Layup Parts Software Engineer Take Home Test

Task 1 - Airplane Control

Develop a basic application that simulates the control of an airplane on a 2D plane viewed from above. The airplane is assumed to be at a constant cruising altitude. The application should allow users to control the airplane's yaw angle (direction) and airspeed, and it should dynamically update and display the airplane's trajectory on a canvas as it moves.

We recommend spending at most 90 min on this task.

Requirements:

1. **Airplane Controls**:
   a. Implement controls for adjusting the airplane's yaw angle (in degrees) and airspeed (in knots)
   b. The user should be able to input these controls via a simple user interface
2. **Trajectory Visualization**:
   a. Draw the airplane's trajectory on a canvas element (or equivalent) as a continuous line that updates in real-time as the airplane moves
   b. The trajectory should start from an initial position, and each change in direction or speed should be reflected in the path
   c. Choose your own solution to the edges of the canvas

Deliverables:

- A working application with a canvas or similar element that visualizes the airplane's trajectory
- A simple UI for adjusting the yaw angle and airspeed
- Basic documentation that explains how to run the application and describes the logic behind the trajectory calculations

Task 2 - Layup Sequence

Implement an algorithm to compute the value of the *Layup Sequence* at $n = 10,000.$

The sequence is defined as follows:

$$S(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ S(n-1) + S(n-2) & \text{if } n \text{ is even} \\ 2S(n-1) - S(n-2) & \text{if } n \text{ is odd} \end{cases}$$

We recommend spending at most 30 min on this task.

Requirements:

1. **Algorithm Implementation:**
   a. Implement the sequence computation function based on the given recurrence relation
   b. The function should efficiently compute the value of $S(n = 10,000)$
2. **Performance Evaluation:**
   a. Measure the runtime of your implementation.
   b. Determine the time complexity of your solution.
   c. If possible, optimize the solution to reduce the computational overhead.

3. **Explanation:**
   a. Provide a short explanation of the time complexity of your algorithm. You should provide a plot of N vs Runtime to backup your reasoning.
   b. Discuss any optimizations applied and how they impact the runtime.

Deliverables:

- A program or script that computes $S(n = 10,000)$
- A report (could be a comment in the code) discussing the runtime, time complexity, and any optimizations made.