# EEN 218
# Lab #6
# A Movie Repository II: Movies, Playlists, and Search Trees

In this assignment, you are going to complete your movie library, using Lab 4 as a starting point.

**You need to provide a report that shows (20pts):**

1. Your class designs
2. The algorithms used
    a. This should be in pseudo-code or flowcharts, not just the code you ended up writing
3. Compiling instructions
4. Sample runs (screenshots are fine)

**Things to have (80pts):**

**Part 1: Binary Search Tree for the Movies (25pts)**

1. A Binary Search Tree that stores all your Movies for the entire library, this is the **MovieTree** and is the main repository for your movie library
    a. Basically you are replacing the order linked list from Lab 4 with a Binary Search Tree
        i. The Movies remain the same with their internal Review Lists from Lab 4.
        ii. Only the MovieList is scrapped in favor of the BST
    b. Movies are sorted by movie title.
    c. No duplicates are allowed.
    d. Remember that the Movies are referenced through pointers in the tree nodes.
    e. Private pointer to the root of the tree and a count of the number of movies in the tree
    f. A constructor and destructor for the BST
    g. The following public methods:
        i. void addMovie(Movie *s);
        ii. Movie *findMovie(string title);
        iii. int getDepth();
        iv. int getNumMovies();
    h. void printTree(bool ascending=true);
2. A function to read the movies from a file and load them into the **MovieTree**

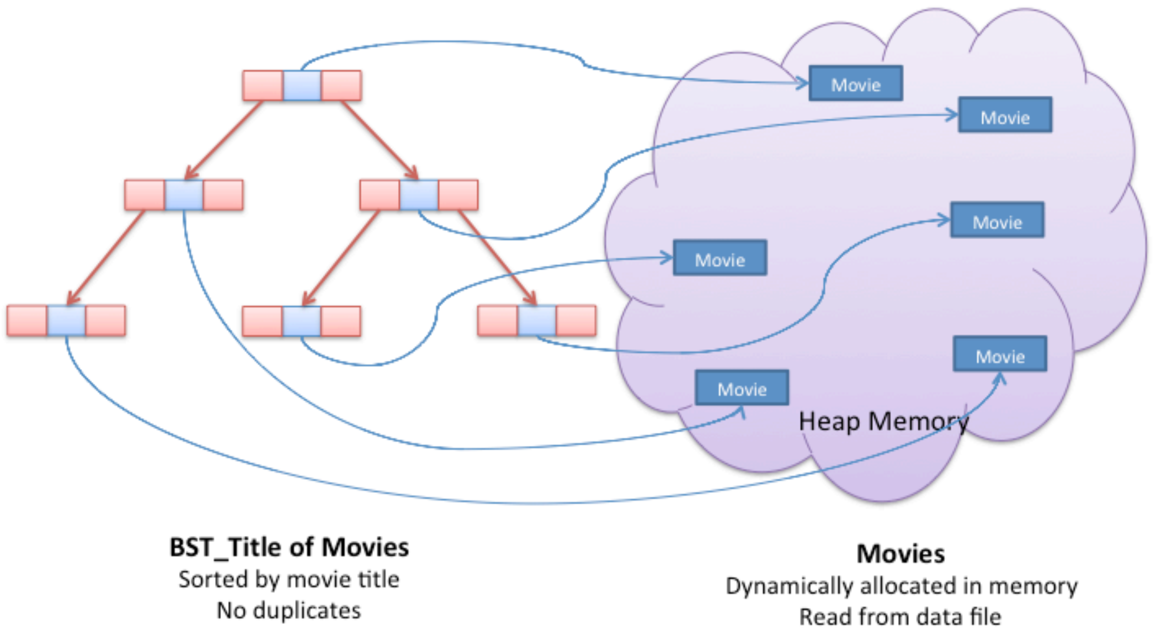

**Figure 1 BST of Movies**

**Part 2: RunTime PlayLists of Movies (25pts)**

3. A PlayList for Movies, **MoviePlayList**
    a. This is a simple linked-list similar to the one you used for the Reviews in Lab 4 except that the Movies in the playlist nodes are pointers to the Movies
        i. You should make the Review and MoviePlay Lists now as a Templates.
    b. Note that the same Movie can exist in different Playlists
    c. Enforce that the same Movie cannot exist in the same Playlist
    d. Not all Movies have to be in a Playlist
    e. Each MoviePlayList also has a title
4. An algorithm that automatically populates a MoviePlayList with movies based on the Maximum Runtime of the Playlist
    a. The user specifies the total runtime of the PlayList
    b. The system determines the most Movies that can fit into the PlayList from all the available movies.

**Part 3: Binary Search Tree for the MoviePlaylists (20pts)**

5. A Binary Search Tree that stores all the Playlists in your repository, this is the **PlaylistTree**
    a. The nodes of this tree use pointers to the Playlists to reference them
    b. The tree is sorted by Playlist title
    c. No duplicates are allowed
6. A function to read the playlists from a file, playlists.txt, create the Playlists and create the **PlaylistTree (10pts)**

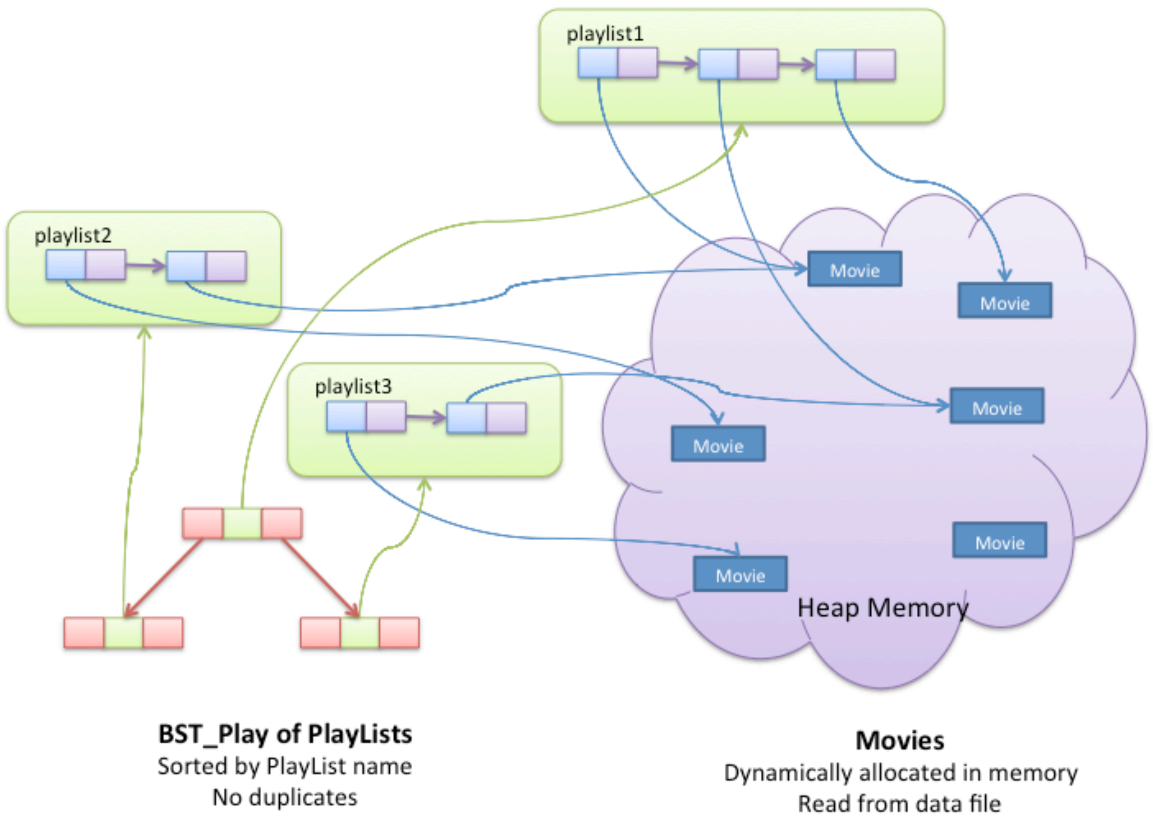

**Figure 2 Binary Search Tree of Playlists**

**Part 4: Menu (10pts)**

7. A menu to allow the user to interact with the system. It should allow **(30pts):**
    a. Main
        i. Load Movies from file
        ii. Load Playlists from file
    b. Songs
        i. List all Movies
        ii. Search for Movie by title
    c. Playlists
        i. List all Playlists
        ii. Search for Playlist by title
        iii. List Movies for Playlist
        iv. Create Playlist
        v. Save Playlists
    d. Exit

**Extra Credit: (do option 1 OR option 2, no extra will be given for both)**

***Option1: Implement Polymorphic Trees (10pts)***

Instead of creating two different tree class (one with Movie pointers and one with Playlist pointers), create an abstract superclass with Movie and Playlist as subclasses. Create one tree class that uses pointers to this superclass. Implement a polymorphic compare method for the use by the Tree.

***Option2: Implement Genres (10pts)***

These are similar to Playlists, but hold related Movies based on Genre. You also need to add another Binary Search Tree for Genre, **GenreTree**.
Update the menu to allow for Listing/Searching Genre, and the RunTime Playlists to allow specification of Genre in addition to total runtime.