NOTE: You may solve each problem using the language within the bracket. For example, **[Python]** means the solution should be in Python.

Q1 [Python] The probability of rain on a given calendar day in Vancouver is p[i], where i is the day's index. For example, p[0] is the probability of rain on January 1st, and p[10] is the probability of precipitation on January 11th. Assume the year has 365 days (i.e. p has 365 elements). What is the chance it rains more than n (e.g. 100) days in Vancouver? Write a function that accepts p (probabilities of rain on a given calendar day) and n as input arguments and returns the possibility of raining at least n days.

```
def prob_rain_more_than_n(p: Sequence[float], n: int) -> float:
    pass
```

Q2 [Python] A phoneme is a sound unit (similar to a character for text). We have an extensive pronunciation dictionary (think millions of words). Below is a snippet:

ABACUS	AE B AH K AH S
BOOK	B UH K
THEIR	DH EH R
THERE	DH EH R
TOMATO	T AH M AA T OW
TOMATO	T AH M EY T OW

Given a sequence of phonemes as input (e.g. ["DH", "EH", "R", "DH", "EH", "R"]), find all the combinations of the words that can produce this sequence (e.g. [["THEIR", "THEIR"], ["THEIR", "THERE"], ["THERE", "THEIR"], ["THERE", "THERE"]]). You can preprocess the dictionary into a different data structure if needed.

```
def find_word_combos_with_pronunciation(phonemes: Sequence[str]) -> Sequence[Sequence[str]]:
    pass
```

Q3 [C] Find the n most frequent words in the TensorFlow Shakespeare dataset.

```
char **find_frequent_words(const char *path, int32_t n) {
    // implementation
}
```

Q4 [Python] Implement CTC as described in this paper. Your implementation should support both forward and backward propagation operations.