**Lab Assignment 7: Queue Data Structure**

**Objectives:**

* Using a queue to store the information
* Save the queue data into a file and read them back.
* Handle input errors and invalid values
* Design and create a well-structure program using C++ basic programming constructs

**Description:**

Write a menu-driven program to keep track of your favorite movies:

MENU  
A Add a movie  
U Update a movie  
E rEmove a movie  
L List all movies  
T arrange by Title  
V arrange by year Viewed  
R arrange by Rating  
Q Quit  
...your choice: **a[ENTER]**  
  
Enter a movie's name: **Star Wars[ENTER]**  
Enter the year you saw Star Wars [like 2016]: **1977[ENTER]**  
Enter your rating for Star Wars [1, 2, 3, 4, 5]: **1[ENTER]**  
  
MENU  
A Add a movie  
U Update a movie  
E rEmove a movie  
L List all movies  
T arrange by Title  
V arrange by year Viewed  
R arrange by Rating  
Q Quit  
...your choice: **a[ENTER]**  
  
Enter a movies' name: **Mission Impossible[ENTER]**  
Enter the year you saw Mission Impossible [like 2016]: **2015[ENTER]**  
Enter your rating for Mission Impossible [1, 2, 3, 4, 5]: **2[ENTER]**  
  
MENU  
A Add a movie  
U Update a movie  
E rEmove a movie  
L List all movies  
T arrange by Title  
V arrange by year Viewed  
R arrange by Rating  
Q Quit  
...your choice: **L[ENTER]**  
  
 # Title Viewed Rating  
-- --------------------------- ------ ------  
 1 Mission Impossible 2015 2  
 2 Star Wars 2016 1  
  
MENU  
A Add a movie  
U Update a movie  
E rEmove a movie  
L List all movies  
T arrange by Title  
V arrange by year Viewed  
R arrange by Rating  
Q Quit  
...your choice: **q[ENTER]**

Use **sequence numbers** for the "remove" and "update" options, like this:

# Title Viewed Rating  
-- --------------------------- ------ ------  
 1 Mission Impossible 2015 2  
 2 Star Wars 2016 1  
  
MENU  
A Add a movie  
U Update a movie  
E rEmove a movie  
L List all movies  
T arrange by Title  
V arrange by year Viewed  
R arrange by Rating  
Q Quit  
...your choice: E**[ENTER]**...which movie to remove (1-2)? **2[ENTER]**

**Requirements:**

1. You decide whether to use **C or C++ strings**, as long as movie titles can be up to 50 characters long.
2. Do **case-independent sorting**, but do *not* convert case of any input.
3. You decide upon your **rating system** -- can be any data type.
4. The "**remove**" option should actually *remove* the node. Use **sequence numbering** to identify the movie to remove, with the valid range of numbers in the prompt.
5. You decide what to say, if anything, if there's **no match** for *any* of the user's input.
6. Use a **queue** to store the movie information. Initially, the queue is loaded with data from a file named **“movies.txt”.** If the file doesn’t exist, start with an empty queue.
7. Write a function to add a new node at the **back** of the queue.
8. The output table should have nicely-spaced **column headings** and sequence numbering.
9. Use **blank lines** in the output to separate blocks of text as modeled in the sample output above.
10. Avoid memory leaks as nodes are removed and when the program ends -- use a **delete** loop.
11. At the end of your program, save all data into a file named **“movies.txt”**. You should write a string “EOF” at the end of your file so that you can tell there is no more data when you read it back.

### Hints:

### You may want to keep track of the last node in the queue by using a tail pointer so that you don’t need to traverse the queue from the beginning to add a new node.

### Keeping track of the number of nodes.

Did you notice the "Which movie to update [1-3]:" prompt? It's not as easy as it looks! What if a node has been removed? Then it's "[1-2]". You'll need a way to keep track of the number of nodes in your list, and a way to put that number in the update and remove prompts. It's not too hard if you think about it and plan it out.

Do **not** save the sequence number as an attribute of the movie objects!