Prerequisite: You have a positive file and a negative file with the same sequence length.

Step-1 : Create a new file that is going to merge the two files (positive and negative). Zipper merge the two files. Do it for all the files and create one big file (this is not a good approach, I have done it in a different way, but for that you have to have advance understanding of datloader and PyTorch dataclass. Where you will be calling it pairwise). Put a label in front of each row, 0 for negative 1 for positive.

Step-2 : Now create three different files: train file, test file and validation file. The train file should contain 75% of the rows of the big file. Test file 20% and validation file 5%. Each file should look something like this: Example
0 ATTTGGSGSG……..CCCTTT
0 GGGTT……..AAATTTTCCCC
1 GGGAAAAAA….AAAACCCC
0 AAATTTTTGG….CCCCCCC
1 AAAATTTTTT…..CCGGGGG

Step3 : Create a function called one_hot_encoder, Which is going to return a **numpy array** of one hot encoding of the sequence. If you convert your return type to a numpy array it will be easy to convert into a tensor at a later step.

Step4: Read from the file test.txt all the sequences. Pass the data part to the one_hot_encoding function. It will return an numpy array. In that array x[0] represent the label and the x[1] represent the data (one hot encoded data).

Step5: Create the data class that will inherit Dataset (look at the notebook in Moodle).

Step6: Remember your data is not in tensor format. Inside the class __init__ function, convert the data into a tensor by using the function torch.tensor(). Also use torch.stack(). It should be `self.data=torch.stack(torch.tensor(data))` [just an hint you may have to modify this code]

Step6: Convert the labels to a tensor too with torch.tensor().

Step7: Create __len__ and __getitem__ appropriately (see the Moodle code)

Step8: Now call this class that you just created.
For example, `test_data=data_class(one_hot_encoded_data, labels)`

Step9: Create a training data loader `train_dataloader=DataLoader(test_data,batch_size=512,shuffle=True)`

Step10: Create a for loop and print at least one label and one one_hot_encoding, and break from the loop.

```
for data,label in train_dataloader:
      print(data, label)
      break
```

If this is working everything should work.

Step11: now do it for the test and validation dataset.

This is a basic framework. If I give you any more I should be completing your projects.