**Sample Questions to Prepare for Lab Eval-1**

Required Skills:

1. Fundamental concepts of Linear Data Structures operations, sorting and searching.
2. Implementation of ADTs of linear data structures.
3. Implementation of Data structure libraries.
4. Choice of appropriate data structures while designing an algorithm to solve a given problem.
5. Time and space complexity analysis of your designed algorithm.

**Questions (Hybrid Data Structure Design)**

1. **Task Priority Management System:** Design a task management system where tasks are categorized into different priority levels (ex: 1, 2, 3, 4). Each priority level is represented as a queue, allowing tasks to be added to the appropriate queue based on priority. Utilize a combination of queues for each priority level within a larger data structure for managing tasks efficiently.
2. **Multi-level Undo/Redo Functionality:** Design a text editing application that incorporates multi-level undo/redo functionality. Each change made to the document should be stored in a linked list, with each node containing a stack of changes. Use a doubly linked list for efficient traversal and stack for managing changes in each node.
3. **Event Registration System for a Conference:** Design an event registration system for a conference where attendees register for different sessions. Each session's attendee list is represented as a stack, allowing attendees to register and deregister for sessions. Utilize a combination of stacks within a larger data structure to manage session registrations efficiently.
4. **Restaurant Reservation System**: Design a reservation system for a restaurant where each table's reservation list is represented as a queue of reservations. New reservations are added to the appropriate table's queue based on factors like reservation time and table availability. Utilize a combination of queues within a larger data structure for efficient reservation management.

1. **Bug Tracking System for Software Development**: Design a bug tracking system for software development where each software component's bug list is represented as a stack of bugs. New bugs are added to the appropriate component's stack, and bugs are resolved by popping them from the stack. Utilize a combination of stacks within a larger data structure for efficient bug tracking across software components.

**Scenario-based Question**

You are a software engineer working on a music player application for a popular streaming service. The application allows users to create playlists containing their favorite songs and provides features for seamless navigation through the playlist. Your client has come to you with the following requirements:

1. Users should be able to create playlists, add songs to them, and navigate through the playlist easily.
2. The application should support features like playing, skipping to the next song, and going back to the previous song.
3. Users should be able to undo or redo their actions while navigating the playlist.
4. Users should be able to find their favorite song. This is the song most frequently played by the user.

Based on the above scenario, answer the following questions:

1. Write the algorithm and explain your logic towards the design of your solution for the music player application to fulfill the client's requirements by providing seamless playlist management, navigation features, undo/redo functionalities, and the ability to find the user's favorite song.
2. Analyse your proposed solution and mention the asymptotic time complexity for the same. Justify your answer.
3. Convert the above scenario into viable code abstractions and explain the same as a preamble to the code using appropriate code comments and explanations.
4. Implement the logic explained in the preamble in C++, or, Java or Python.
5. Generate appropriate test cases for your proposed solution to showcase the impact of various implementation and design choices on the data structure's performance.