PROJECT REPORT
DATA STRUCTURE AND ALGORITHMS

- **PROJECT TITLE:**
  - CRICKET LEAGUE MANAGEMENT SYSTEM

- **PROJECT GROUP MEMBERS:**
  - MUHAMMAD MOOSA KHALIL **(01-131222-035)**
  - TAYYAB AAMIR ALI **(01-131222-048)**

- **PROJECT SUBMITTED TO:**
  - ENGINEER AAMIR SOHAIL

TABLE OF CONTENTS

1. HEADER FILES

1) `#include <algorithm>`:
Header file providing various algorithms like sorting and searching, aiding in efficient data manipulation.

2) `#include <conio.h>`:
Legacy header for console input and output functions, commonly used for capturing keystrokes without echoing them.

3) `#include <cstdlib>`:
Header file for the C Standard Library, offering general-purpose functions like memory allocation and random number generation.

4) `#include <ctime>`:
Header file for date and time functions, facilitating operations related to time measurement and manipulation.

5) `#include <fstream>`:
Header file for file input and output operations, enabling reading from and writing to files.

6) `#include <iostream>`:
Standard input-output stream objects like `cin` and `cout`, essential for console-based input and output.

7) `#include <limits>`:
Provides information about the numeric limits of the system, assisting in handling extreme values.

8) `#include <string>`:
Header file for string manipulation functions and classes, enabling efficient handling of textual data.

9) **`#include <vector>`:**
Standard template library container for dynamic arrays, allowing flexible storage and manipulation of elements.

10) **`#include <iomanip>`:**
Header file for input and output manipulators, facilitating custom formatting of console output.

11) **`#include <sstream>`:**
Provides string stream classes for string manipulation, allowing conversion between strings and other data types.

12) **`#include <set>`:**
Implements a set container that contains unique elements, useful for maintaining unique values in a collection.

13) **`#include <stack>`:**

Implements a stack data structure, allowing operations like push and pop on a Last-In-First-Out (LIFO) basis.

2. FUNCTIONS USED

a) Authentication-related Functions

- **`displayUserAuthenticationMenu ():`** Displays the user authentication menu, allowing users to log in or sign up.
- **`signup ():`** Enables users to create a new account by providing necessary details.
- **`userAuthentication():`** Manages the authentication process by verifying user credentials.

b) Menu-related Functions

- **`displayMainMenu ():`** Presents the main menu options for accessing different functionalities.
- **`displayTeamManagementMenu (), displayPlayerManagementMenu(), displaySponsorManagementMenu(), displayMatchManagementMenu():`** Displays submenus for team, player, sponsor, and match management.

c) Team Management Functions

- **`deleteTeam(), displayRegisteredTeams(), registerNewTeam(), renameTeam(), teamManagement(), viewPlayers():`** Provides functionalities for managing teams, including creation, deletion, renaming, and viewing.

d) **Player Management Functions**

- **`releasePlayer()`, `purchasePlayer()`, `displayPlayerStatistics()`,`displayPlayerCategories()`, `performPlayerExchange()`,`playerManagement()`:** Handles player-related operations such as adding, releasing, displaying statistics, and managing exchanges.

e) **Sponsor Management Functions**

- **`sponsorManagement()`,`displaySponsorCategories()`, `selectSponsor()`,`isTeamRegistered()`:** Facilitates sponsor management, including category display, selection, and registration.

f) **Match Management Functions**

- **`matchManagement()`,`generateRandomDate()`, `generateRandomVenue()`,`viewMatchSchedule()`, `viewPointsTable()`,`clearInputBuffer()`, `viewPointTable()`:** Manages match-related tasks such as scheduling, venue generation, and displaying match schedules and points tables.

3. KEY CONCEPTS & FUNCTIONALITIES

1) **File Handling:**
   Utilized for storing and retrieving data, ensuring persistence across sessions.

2) **User Authentication:**
   Provides secure access control through login and signup mechanisms.

3) **Menu-driven Interface:**
   Offers intuitive navigation and accessibility to various functionalities.

4) **Data Management:**
   Efficient organization and manipulation of team, player, and sponsor data for streamlined operations.

5) **Randomization:**
   Realistic generation of match dates and venues for enhanced realism in match management.

4. DSA CONCEPTS USED

a) Linked Lists:

Utilized in managing player data, where a linked list structure facilitates dynamic memory allocation and efficient traversal, insertion, and deletion of player nodes.

b) Trees:

Not explicitly mentioned in the code snippet provided, but could potentially be employed in other parts of the project, such as representing hierarchical data structures like tournament brackets or organizational charts.

c) Vectors:

Used for storing dynamic arrays of player and sponsor data, offering flexibility in size and efficient random access to elements.

d) Sorting Algorithms:

Likely implemented in functionalities such as sorting players based on statistics or arranging sponsors alphabetically, ensuring data is presented in a meaningful and organized manner.

5. CODE STRUCTURE

➢ The code follows a modular structure with separate sections for function declarations, including authentication-related functions, menu-related functions, team management functions, player management functions, sponsor management functions, and match management functions. It starts with necessary header file inclusions, followed by function declarations and implementations, organized logically to handle different aspects of the cricket management system.

6. CONCLUSION

- The Cricket Management System is a well-structured console-based application designed to efficiently manage various aspects of cricket team operations. Through a combination of header files, global variables, and modular functions, the project achieves a high level of organization and readability.
- Overall, the Cricket Management System demonstrates effective implementation of concepts such as modular programming, file handling, and user interaction, making it a valuable tool for managing cricket teams efficiently. Further enhancements could include refining the user interface, implementing advanced authentication mechanisms, and integrating additional features to cater to diverse user requirements.