# CSCI 2270 – Data Structures Fall 2022

# Assignment 3 - Linked Lists

## Show time scheduler

### OBJECTIVES

1. **Create, traverse and add nodes to a linked list**
2. **Get practice implementing classes**

**Corrections:**

**1. Removed 'Vikings' from buildShowsList()  2. Outputs have been rectified.**

## Building your own show schedule

You will be implementing a class to simulate a linked list of shows. There are three files in your repository containing a code skeleton to get you started. *Do not modify the header file or your code won't work!* You will have to complete both the class implementation in ShowsList.cpp and the driver file main_1.cpp.

The linked-list itself will be implemented using the following struct (already included in the header file):

```
//Show: node struct that will be stored in the ShowsList linked-list
   struct Show
   {
      std::string name;          // name of the show
      double rating;         // average rating this show has received
      int numberRatings;    // total count of ratings received by this
show
      Show *next;            // pointer to the next show
   };
```

# CSCI 2270 – Data Structures Fall 2022

## Class Specifications

The **ShowsList** class definition is provided in the file *ShowsList.hpp* in Canvas. *Do not modify this file or your code won't work!* Fill in the file *ShowsList.cpp* according to the following specifications. **Make sure you complete the functions in the order they are given here**

**Show* head;**

➔ Points to the first node in the linked list

**ShowsList();**

➔ Class constructor; set the head pointer to NULL

**bool isEmpty();**

➔ Return true if the head is NULL, false otherwise

**void displayShows();**

➔ Print the names of each node in the linked list. Below is an example of correct output using the default setup. (Note that you will **cout << "NULL"** at the end of the list)

```
== CURRENT SHOWS LIST ==
Friends-> Ozark-> Stranger Things-> The Boys-> Better Call Saul-> NULL
===
```

➔ If the show list is empty then print *"nothing in path"* with an endline.

**void addShow (Show* previousShow, string showName);** *// Beware of edge cases*

➔ Insert a new show with name **showName** in the linked list after the show pointed to by **previousShow**.

➔ If **previousShow** is NULL, then add the new show to the beginning of the list.

➔ Print the name of the show you added according to the following format:

```
// If you are adding at the beginning use this: cout
<< "adding: " << showName << " (HEAD)" << endl;

// Otherwise use this:
cout << "adding: " << showName << " (prev: " <<
previousShow->name << ")" << endl;
```

**void buildShowsList();**

➔ Add the following five shows, in order, to the list with **addShow**: "Friends", "Ozark", "Stranger Things", "The Boys", "Better Call Saul". This is the default setup of shows in the linked list, therefore, you can keep 0 as the **`rating`** value and the **`numberRatings`** for each of them.

**Show* searchShow(string showName);**

➔ Return a pointer to the node with name **showName**. If **showName** cannot be found, return NULL

**void addRating(string receiver, double rating);**

➔ Find the `node` with the name **receiver**. For the receiver `node`, update the rating. You will update the rating following the running average rating formula of the previous assignment. You will also update the numberRatings.

- `node->numberRatings++;`
- `node->rating = (node->rating*(node->numberRatings-1) + rating)/node->numberRatings`

If the list is empty, print "Empty list" and exit the function. If the node is not present, print “Show not found”. For both cases, change the line after printing.

Otherwise print the updated rating using following cout assuming temp is pointing to the concerned node

```
cout << "The rating has been updated: " << node->rating << endl;
```

# Main driver file

Your program will start by displaying a menu by calling the **displayMenu** function included in main.cpp. The user will select an option from the menu to decide what the program will do, after which, the menu will be displayed again. The specifics of each menu option are described below.

### Option 1: Build schedule

This option calls the **buildShowsList()** function, then calls the **displayShows()** function. You should get the following output:

```
adding: Friends (HEAD)
adding: Ozark (prev: Friends)
adding: Stranger Things (prev: Ozark)
adding: The Boys (prev: Stranger Things)
adding: Better Call Saul (prev: The Boys)
== CURRENT SHOWS LIST ==
Friends-> Ozark-> Stranger Things-> The Boys-> Better Call Saul-> NULL
===
```

### Option 2: Display Shows

Call the **displayShows** function. Output should be in the format below:

```
// Output for the default setup
== CURRENT SHOWS LIST ==
Friends-> Ozark-> Stranger Things-> The Boys-> Better Call Saul-> NULL
===
// Output when the linked list is empty
== CURRENT SHOWS LIST ==
nothing in path
===
```

### Option 3: Add Show

Prompt the user for two inputs: the name of a new show to add to the list, **showName**, and the name of a show already in the list, **previousShow**, which will precede the new show. Use the member functions **searchShows** and **addShows** to insert **showName** into the linked-list right after the node with the show name **previousShow**.

- If the user wants to add the new show to the head of the list then they should enter “First” instead of a previous show name.
- If the user enters an invalid previous show name (not present in the linked list), then you need to prompt the user with the following error message and collect input again until they enter a valid previous show name or “First”:

```
cout << "INVALID(previous show name)... Please enter a VALID previous show name!" << endl;
```

- Once a valid previous show name is passed and the new show is added, call the function **displayShows** to demonstrate the new linked-list.

For example, the following should be the output if the linked-list contains the default setup from option (1) and the user wants to add Viking after Ozark:

```
Enter a new show name:
Vikings
Enter the previous show name (or First):
Ozark

adding: Vikings (prev: Ozark)
== CURRENT SHOWS LIST ==
Friends-> Ozark-> Vikings-> Stranger Things-> The Boys-> Better Call Saul->
NULL
===
```

### Option 4: Add Rating

Prompt the user for two inputs: Show Name, and the Rating *(Hint: use **getline** in case there are spaces in the user input)*. Pass the Show name and rating to the **addRating**

For example, the following should be the output if the linked-list contains the default setup from option (1):

```
Example 1:
Enter name of the show to add the rating:
Ozark
Enter the rating:
4.5
The rating has been updated: 4.5
```

If the user then decides to add rating to show "Peaky Blinders" which does not exists in the show list

```
Example 2:
Enter name of the show to add the rating:
Peaky Blinders
Enter the rating:
4.5
Show not found
```

**Option 5: Quit**

Print the following message:

```
cout << "Quitting..." << endl;
```

In the end, print

```
cout << "Goodbye!" << endl;
```

## Important Note regarding Test Cases

Since there is interdependence between the following test cases, even though you will get some score for implementing them, they will only pass if all the functions have been implemented correctly:

- TestBuildShowsList
- TestIsEmpty
- TestDisplayShows
- TestSearchShow
- TestAddShow

## Appendix:

- **ShowsList::insertShow()**
    - `cout << "adding: " << showName << " (HEAD)" << endl;`
    - `cout << "adding: " << showName << " (prev: " << previousShow->name << ")" << endl;`
- **ShowsList::displayShow()**
    - `cout << "== CURRENT SHOWS LIST ==" << endl;`
    - `cout << "nothing in path" << endl;`
    - `cout << ptr->name << "-> ";`
    - `cout << "NULL" << endl;`
    - `cout << "===" << endl;`

# CSCI 2270 – Data Structures Fall 2022

- **main()**
  - cout << "Enter name of the show to add the rating: "<<endl;
  - cout << "Enter the rating: " << endl;
  - cout << endl;
  - cout << "Enter a new show name: " << endl;
    - cout << "Enter the previous show name (or First): "<<endl;
    - cout << "INVALID(previous show name)... Please enter a VALID previous show name!" << endl;
  - cout << "Quitting..." << endl;
  - cout << "Invalid Input" << endl;
  - cout << "Goodbye!" << endl;

- **displayMenu()**
  - cout << endl;
  - cout << "Select a numerical option:" << endl;
  - cout << "+=====Main Menu=========+" << endl;
  - cout << " 1. Build schedule " << endl;
  - cout << " 2. Display Shows " << endl;
  - cout << " 3. Add Show " << endl;
  - cout << " 4. Add rating" << endl;
  - cout << " 5. Quit " << endl;
  - cout << "+----------------------+" << endl;
  - cout << "#> ";