Problem 1

The function "*segment_and_tokenize"* receives a piece of writing stored in a string and splits the whole article into sentences as a list of list. Each list is a sentence composing the whole article. All the sentences are tokenized in a word-based way. Moreover, it turns the words which appear only once into <UNKOWN>.

Problem 2

The "*make_word_to_ix*" takes tokenized articles as input and returns a dictionary. This dictionary has the words in corpus as keys and their corresponding values starting from 0 to the end of the article(eg. the total number of words-1).

Problem 3

The function first creates an empty list called "*one_hot_vecs*", then it loops over as "s" the input "*sents*", which is the tokenized corpus done by "*segment_and_tokenize".* Each element(eg. the "s") of "sents" is a sentence consisting of words in a tokenized way. Each "s" is sent into the "*sent_to_onehot_vecs*" function. This function sets the index of the vector as 1 if the word is in that index and other indexes of the vector remain as 0.

Problem 4

The "*embed_word*" takes the weight matrix W_e from param_dict and the one-hot vector "word" as input, and then compute the multiplication of the matrix W_e and the one-hot vector "word".

Problem 5

The "*elman_unit*" takes the weight matrix W_e, W_h from param_dict, the one-hot vector "current_word_embedding,h_previous" and the constant b as inputs. It first computes the multiplication between W_e and current_word_embedding and between W_h and h_previous. Then it sums up the two results of multiplication and the constant b. Finally, the summation result is put into a sigmoid function. The function returns the result coming from the sigmoid function.

$$sigmoid(x) = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-}(Wx * word\ embedding + Wh * h\ previous + b)}$$

Problem 6

The function "*single_layer_perceptron*" takes h and W_h as inputs. It first computes the multiplication of vector h and matrix W_h, then the multiplication result will be sent to a softmax function. The function will output the result coming out from the softmax function.

$$softmax(x_j) = \frac{e^{x_j}}{\Sigma i\ e^{x_i}}$$

Problem 7

The loss is decreasing throughout the training process. Back propagation is used in this network. Therefore, the problem is that it might eventually reach the local minima rather than the global minima.