep your elbows at about a 45-degree angle to your body to protect your shoulders.
0	How do I do a pushup correctly	Keep your hands shoulder-width apart, core tight, and body straight. Lower yourself until your chest is just above the ground, the push back up.
3	How far should I lower myself in a pushup	Lower yourself until your chest is about an inch above the ground.
1	What is the correct pushup form	Your body should form a straight line from head to heels. Don't let your hips sag or stick up.
2	Where should my hands be when doing pushups	Your hands should be placed just outside shoulder width, aligned with your chest.

Cleaning the data , tokenization and building vocabulary:

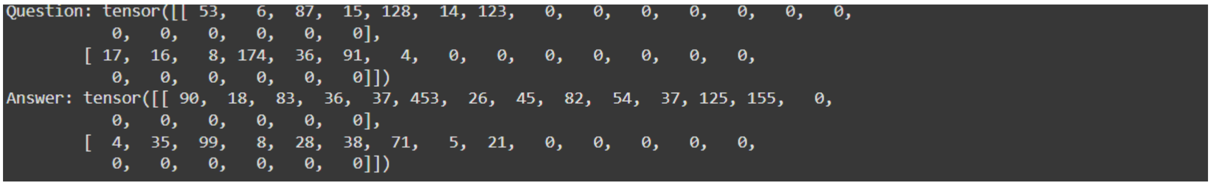
```
def clean_text(text):
    text = text.lower()
    text = re.sub(r"[^a-zA-Z\s]", "", text)
    text = re.sub(r"\s+", " ", text).strip()
    return text
def tokenize(text, vocab):
    return [vocab.get(word, vocab['<unk>']) for word in text.split()]
def build_vocab(data):
    word_count = Counter()
    for q, a in data:
        word_count.update(q.split())
        word_count.update(a.split())
    vocab = {word: idx for idx, (word, _) in enumerate(word_count.most_common(), start=4)}
    vocab['<pad>'] = 0
    vocab['<unk>'] = 1
    vocab['<sos>'] = 2
    vocab['<eos>'] = 3
    return vocab
```

```
cleaned_data = [(clean_text(q), clean_text(a)) for q, a in zip(df['Question'], df['Response'])]
vocab = build_vocab(cleaned_data)
tokenized_data = [(tokenize(q, vocab), tokenize(a, vocab)) for q, a in cleaned_data]
print(tokenized_data[:5])
```

Creating dataset class:

```
from torch.utils.data import Dataset, DataLoader
class QADataset(Dataset):
    def __init__(self, data, vocab, max_length=20):
        self.data = data
        self.vocab = vocab
        self.max_length = max_length
    def __len__(self):
        return len(self.data)
    def __getitem__(self, idx):
        question, answer = self.data[idx]

        question = question[:self.max_length] + [self.vocab['<pad>']] * (self.max_length - len(question))
        answer = answer[:self.max_length] + [self.vocab['<pad>']] * (self.max_length - len(answer))

        return torch.tensor(question), torch.tensor(answer)
train_dataset = QADataset(tokenized_data, vocab)
train_loader = DataLoader(train_dataset, batch_size=2, shuffle=True)
for question, answer in train_loader:
    print(f"Question: {question}")
    print(f"Answer: {answer}")
    break
```

Tensor output:



```
Question: tensor([[ 53,   6,  87,  15, 128,  14, 123,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0],
        [ 17,  16,   8, 174,  36,  91,   4,   0,   0,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0]])
Answer: tensor([[ 90,  18,  83,  36,  37, 453,  26,  45,  82,  54,  37, 125, 155,   0,
           0,   0,   0,   0,   0,   0],
        [  4,  35,  99,   8,  28,  38,  71,   5,  21,   0,   0,   0,   0,   0,
           0,   0,   0,   0,   0,   0]])
```

Encoder Model:

```
class Encoder(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim, num_layers=1):
        super(Encoder, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embedding_dim)
        self.rnn = nn.TransformerEncoderLayer(d_model=embedding_dim, nhead=8)
        self.fc_out = nn.Linear(embedding_dim, hidden_dim)
    def forward(self, src):
        embedded = self.embedding(src)
        embedded = embedded.permute(1, 0, 2)
        outputs = self.rnn(embedded)
        return outputs
```

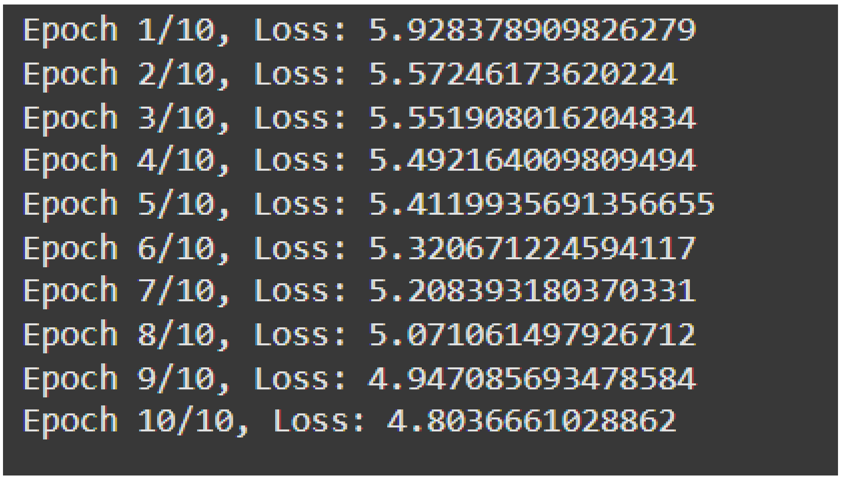
Initializing the Model, Optimizer, and Loss Function:

```
model = Seq2Seq(vocab_size=len(vocab), embedding_dim=128, hidden_dim=256)

optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
criterion = nn.CrossEntropyLoss(ignore_index=vocab['<pad>'])
```

Training the Model:

```
def train_model(model, train_loader, optimizer, criterion, epochs=10):
    model.train()
    for epoch in range(epochs):
        epoch_loss = 0
        for question, answer in train_loader:
            optimizer.zero_grad()

            output = model(question, answer[:, :-1])

            loss = criterion(output.reshape(-1, len(vocab)), answer[:, 1:].reshape(-1))
            loss.backward()
            optimizer.step()

            epoch_loss += loss.item()

        print(f'Epoch {epoch+1}/{epochs}, Loss: {epoch_loss / len(train_loader)}')

train_model(model, train_loader, optimizer, criterion, epochs=10)
```

Output:



```
Epoch 1/10, Loss: 5.928378909826279
Epoch 2/10, Loss: 5.57246173620224
Epoch 3/10, Loss: 5.551908016204834
Epoch 4/10, Loss: 5.492164009809494
Epoch 5/10, Loss: 5.4119935691356655
Epoch 6/10, Loss: 5.320671224594117
Epoch 7/10, Loss: 5.208393180370331
Epoch 8/10, Loss: 5.071061497926712
Epoch 9/10, Loss: 4.947085693478584
Epoch 10/10, Loss: 4.8036661028862
```

Basic dataset used in this code about pushup questions and responses:

https://docs.google.com/spreadsheets/d/1AcwT-eN5GErt4APCPzRekp9tVjV9rkrm/edit?usp=sharing&ouid=110569383377391677795&rtpof=true&sd=true

But this model is predicting wrong
The accuracy is too low, we need to work on BERT transformer model with more proper dataset.