

CS-240: Programming Assignment Phase 1

Winter Semester 2023-2024

John Malliotakis – jmal@csd.uoc.gr

27/10/2023



From the description:

In this assignment you are asked to implement a simplified movie streaming service. The service offers movies classified into different categories. Users register to the service, watch movies by adding them to their history, accept movie suggestions based on other users' watch history, and perform filtered searches across movie categories.

Structures & Organization

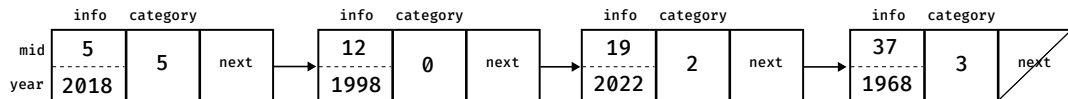
Belong to **one** of 6 categories:

1. Horror
2. Science-Fiction
3. Drama
4. Romance
5. Documentary
6. Comedy

Have the following **info** available:

mid *Unique identifier per movie, unsigned int*
year *Movie release year, unsigned int*

New Releases



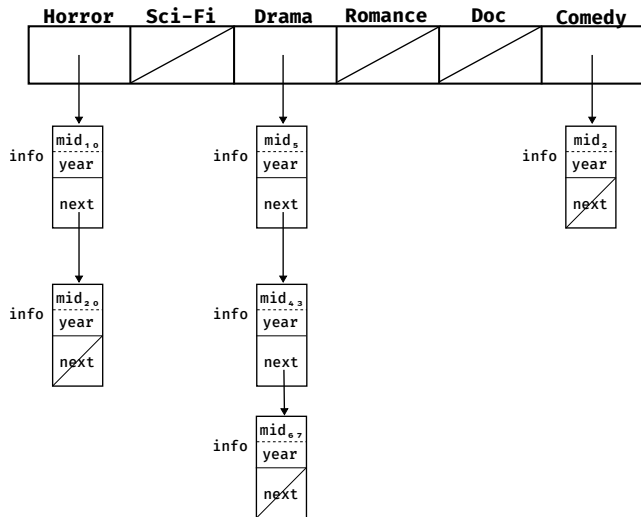
Singly-linked list

- Stores movies of **all categories**
- **Before** they are added to category table
- Sorted, based on **increasing movie ID**

Category Table

An array of singly-linked lists

- Movies moved here from *new releases list*
- One list *per category*
- Sorted, based on **increasing** movie ID



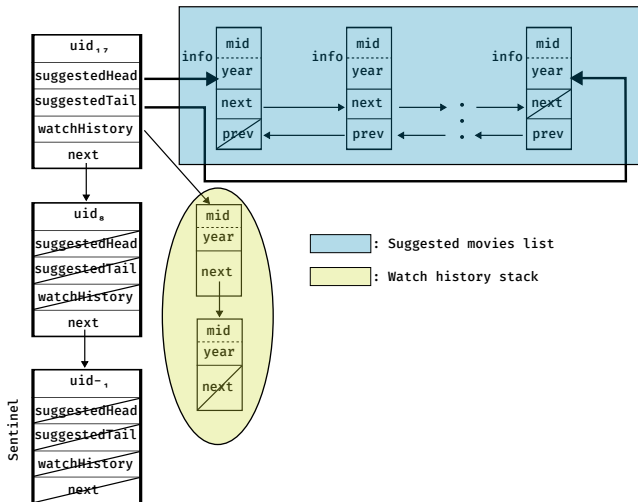
Users

Information per User

- **Unique** identifier, *int*
- *Suggested movies*
 - **Doubly-linked** list
 - *Head* and *tail* pointers
- *Watch history* **stack**

Stored in user list

- **Unsorted**, singly-linked
- With **sentinel** node (*uid -1*)



Events

Register user, Unregister user

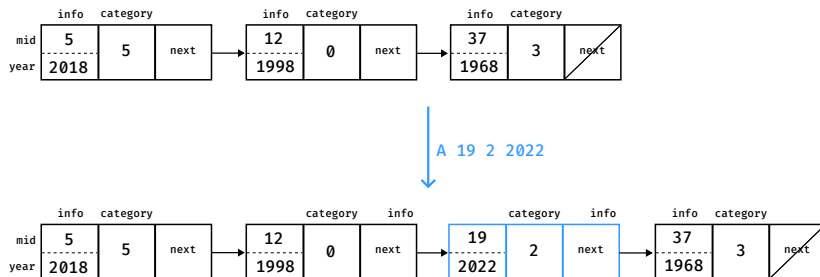
- **Insert** and **remove** user to/from user list respectively
- Check that provided ID is **valid**
- $O(1)$ time complexity insertion!



A mid category year

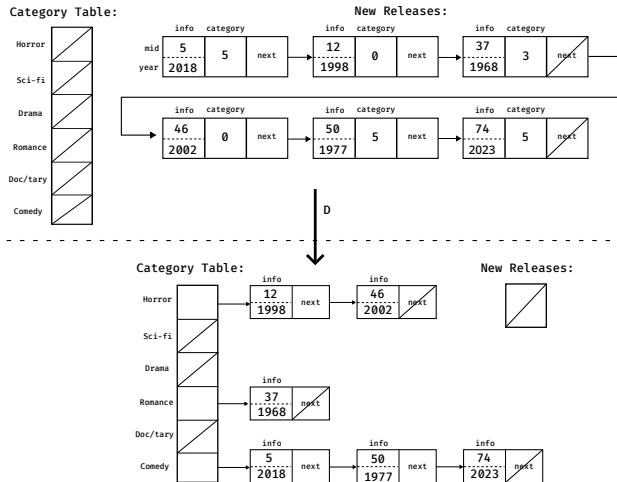
Add new movie

- Insertion to **new releases** list
- List must **remain sorted**!



Distribute new movies

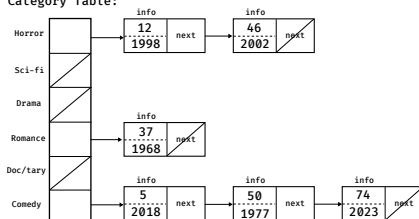
- For each movie in new releases:
 - Insert to **appropriate** category list
 - Remove from new releases list
- Category lists **must be sorted**
- $O(n)$ time complexity
($n = \text{num_new_releases}$)
 - May require auxiliary pointers



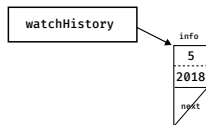
User watches movie

- Locate movie in *category table*
- Push movie to user's **watch history stack**

Category Table:

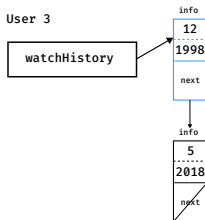


User 3



W 3 12

User 3



Service *suggests* movies to user uid

- Based on **other users' watch history**
- Movies inserted to user uid's **suggested movies**
- **$O(n)$ time complexity** ($n = \text{num_users}$)
- Front and back **alternating insertions**
 - See illustrated example in assignment description

```

1: currFront = uid→suggestedHead
2: currBack = uid→suggestedTail
3: counter = 1
4: for all user ∈ UserList do
5:   if user→uid ≠ uid then
6:     movie = pop(user→watchHistory)
7:     if counter % 2 ≠ 0 then
8:       InsertAsNext(currFront, movie)
9:       AdvanceNext(currFront)
10:    else
11:      InsertAsPrev(currBack, movie)
12:      AdvancePrev(currBack)
13:    end if
14:    counter++
15:  end if
16: end for
17: LinkIfNecessary(currFront, currBack)

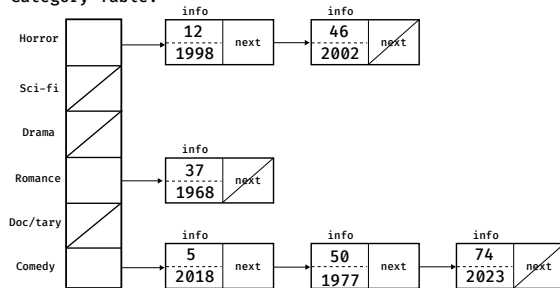
```

Filtered movie search

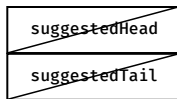
- Iterate category1, category2 lists in category table
- Insert each movie with release year $\geq$ year to new list
- Resulting new list must be sorted
- Finally, append new list to user uid's suggested movies list
- $O(n + m)$ time complexity (n, m : category list sizes)

F uid category1 category2 year

Category Table:

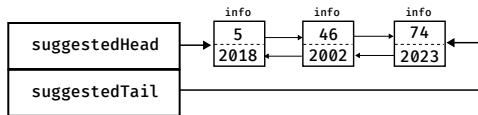


User 3



F 3 0 5 2000

User 3



For a smooth assignment

- Ask questions!
 - How can you learn if you do not ask?
- Use **gdb** for quick **debugging**
 - Great tool to detect **segfaults**
 - Check tutorial on course website (https://www.csd.uoc.gr/~hy240/current/material/assistiveClasses/gdb_tutorial.pdf)
 - (Optional) **valgrind** for **memory leaks**