1. Language models 101 [14 pt]

a. [4 pt] Give both a short informal definition of a language model using natural language and a more formal definition using additional mathematical notation.

b. [4 pt] Which, if any, of the following expressions describes a commonly used objective for training causal language models?

	expression	yes, is common	no, not common
1	$argmin_\theta \frac{1}{n} \sum - \log P_\theta(w_i \mid w_{1:i-1})$	☐	☐
2	$argmin_\theta \frac{1}{n} \sum (\hat{w} - w_i)^2$	☐	☐
3	$argmin_\theta \log \left(\prod_{i=1}^{n} P_\theta(w_i \mid w_{1:i-1}) \right)$	☐	☐
4	$argmin_\theta \frac{1}{n} \prod (\hat{w} - w_i)^2$	☐	☐

c. [6 pt] Describe succinctly the difference between the following pairs of types of language models:

(i) masked vs. causal:

(ii) causal vs. autoregressive:

(iii) bidirectional vs left-to-right:

2. RNNs, LSTMs & Transformers [16 pt]

a. [4 pt] Describe in informal, short but precise terms a key conceptual difference between the following types of language model architectures:

(i) RNN vs. LSTM:

(ii) LSTM vs. transformer:

b. [6 pt] Name and describe (in short, informal but precise terms) one important technical problem that arises when training RNNs for large sequences of tokens (which transformers do not have).

c. [6 pt] Consider the following image of a transformer architecture. Fill in the names of the components which are currently blanked out. What kind of architecture is it (encoder, decoder, encoder-decoder)?

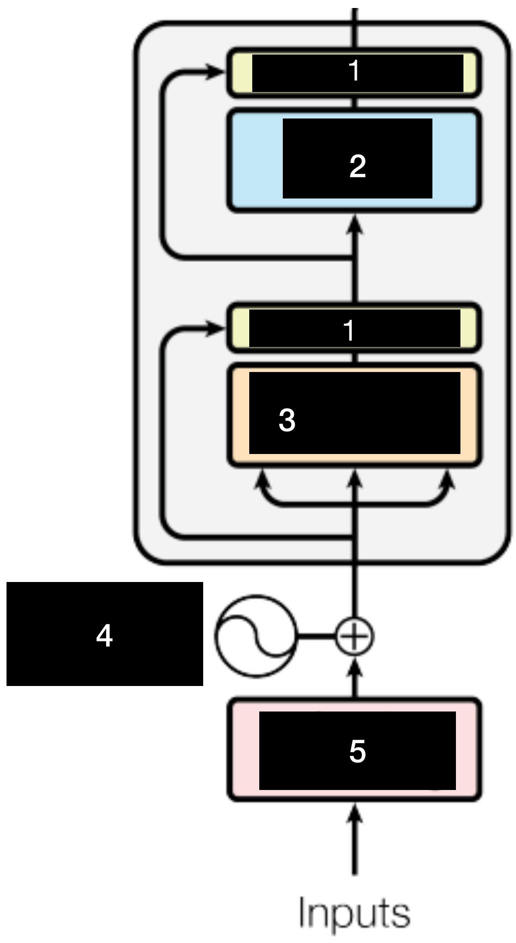
1 ______________________________

2 ______________________________

3 ______________________________

4 ______________________________

5 ______________________________



Encoder, decoder, encoder-decoder?

____

3. ANNs, implementations & PyTorch [8 pt]

a. [4 pt] Oh no! Some genius scrambled the steps of the back-propagation training loop. Fix it by writing the correct consecutive number next to each row:

```
    opt.step()
    prediction = torch.distributions.Normal(loc=location, scale=1.0)
n_training_steps = 10000
    loss = -torch.sum(prediction.log_prob(train_data))
for i in range(n_training_steps):
    opt.zero_grad()
    loss.backward()
```

b. [4 pt] Here is code to predict the next token from an RNN (as we used in class to predict surnames for a given country). Which decoding scheme is implemented here? Mark the lines that you would most likely want to change to implement a different decoding scheme.

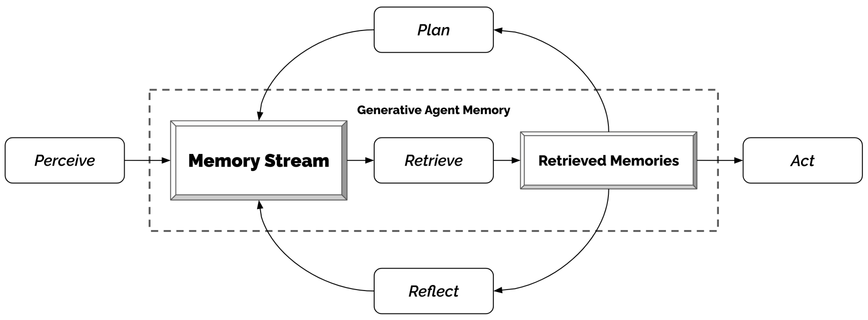
```
def predict(category, initial_sequence):

    if len(initial_sequence) >= max_length:
        return initial_sequence

    category_tensor_ = category_tensor(category)
    input_line_tensor = input_tensor(initial_sequence)
    hidden = rnn.init_hidden()

    name = initial_sequence

    for i in range(input_line_tensor.size(0)):
        with torch.no_grad():
            output, hidden = rnn(
                category_tensor_,
                input_line_tensor[i],
                hidden,
                False)

    topv, topi = output.topk(1)
    topi = topi[0][0]

    if topi == EOSIndex:
        return name
    else:
        name += all_letters[topi]

    return predict(category, name)
```

4. LM-based applications [14 pt]

a. [4 pt] Your friend Bubu is amazed that there is now a completely new technology called “chat agent”. They are convinced that Lama3-Chat is not a language model and conceptually distinct from the regular Lama3 foundation model. You sigh and explain to them that is not so:

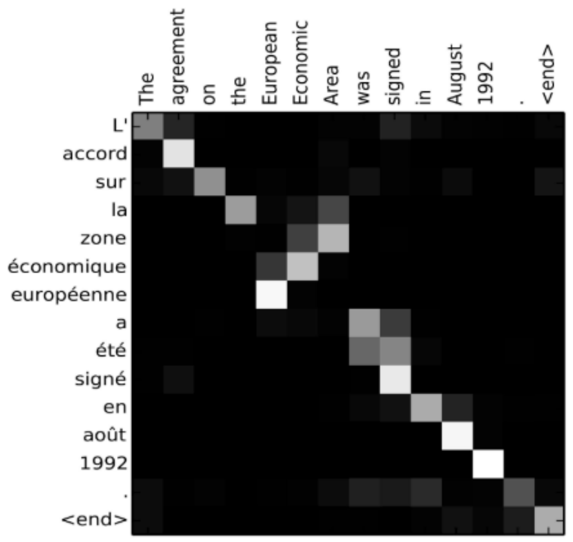
b. [4 pt] Your friend Kiki is amazed that there is now a completely new technology called “tool user”, which is conceptually different from a normal (causal) language model, because it can do stuff like write and execute code, or search the internet and inspect the results. You sigh and explain to them how it actually works:

c. [6 pt] Consider the “generative agents” architecture shown below, which we discussed in class (in the context of simulacra, acting as agents in a Smallville-like game environment). A critical component in this architecture is an elaborate model of memory, inspired by human memory. Name three uses in the memory architecture of LLMs, each supplying information that the agent model uses to decide what the agent remembers.



5. Probing & attribution [16 pt]

a. [2 pt] Which attention type (self-attention, cross attention, encoder or decoder) is shown on the left? Briefly justify your answer.



b. [4 pt] Your friend Bo has heard of the transformer architecture and is vaguely familiar with it, but has never seen such a figure before. Explain to Bo in brief and intuitive terms what the figure shows.

c. [8 pt] Your are running a probing experiment in order to investigate which linguistic information is learned by which layers of a transformer model. Your colleagues have already constructed three datasets (part-of-speech tagging, natural language inference, dataset of minimal pairs of (un)grammatical sentences). They have trained classifiers for all these datasets for each layer, presented below.

Layer	POS tagging	NLI	Minimal pairs
1	0.24	0.19	0.22
2	0.56	0.20	
3	0.87	0.19	0.57

(i) Your colleague forgot to calculate the accuracy for the classifier for the second layer on the minimal pairs dataset. The dataset has 10 samples and your colleague has the following list of results: [1, 0, 0, 0, 1, 1, 0, 1, 0, 0]. Fill in the missing accuracy value.

(ii) Now your job is now to summarize the experiment and the results in a short abstract, answering the following questions:
- In high-level intuitive terms, how does probing work?
- Which linguistic capabilities (phonology, syntax, semantics?) are targeted by each data set?
- What do you conclude about whether the model's linguistic abilities are good?

6. Calculating parts of the forward pass [XX pt]

We do not give a concrete example here, but you should also be prepared to calculate small-ish chunks of the forward pass for an MLP, RNN, LSTM, or Transformer. We might give you some vectors and matrices (very small scale) and some operations (e.g., how to calculate attention scores, or embeddings) and ask you to compute actual numbers. We will NOT require that you know all operations / functions / formulas. But we do require you to be able to apply a given operation / function / formula to concrete input.