HW2

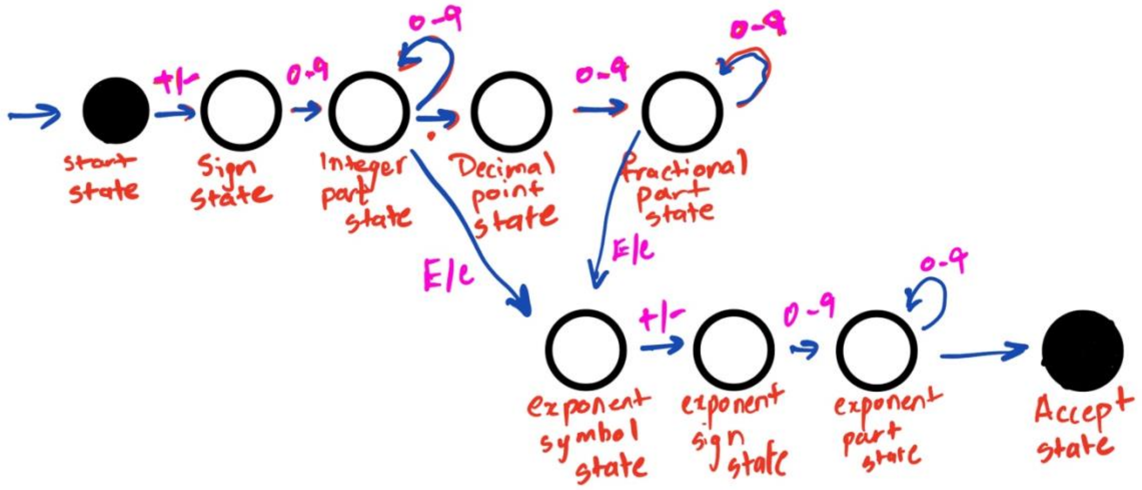
1. Component-Based Architecture (CBA) and Service-Oriented Architecture (SOA) are distinct approaches to software development, each with its unique focus and application. CBA emphasizes the modularization and reuse of software through self-contained, interchangeable components that encapsulate specific functionalities within an application. These components, often fine-grained, are integrated at development time, enhancing modularity and reuse within software applications. In contrast, SOA is centered around the concept of loosely coupled services, which are coarse-grained, communicate over a network, and represent larger business processes. SOA aims to improve agility and flexibility by enabling services to be combined and reused across different systems and applications, promoting interoperability through standardized communication protocols. While CBA focuses on the internal structure of an application by breaking it down into modular components, SOA addresses the integration and orchestration of distributed services across various platforms, emphasizing external service composition and dynamic interaction.

2. For a tic-tac-toe phone application featuring gameplay against a computer and local high score storage, a Component-Based Architecture (CBA) is most suitable. CBA's modular approach aligns well with the game's simplicity, allowing for clear separation between game logic, user interface, and score management components. This architecture facilitates easier development and maintenance, particularly for applications with a significant focus on user interface and local data handling, without the need for external data exchanges or service integrations. Given the game's requirements for responsiveness and local storage, without the complexities of network communications or scalability across systems, CBA offers an efficient, streamlined framework for building and managing the game's functionalities within the mobile environment.

3. For a chess application that enables two users to play against each other over the Internet, incorporating elements of both Component-Based Architecture (CBA) and Service-Oriented Architecture (SOA) would be most effective. The modular nature of CBA is still beneficial for structuring the game's core components, such as the chessboard interface, game logic, and user management. However, the requirement for Internet-based multiplayer functionality introduces a need for SOA elements. SOA would facilitate the communication between client applications and a backend server, managing game sessions, player moves, and state synchronization over the network. This hybrid approach leverages CBA for the application's internal structure and user interface, while employing SOA principles for network communication and service integration, ensuring a seamless and responsive online multiplayer experience.

4. For the ClassyDraw application, utilizing a NoSQL database, such as MongoDB, would provide the necessary flexibility and scalability. This database type is well-suited for storing complex data structures, which is essential for representing the various graphical elements and attributes within each drawing. Each project or drawing could be stored as an individual document, encompassing all related data including layers, colors, and tool settings, facilitating straightforward retrieval and updates.To ensure the application remains efficient and responsive, particularly as the user base and volume of stored drawings increase, regular database maintenance practices should be implemented. This includes conducting routine backups to safeguard against data loss and creating indexes on frequently queried fields to speed up search operations.Considering the potential for the application to scale, both in terms of data volume and user count, the database architecture should be designed with scalability in mind from the outset. Techniques such as shading, which distributes data across multiple servers, and replication, ensuring data is copied to multiple locations for redundancy and availability, can be employed to maintain performance levels and ensure a seamless user experience as the application grows.

5.



6. In a drawing application like ClassyDraw, classes like Line, Rectangle, Ellipse, Star, and Text share common properties such as position, color, stroke, visibility, and layer. These should be implemented in a base class to promote code reuse. Unique properties, like the end points for a Line or the text content for Text, are specific to each class and should be implemented individually. Properties like corner radius or line dash pattern, shared by some but not all shapes, could be managed through more specialized inheritance or interfaces. This approach ensures a clean and maintainable code structure, separating common functionalities from unique attributes.

7.