NOTE: You need to solve ONLY 2 questions. You can choose any of the following.
NOTE: You may solve each problem using any of the languages within the bracket. For example **[C, Python]** means that the solution should be in C or Python.

Q1 [C, Python] Assume we have a function *get_book_info(isbn)* that takes a string ISBN argument and retrieves an object containing the Title, Author, and Language of a book (each represented as a string) that takes a nontrivial amount of time to run (perhaps because it's making a call to a database). Write a wrapper function that increases performance by keeping results in memory for the quick lookup. To prevent memory from growing unbounded, we only want to store a maximum of N book records. At any given time, we should be storing the N books that we accessed most recently. Assume that N can be a large number when choosing data structure(s) and algorithm(s).

Q2 [JavaScript, TypeScript] Implement a 5-star widget for an eCommerce site for users to record a product rating. The widget displays a horizontal row of stars that are either outlined or black according to the product rating, from left to right. E.g. ★★★☆☆ = rating of 3. Multiple 5-star widgets can be present on a single page. If a user has not rated a product, the widget will have 5 outlined stars (☆☆☆☆☆). Each product on the page has a UUID. Hovering over the Nth star will light up stars 1 to N with a grey colour (e.g. ★★★★☆). Clicking a star will record the rating by sending a request to a web server with enough information to record the product and the rating. After clicking, the widget will then display the rating the user submitted with black stars (e.g. ★★★★☆). Submitting the rating is handled without refreshing the page.
NOTE: You do not need to implement the backend.

Q3 [C] Design and implement a stack class (i.e. struct with accompanying functions). The interface should allow storing any data type including complex structures. Describe different implementation strategies and compare their pros and cons. What is the best approach in an embedded real-time system? What is the best approach when memory resources are very limited? You can use "malloc" and "free" functions.

Q4 [Python (NumPy)] Implement CTC as described in this paper. Your implementation should support both forward and backward propagation operations.
NOTE: The implementation should be in pure NumPy and you are not allowed to use TF, PyTorch, Keras, etc.

Q5 [Python (NumPy)] Implement a unidirectional multi-layer LSTM classifier with input and forget gates coupled. You can find information about this variant of LSTM here (look for CIFG). The model should accept a feature vector as input and emit the corresponding posterior. Then train a character-based language model to generate text resembling Shakespeare (use any online dataset you see fit). How do you measure the quality of the generated text? Justify all the design decisions you've made in your training and inference pipelines.
NOTE: The implementation should be in pure NumPy and you are not allowed to use TF, PyTorch, Keras, etc.

Q6 [C] Implement a matrix-vector multiplication function in C. All elements are stored as fixed-point numbers. The elements of the matrix are stored as Q7. Elements of the vector are stored as Q5.10. The result should be stored as Q5.10. Implement the function in pure C. Now assume you are running on an Arm Cortex-A with NEON SIMD (e.g. Raspberry Pi 3 or NVIDIA Jetson Nano). Re-write the C implementation using intrinsics to optimize for speed. Measure the speedup after re-writing using intrinsics.

```
void matrix_vector_multiplication(
    const int8_t *matrix,
    int32_t num_rows,
    Int32_t num_columns,
    const int16_t *input,
    int16_t *output);
```