The purpose of this assignment is to give you some experience building a tokenizer, and some practice thinking about the kind of math we'll use in the course.

Problem 1 (2 points) *Content from lecture 2*

One way to build a tokenizer is to break text into tokens at each whitespace, like

```
"Tokenization can be simple"

-> ["Tokenization", "can", "be", "simple"]
```

If we build a tokenizer from a document, its vocabulary includes all such words. So, our vocabulary here is simply

```
{"Tokenization":0, "can":1, "be":2, "simple":3, "UNK":4}
```

We can then use that tokenizer on a new document, and if a word shows up that isn't in our vocabulary, like `apple`, we can replace it with an "unknown" symbol `UNK`, and use token id 4 for it.

Some languages, like English, have relatively few conjugations for each verb; conjugation only communicates the number of the person, the tense, and the mood. For example, for the verb *to eat*, the conjugations are *eat eats, ate, (had/have) eaten*. Some languages, like Swahili, or Finnish, express much more information via conjugation, and as such, there are many more (e.g., hundreds in Swahili) conjugations. Explain briefly why this tokenization strategy works worse for those languages than for English.[1]

Problem 2 (2 points) *Content from lecture 2*
Tokenization can be somewhat unintuitive. Consider, for example, a tokenizer with the following properties:

```
{
 " Columbia" : 0,
 " columbia": 1,
 ...
}
```

This means whenever we see uncapitalized columbia, we get token 1, and when we see Columbia, we get token 0. These tokens are independent!

Imagine we train embeddings like those seen in Lecture 1 on such a vocabulary. Explain briefly (2-3 sentences) how the embeddings for token 0 may differ from the embeddings for token 1.

Problem 3 (5 points) *Content from lecture 1 and 3 (background)*
Demonstrate that the gradient of the softmax function with respect to an input index that receives (approaching) zero probability is (approaching) zero. Here's some notation to use: let $x \in \mathbb{R}^d$. For some fixed j, you're given that the output of $\text{softmax}(x)_j$ is very close to zero. Compute the gradient of the function. Then show that when $\text{softmax}(x)_j$ is close to zero, the gradient $\nabla_{x_j}\text{softmax}(x)_j$ is also close to zero. Formalize this closeness as you wish. Recall that

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{d}\exp(x_j)} \tag{1}$$

[1] You can call these languages "highly inflected". Look it up; it's an interesting linguistic topic!

Problem 4 (5 points) *Content from lecture 1 and 3 (background) and 4*
Identify a key limitation with the following neural model: let $w_{<t} = w_1, \ldots, w_{t-1}$ be the input sequence, with $w_i \in \mathcal{V}$. We embed the input sequence as $x_i = Ew_i$ to get embeddings $x_i \in \mathbb{R}^d$. We embed positions as $p_1, \ldots, p_{t-1}$, with $p_i \in \mathbb{R}^d$. Next, we nonlinearly combine embeddings and positions, as

$$h_i = \sigma(x_i + p_i) \qquad \forall_{i=1,\ldots,t-1} \tag{2}$$

We then compute similarities between the last embedding and all previous embeddings:

$$a_{i,t-1} = x_i^\top x_{t-1} \qquad \forall_{i=1,\ldots,t-1} \tag{3}$$

we then compute our representation as a weighted sum of all embeddings so far:

$$\tilde{h}_{t-1} = \sum_{i=1}^{t-1} a_{i,t-1} h_i \tag{4}$$

This limitation will be one of *expressivity*, like we discussed in our classes. It's not that order is not considered, and it's not that tokens cannot interact. Describe the limitation concisely.

Problem 5 (10 points) *Content from lecture 2*
Complete the tokenizer code provided in `a0.ipynb`, and upload to gradescope.

Problem 6 (1 points) *Content from lecture 2*
In the tokenizer you trained, when you tokenize the snowman emoji, in the line marked `Individual 'decoded' token strings:`, what do the snowman tokens look like? Why?