**Project Report**

**Syed Ashhal Hasan 22k-4306**

**Shuja Ur Rahman 22k-4456**

**Omer Altaf Siddiqui 22k-4266**

**\*\*Project Report: Movie Recommendation System\*\***

\*\*1. Introduction:\*\*

- The project aims to develop a movie recommendation system using C++. The system utilizes data structures such as Binary Search Trees (BST), linked lists, and a hash table to efficiently store and retrieve movie information.

1. **Node Class:**
   * Represents a node in the Binary Search Tree (BST).
   * Holds information about a movie, including user ID, movie details (name, ID, rating, genre, etc.), and pointers to left and right child nodes.
   * Used for constructing the Binary Search Tree of movies.
2. **BSTNode Class:**
   * Generic class for a node in a Binary Search Tree.
   * Used in the implementation of a linked list for storing movies in the hash table.
3. **List Class:**
   * Linked list implementation for storing movies.
   * Used for maintaining lists such as watchlist, liked movies, and movies watched by a user.
4. **HashTable Class:**
   * Implements a hash table for efficient storage and retrieval of movies.
   * Each bucket in the hash table is a linked list of BSTNodes, allowing for collision resolution.
   * Uses separate chaining to handle collisions.
5. **CompareMovies Structure:**
   * Function for comparing movies based on their ratings.
   * Used in the priority queue to prioritize movies with higher ratings.
6. **Movies Class:**
   * Represents the movie database and includes methods for inserting, searching, and recommending movies.
   * Utilizes AVL tree-based BST for efficient storage and retrieval of movies.
   * Implements various search criteria such as name, genre, release date, and rating.
7. **User Class:**
   * Represents a user in the movie recommendation system.
   * Maintains information about the user, including user ID, name, liked movies, movies watched, watchlist, liked genres, and user responses (ratings and reviews).
   * Includes methods for signing in, adding liked genres, recommending movies, displaying watchlists and favorites, and searching for movies based on different criteria.

These data structures work together to create a comprehensive movie recommendation system that supports user interactions, movie searches, and personalized recommendations based on user preferences and behavior. The use of AVL trees ensures efficient storage and retrieval of movies in a balanced manner. The hash table helps optimize the search process for specific movies, and linked lists facilitate the management of user-specific lists like watchlist and liked movies.

\*\*2.2 Functions:\*\*

\*\*Utility Functions:\*\*

- `stringtoint`: Converts a string to an integer.

- `hashfunc`: Computes a hash value for a given string.

\*\*BST Operations:\*\*

- `insert`: Inserts a new movie into the Binary Search Tree (BST).

- `heightFinder`: Computes the height of a tree.

- `get\_max`: Returns the maximum of two values.

- `checkBalance`: Checks the balance factor of a node.

- `RightRotation`: Performs a right rotation in AVL tree.

- `LeftRotation`: Performs a left rotation in AVL tree.

- `get\_min`: Finds the minimum node in the tree.

- `searchName`: Searches for a movie by name.

- `ratingInorder`: Traverses the tree to find top-rated movies.

- `preferedInorder`: Traverses the tree to find movies with a preferred rating.

- `ByGenre`: Traverses the tree to find movies based on genre.

- `inorderByDate`: Traverses the tree to find movies based on release date.

\*\*List Operations:\*\*

- `pushBack`: Adds a node to the end of the linked list.

- `printList`: Prints the elements of the linked list.

- `deleteNode`: Deletes a node from the linked list.

\*\*Hash Table Operations:\*\*

- `insert`: Inserts a movie into the hash table.

- `searchTable`: Searches for a movie in the hash table.

- `print`: Prints recommended movies from the hash table.

\*\*User Class Operations:\*\*

- `addLikedGenre`: Adds a genre to the user's liked genres.

- `signin`: Signs in the user.

- `recommendMovies`: Recommends movies to the user based on preferences.

- `addMoviesWatched`: Adds a movie to the user's watched list.

- `displayWatchlist`: Displays the user's watchlist.

- `displayFavourites`: Displays the user's liked movies.

- `displayMoviesWatched`: Displays the user's watched movies.

- `addRating`: Adds a rating and review for a movie.

- `searchMovies`: Allows the user to search for movies based on various criteria.

\*\*3. Conclusion:\*\*

- The movie recommendation system provides users with a personalized movie-watching experience. The use of data structures such as BST and hash tables ensures efficient storage and retrieval of movie information. Users can explore, rate, and review movies based on their preferences, creating a dynamic and interactive system.

\*\*4. Future Enhancements:\*\*

- Implementation of a graphical user interface (GUI).

- Integration with external databases for an extensive movie collection.

- Collaborative filtering for more accurate recommendations.

\*\*5. Acknowledgments:\*\*

- The project acknowledges the use of data structures and algorithms for creating an efficient and user-friendly movie recommendation system. Special thanks to the developers and contributors involved in building and maintaining the C++ programming language.

\*\*6. Appendix:\*\*

- Code snippets and detailed function documentation can be found in the attached source code files.

\*\*List of Key Functions:\*\*

1. `stringtoint`

2. `hashfunc`

3. `display`

4. `print`

5. `insert` (BST)

6. `heightFinder`

7. `get\_max`

8. `checkBalance`

9. `RightRotation`

10. `LeftRotation`

11. `get\_min`

12. `containsSubstring`

13. `searchName`

14. `ratingInorder`

15. `searchTopRatedMovies`

16. `preferedInorder`

17. `searchByRating`

18. `searchByGenre`

19. `ByGenre`

20. `searchByDate`

21. `inorderByDate`

22. `traverse` (User class)

23. `addLikedGenre`

24. `signin`

25. `recommendMovies`

26. `addMoviesWatched`

27. `displayWatchlist`

28. `displayFavourites`

29. `displayMoviesWatched`

30. `addRating`

31. `searchMovies`